



MUSIC AND AI

EDITED BY: Alexandra Bonnici, Roger B. Dannenberg, Steven Kemper and
Kenneth P. Camilleri

PUBLISHED IN: Frontiers in Artificial Intelligence



frontiers

Frontiers eBook Copyright Statement

The copyright in the text of individual articles in this eBook is the property of their respective authors or their respective institutions or funders. The copyright in graphics and images within each article may be subject to copyright of other parties. In both cases this is subject to a license granted to Frontiers.

The compilation of articles constituting this eBook is the property of Frontiers.

Each article within this eBook, and the eBook itself, are published under the most recent version of the Creative Commons CC-BY licence.

The version current at the date of publication of this eBook is CC-BY 4.0. If the CC-BY licence is updated, the licence granted by Frontiers is automatically updated to the new version.

When exercising any right under the CC-BY licence, Frontiers must be attributed as the original publisher of the article or eBook, as applicable.

Authors have the responsibility of ensuring that any graphics or other materials which are the property of others may be included in the CC-BY licence, but this should be checked before relying on the CC-BY licence to reproduce those materials. Any copyright notices relating to those materials must be complied with.

Copyright and source acknowledgement notices may not be removed and must be displayed in any copy, derivative work or partial copy which includes the elements in question.

All copyright, and all rights therein, are protected by national and international copyright laws. The above represents a summary only. For further information please read Frontiers' Conditions for Website Use and Copyright Statement, and the applicable CC-BY licence.

ISSN 1664-8714

ISBN 978-2-88966-602-7

DOI 10.3389/978-2-88966-602-7

About Frontiers

Frontiers is more than just an open-access publisher of scholarly articles: it is a pioneering approach to the world of academia, radically improving the way scholarly research is managed. The grand vision of Frontiers is a world where all people have an equal opportunity to seek, share and generate knowledge. Frontiers provides immediate and permanent online open access to all its publications, but this alone is not enough to realize our grand goals.

Frontiers Journal Series

The Frontiers Journal Series is a multi-tier and interdisciplinary set of open-access, online journals, promising a paradigm shift from the current review, selection and dissemination processes in academic publishing. All Frontiers journals are driven by researchers for researchers; therefore, they constitute a service to the scholarly community. At the same time, the Frontiers Journal Series operates on a revolutionary invention, the tiered publishing system, initially addressing specific communities of scholars, and gradually climbing up to broader public understanding, thus serving the interests of the lay society, too.

Dedication to Quality

Each Frontiers article is a landmark of the highest quality, thanks to genuinely collaborative interactions between authors and review editors, who include some of the world's best academicians. Research must be certified by peers before entering a stream of knowledge that may eventually reach the public - and shape society; therefore, Frontiers only applies the most rigorous and unbiased reviews.

Frontiers revolutionizes research publishing by freely delivering the most outstanding research, evaluated with no bias from both the academic and social point of view. By applying the most advanced information technologies, Frontiers is catapulting scholarly publishing into a new generation.

What are Frontiers Research Topics?

Frontiers Research Topics are very popular trademarks of the Frontiers Journals Series: they are collections of at least ten articles, all centered on a particular subject. With their unique mix of varied contributions from Original Research to Review Articles, Frontiers Research Topics unify the most influential researchers, the latest key findings and historical advances in a hot research area! Find out more on how to host your own Frontiers Research Topic or contribute to one as an author by contacting the Frontiers Editorial Office: frontiersin.org/about/contact

MUSIC AND AI

Topic Editors:

Alexandra Bonnici, University of Malta, Malta

Roger B. Dannenberg, Carnegie Mellon University, United States

Steven Kemper, The State University of New Jersey, United States

Kenneth P. Camilleri, University of Malta, Malta

Citation: Bonnici, A., Dannenberg, R. B., Kemper, S., Camilleri, K. P., eds. (2021). Music and AI. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-88966-602-7

Table of Contents

| | | |
|------------|---|--|
| 04 | <i>Editorial: Music and AI</i> | Alexandra Bonnici, Roger B. Dannenberg, Steven Kemper and Kenneth P. Camilleri |
| 06 | <i>Understanding Musical Predictions With an Embodied Interface for Musical Machine Learning</i> | Charles Patrick Martin, Kyrre Glette, Tønnes Frostad Nygaard and Jim Torresen |
| 20 | <i>Computational Creativity and Music Generation Systems: An Introduction to the State of the Art</i> | Filippo Carnovalini and Antonio Rodà |
| 40 | <i>Designing and Evaluating the Usability of a Machine Learning API for Rapid Prototyping Music Technology</i> | Francisco Bernardo, Michael Zbyszyński, Mick Grierson and Rebecca Fiebrink |
| 58 | <i>A Dynamic Representation Solution for Machine Learning-Aided Performance Technology</i> | Jason Palamara and W. Scott Deal |
| 71 | <i>Automated Page Turner for Musicians</i> | André Tabone, Alexandra Bonnici and Stefania Cristina |
| 88 | <i>Creating Music With Fuzzy Logic</i> | Rodrigo F. Cádiz |
| 108 | <i>On the use of AI for Generation of Functional Music to Improve Mental Health</i> | Duncan Williams, Victoria J. Hodge and Chia-Yu Wu |
| 114 | <i>Evolving Musical Sight Reading Exercises Using Expert Models</i> | Charlotte Pierce, Tim Hendtlass, Anthony Bartel and Clinton J. Woodward |
| 134 | <i>Listener Modeling and Context-Aware Music Recommendation Based on Country Archetypes</i> | Markus Schedl, Christine Bauer, Wolfgang Reisinger, Dominik Kowald and Elisabeth Lex |
| 155 | <i>On the Adaptability of Recurrent Neural Networks for Real-Time Jazz Improvisation Accompaniment</i> | Kosmas Kritsis, Theatina Kylafi, Maximos Kaliakatsos-Papakostas, Aggelos Pikrakis and Vassilis Katsouros |



Editorial: Music and AI

Alexandra Bonnici^{1*}, Roger B. Dannenberg², Steven Kemper³ and Kenneth P. Camilleri¹

¹Department of Systems and Control Engineering, University of Malta, Msida, Malta, ²School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, United States, ³Mason Gross School of the Arts, Rutgers, The State University of New Jersey, New Brunswick, NJ, United States

Keywords: music composition, expressive playing, music recommendation, sight-reading, music interfaces, music and mental health, artificial intelligence, machine learning

Editorial on the Research Topic

Music and AI

Computer algorithms have been shaping the music scene since the 1950s. Artificial intelligence, machine learning and computational methods have left their mark not only on the way that music is composed and performed but also on the adoption of new musical notations; different music learning approaches as well as in different marketing strategies which change the way music is consumed.

AI IN THE PRODUCTION OF NEW MUSIC

Computational techniques have been used in a variety of ways for the creation and production of musical compositions with this field predating even digital music synthesis. Algorithmic music composition techniques range from the use of stochastic processes to create music based on random events to learning-based approaches. The paper “**Computational Creativity and Music Generation Systems: an Introduction to the State of the Art**” (Carnovalini and Roda) reviews work extending over six decades, organizing their presentation by methods, ranging from Markov chains to deep networks, and offering a set of open challenges. An extensive bibliography is included. Just as the techniques used to generate the music are varied, so too is the style of music generated, from the creation of written scores to musical accompaniment. “**Evolving Musical Sight Reading Exercises Using Expert Models**” (Pierce et al.) presents a novel evolutionary algorithm for generating monophonic sight-reading exercises in the Western art music tradition. Drawing on expert models of published sight-reading exercises, the evolutionary process draws on six fitness measures to create new exercises designed for specific grade levels of musical instruction. These include target note lengths, target rest lengths, allowable lengths, target intervals, allowable intervals, and melody shape. “**On the Adaptability of Recurrent Neural Networks for Real-Time Jazz Improvisation Accompaniment**” (Kritsis et al.) describes the basic implementation of an artificial jazz accompanist system that provides real-time accompaniment to a human musician soloist, based on a given harmonic description of lead sheet chord symbols. Recurrent Neural Networks are employed both for modeling the predictions of the artificial agent and for modeling the expectations of human intention. Fuzzy logic is a branch of AI that is often overlooked in this age of big data and neural networks. Nevertheless, fuzzy logic can be a powerful tool for modeling and learning music information expressed as signals, parameters or symbols. Fuzzy logic offers a useful framework for expressing models that can assist learning from less data. “**Creating Music with Fuzzy logic**” (Cadiz) offers an introduction to this field and describes a software toolkit created for composition and real-time music applications.

OPEN ACCESS

Edited and reviewed by:

Sriram Natarajan,
The University of Texas at Dallas,
United States

*Correspondence:

Alexandra Bonnici
alexandra.bonnici@um.edu.mt

Specialty section:

This article was submitted to
Machine Learning and
Artificial Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 09 January 2021

Accepted: 13 January 2021

Published: 11 February 2021

Citation:

Bonnici A, Dannenberg RB, Kemper S
and Camilleri KP (2021) Editorial:
Music and AI.
Front. Artif. Intell. 4:651446.
doi: 10.3389/frai.2021.651446

AI TO CREATE EXPRESSIVE MUSIC

Music must be performed expressively to be engaging. Computational musical expression systems learn expressive performance models from examples of human performances to adapt these to the music at hand. Expressive music performance thus alters the "mechanical" or literal performances implied by discrete music notation symbols into more nuanced realizations with alterations in timing, dynamics, timbre, vibrato and other details. **"A Dynamic Representation Solution for Machine Learning-Aided Performance Technology"** (Palamara and Deal) considers the use of AI techniques to interpret discrete dynamic values, such as *p*, *mp*, *f* and *ff* to control parameters that can adapt to performance context.

EASY-TO-USE MUSIC INTERFACES

As machine learning makes its way from the worlds of science, technology and commerce to the arts, there is a need for easy-to-use systems and interfaces that can be applied directly by artists, composers and performers. **"Evaluating the Usability of an API for Rapid Prototyping Music Technology with Interactive Machine Learning"** (Bernardo et al.) considers the problem of supporting designers of creative software projects with tools for machine learning. The study offers insights into both the design of machine learning frameworks and evaluation strategies. Another application of AI is toward intelligent instruments that adapt to or enrich human performance gestures. **"Understanding Musical Predictions with an Embodied Interface for Musical Machine Learning"** (Martin et al.) implements a purposefully simple musical instrument with just one input, a lever controlling pitch, and an internal sequence-prediction algorithm based on a recurrent neural network and trained on human performances. The study sheds light on how humans interact with predictive gestural interfaces. In **"Automated Page Turner for Musicians"** (Tabone et al.), the authors describe using eye-gaze tracking to enable hands-free page turning, employing Kalman filtering to balance the music-reading model and the noisy eye-gaze data, thus obtaining stable and reliable page-turning.

AI IN MARKETING OF MUSIC

Music providers are also using AI to learn the musical preferences of consumers to provide customised playlists based on listening patterns. **"Listener Modeling and Context-aware Music Recommendation Based on Country Archetypes"** (Schedl et al.) considers how music preferences are shaped by the country of the listener. The study uses unsupervised learning to suggest nine archetypes or clusters of listening preferences and shows that recommendation systems can be enhanced by using country information.

MUSIC AND HEALTHCARE

Music has many implications for health care, and computer-generated music is of special interest due to the possibility of making music for specific functions or according to particular therapeutic constraints. **"On the use of AI for Generation of Functional Music to Improve Mental Health"** (Williams et al.) uses machine learning to create music targeting a specific physiological response. This work suggests a new direction for the evaluation of music generation systems as well as future applications such as games and health care.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2021 Bonnici, Dannenberg, Kemper and Camilleri. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Understanding Musical Predictions With an Embodied Interface for Musical Machine Learning

Charles Patrick Martin^{1,2,3*}, Kyrre Glette^{2,3}, Tønnes Frostad Nygaard² and Jim Torresen^{2,3}

¹ Research School of Computer Science, Australian National University, Canberra, ACT, Australia, ² Department of Informatics, University of Oslo, Oslo, Norway, ³ RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion, University of Oslo, Oslo, Norway

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Rinkaj Goyal,
Guru Gobind Singh Indraprastha
University, India
Chetan Tonde,
Amazon, United States

*Correspondence:

Charles Patrick Martin
charles.martin@anu.edu.au

Specialty section:

This article was submitted to
Machine Learning and Artificial
Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 31 October 2019

Accepted: 07 February 2020

Published: 03 March 2020

Citation:

Martin CP, Glette K, Nygaard TF and
Torresen J (2020) Understanding
Musical Predictions With an Embodied
Interface for Musical Machine
Learning. *Front. Artif. Intell.* 3:6.
doi: 10.3389/frai.2020.00006

Machine-learning models of music often exist outside the worlds of musical performance practice and abstracted from the physical gestures of musicians. In this work, we consider how a recurrent neural network (RNN) model of simple music gestures may be integrated into a physical instrument so that predictions are sonically and physically entwined with the performer's actions. We introduce EMPI, an embodied musical prediction interface that simplifies musical interaction and prediction to just one dimension of continuous input and output. The predictive model is a mixture density RNN trained to estimate the performer's next physical input action and the time at which this will occur. Predictions are represented sonically through synthesized audio, and physically with a motorized output indicator. We use EMPI to investigate how performers understand and exploit different predictive models to make music through a controlled study of performances with different models and levels of physical feedback. We show that while performers often favor a model trained on human-sourced data, they find different musical affordances in models trained on synthetic, and even random, data. Physical representation of predictions seemed to affect the length of performances. This work contributes new understandings of how musicians use generative ML models in real-time performance backed up by experimental evidence. We argue that a constrained musical interface can expose the affordances of embodied predictive interactions.

Keywords: musical performance, interface, mixture density network (MDN), recurrent neural network (RNN), creativity, predictive interaction, embodied performance

1. INTRODUCTION

It is well-known that music is more than just what you hear. Movements, or gestures, also contribute to musical communication (Jensenius et al., 2010). Most acoustic music performance involves control gestures to operate instruments, but performers also use expressive auxiliary gestures to communicate musical expression (Broughton and Stevens, 2008). In contrast, machine-learning models of music often exist outside the world of physical performance with music represented symbolically or as digital audio, both forms abstracted from musicians' physical gestures. If these models are to be applied in real-time musical performance, then it is crucial to know whether performers and listeners understand predicted musical information and how they use it. In this work, we consider how a recurrent neural network (RNN) model of simple music gestures may be integrated into a physical instrument so that predictions are sonically and



FIGURE 1 | The Embodied Music Prediction Interface (EMPI) prototype. The system includes a lever for a performer's physical input (left side) and a motor-controlled lever for physical output, a speaker, and Raspberry Pi. This system represents a minimum set of inputs and outputs to experiment with embodied predictive interaction. A demonstration video can be viewed in the **Supplementary Material**.

physically entwined with the performer's actions. Our system, the embodied musical prediction interface (EMPI, see **Figure 1**), includes a lever for physical input from a performer, and a matching motorized lever to represent predicted output from the RNN model. We use this interface to investigate how performers can make use of musical machine-learning predictions in real-time performance, and whether physical representations might influence their understanding of such an instrument.

Rather than predicting symbolic music, such as MIDI notes, our RNN model predicts future musical control data—the physical positions of the EMPI's lever—in absolute time. These predictions can thus be represented both through the sound produced by predicted movements as well as through physical actuation of these control elements. The goal is to train a machine-learning model that can improvise on a musical instrument directly, rather than compose notes. To examine the potential of this idea, our EMPI system simplifies musical interaction to the barest requirements: just one dimension of continuous input and output which both control the pitch of a synthesized sound. By reducing the musical prediction problem, we seek to expose the performers' understanding of and adaptation to a musical ML system.

The EMPI system includes a single-board computer for machine-learning calculations and synthesis, one lever for physical input, one for actuated physical output, and a built-in speaker. It is completely self-contained, with power supplied by a USB power bank. The machine-learning model is a mixture density RNN trained to predict the performer's next physical input action and the time at which this will occur (Martin and

Torresen, 2019). The system includes three different models: one trained on a corpus of human-sourced performance data; one trained on synthetically produced movements; and one trained on noise, or movements that are uncorrelated in time. Although multiple interaction designs could be possible, we focus here on applying predictions to continue a performer's interactions (Pachet, 2003), or to improvise in a call-and-response manner.

Embedded and self-contained instruments are important current topics in digital musical instrument design (Moro et al., 2016); however, these instruments usually do not include predictive capabilities. On the other hand, musical AI is often focused on composition using high-level symbolic representations (e.g., Sturm and Ben-Tal, 2017), and not the interactive or embodied factors (Leman et al., 2018) of music perception and creation. In this work, an embedded instrument design is combined with a novel, embodied approach to musical AI. This combination of embodied musical prediction with interaction allows us to explore musical AI within genuine performance environments, where movement is entangled with sound as part of musical expression.

We evaluated the success of this system through examination of generated data from these trained models as well as through a study of 72 performances made with this system under controlled conditions with 12 performers. This evaluation sought to identify whether the actions of the different predictive models are understandable to the performers, and whether they perceive useful musical relationships between their control gestures, and the model's response. We also investigated whether embodied interactions with this system's physical output improves or distracts from these understandings.

Our survey findings show that, of the three models, the performers assessed EMPI's human model as most related to their performance, most musically creative, more readily influenced and more influential on their playing than the other models. However, interviews with participants revealed they also saw value in the synthetic and even noise model based on their interactive affordances and musical styles. While performers were split on opinions regarding the physically embodied response lever, the length of improvisations suggests that the lever did effect their perceptions of the model's actions. Our study has demonstrated that a constrained, ML-enabled musical interface can afford a variety of creative performance styles. The performer's understanding of the different ML models seems to have a significant bearing on how they interact with the interface. We argue that physically actuated indicators, although potentially distracting for some performers, can expose the actions of an embodied music model, and encourage users to explore new ways of performing.

2. BACKGROUND

Musical instruments are not typically predictive; instead, definitions of interactive music systems focus on behavior in reaction to gestural input (Rowe, 1993). The advent of electronic musical instruments including powerful computers has allowed

experiments with instruments that are able to make intelligent use of the musical context in which they are used. This has been discussed since at least the early 1990s (Pressing, 1990), but has been extended in recent years with the development and popularity of accessible machine learning frameworks for understanding physical gestures in performance (Fiebrink, 2017). Artificial intelligence techniques can imbue a musical interface with a kind of self-awareness (Lewis et al., 2016; Nymoen et al., 2016), allowing them to act predictively, rather than in reaction to a performer.

The question of how to make best use of musical predictions, particularly from a performance perspective, remains open. Present work in musical deep neural networks is often focused on symbolic music generation (Briot et al., 2020), on the modification (Roberts et al., 2018) or in-filling (Huang et al., 2017) of given musical sequences, and creating musical digital audio (Engel et al., 2019). Examples of these neural networks have recently been embedded into digital audio workstation software to aid users during music composition (Roberts et al., 2019). Predictions are therefore used to make *more* music, or *better* music. We do not stray far from this characterization in the present work, but rather consider musical data to include gestural feedback, as well as more typical notes and sounds. Where a typical musical interface maps gestures into sounds, a predictive interface can also map current gestures into future gestures and represent these gestures themselves as well the sounds they might produce (see **Figure 2**).

Music has many representations, including lead sheets, scores, and recorded audio with varying levels of specificity over the musical work recorded (Davies, 2005). The machine learning models mentioned above have focused on generating music represented either symbolically (e.g., as MIDI notes), or as digital audio, a more-or-less finalized representation. In this work, we use control gestures to represent musical performance; a format that is more open than digital audio, but more specific than MIDI notes, especially in terms of precise expression. As argued in section 1, control and auxiliary gestures are important parts of musical performance (Jensenius et al., 2010). Further, an embodied view is required to understand how we perceive and perform music (Leman et al., 2018). Some machine learning models do predict embodied representations of artistic data. For instance, *SketchRNN* predicts pen movements to draw images (Ha and Eck, 2017), and *SPiRAL* generates instructions for a paint program to generate realistic images (Ganin et al., 2018). This concept has also been applied to musical sketches in *RoboJam* (Martin and Torresen, 2018), and the IMPS system (Martin and Torresen, 2019), which applied similar mixture density RNNs as in the present research to predict movements on a touchscreen or of arbitrary numbers of control values through time. One field where embodied music is crucial is musical robotics (Bretan and Weinberg, 2016), although physical motions in this field are usually not the direct predictions of an ML system, but programmed in response to decisions to actuate certain notes on an acoustic instrument.

The EMPI system in this work is an example of an embedded and self-contained computer music interface. Handheld and self-contained electronic instruments, such as Michel Waisvisz'

CrackleBox (Waisvisz, 2004), the toy *Stylophone* (McNamee, 2009), or Korg's more recent *monotron* synthesizers have been popular since the late 1960s. While most computer music instruments involve a laptop computer externally connected to a controller interface, Berdahl and Ju (2011) argued that it was advantageous to embed a single-board computer (SBC), such as a Raspberry Pi inside the musical instrument to create an integrated and portable musical instrument. The resulting *Satellite CCRMA* system used a Raspberry Pi with a USB-connected microcontroller (Berdahl et al., 2013). The *Bela* system (Moro et al., 2016) developed this idea, with an integrated hardware extension to the Beaglebone Black platform providing an embedded instrument platform with high audio and sensor performance (McPherson et al., 2016).

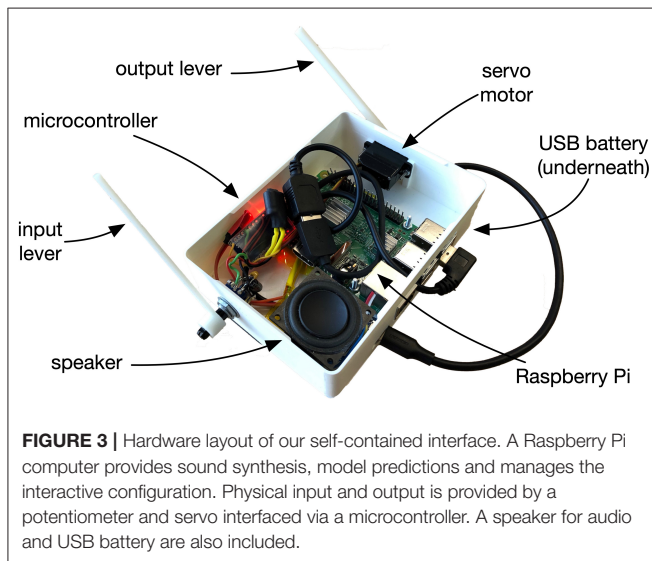
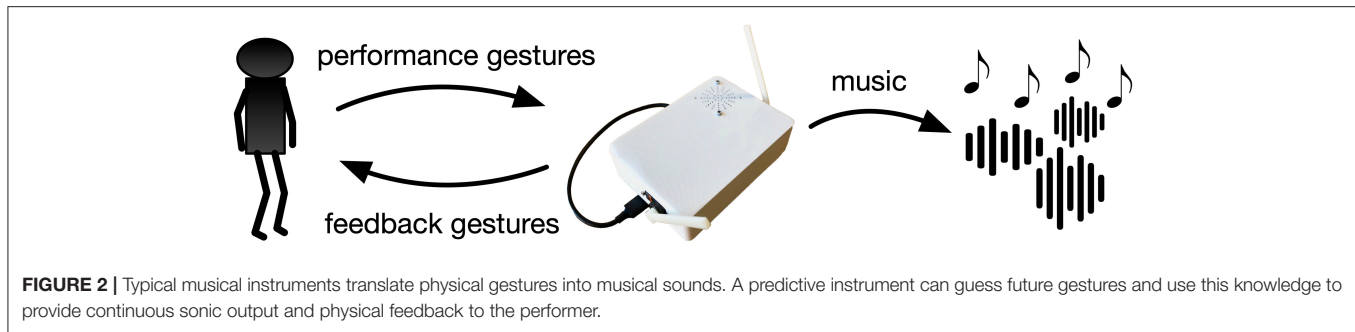
Apart from technical advantages, embedded instrument designs can be artistically advantageous in terms of enabling exploration through physical manipulation (Reus, 2011) and even live hardware hacking (Zappi and McPherson, 2014). Self-containment can also enable new research methodologies. Gurevich et al. (2012) explored a constrained self-contained musical interface. In this case, the self-contained nature of the device allowed it to be distributed to participants and explored by them on their own terms.

So far, there are few examples of embedded computer music interfaces that include music prediction ANNs. This is despite significant interest in ML-prediction on internet of things (IoT) or edge computing platforms (Ananthanarayanan et al., 2017). In one of the only present examples, Næss and Martin (2019) demonstrated an LSTM-RNN-driven embedded music generator based on a Raspberry Pi. This work showed that RNN prediction is practical on an embedded system, and the resulting self-contained interface allows the music generation system to be examined by musicians. In the present research, we also use a Raspberry Pi as the embedded computing platform for an RNN-based musical prediction system. This work goes further by exploring musical predictions at the gestural, rather than symbolic level of representation. Our system embeds a predictive model in a system with physical, as well as sonic output. This allows us to examine both musical expression and predictive interaction in a real-time performance situation.

3. SYSTEM DESIGN

Our Embodied Musical Predictive Interface (EMPI), shown in **Figure 1**, is a self-contained musical interface. EMPI is a highly constrained musical interface, with only one dimension of continuous input. The EMPI's matching physical output allows it to represent the embodied predictive process to a human user. Its self-contained form-factor allows musicians to explore and integrate predictive musical interaction into different scenarios.

The physical design of EMPI is focused on hand-held and self-contained interaction. The 3D-printed enclosure includes a Raspberry Pi model 3B+, one lever for input, a speaker and servo-controlled lever for physical output. A 5,000 mAh USB power bank is attached to the base of the enclosure. The input and output levers are interfaced to the Raspberry Pi through



its USB ports and a small ATmega 32U4 microcontroller board. The speaker and a small amplifier is connected directly to the Raspberry Pi's audio output. A system diagram shows these components in **Figure 3**.

The software aspects of the system provide musical interaction and prediction capabilities. The most important of these is a low-level internal model of performer interactions: a sequence of real-valued potentiometer positions, along with a time-delta value. To model this data, we use a 2D mixture density RNN that predicts the position, and the time, of the next user input. Various trained models can be used with this network based on either real-world or synthetic training data. It should be noted that RNN predictions are computed by the EMPI's Raspberry Pi, not an external system.

The prediction model is implemented in Python using TensorFlow, and applies a special case of our Interactive Music Prediction System (IMPS) which has been previously described (Martin and Torresen, 2019). The IMPS system contains the predictive MDRNN model, and communicates with Pure Data over OSC to receive user interactions and send sound and servo commands. Pure Data synthesizes the sound output and communicates with the microcontroller using MIDI over USB. This system is configured for call-and-response performance. When the performer is playing, their interactions are used to

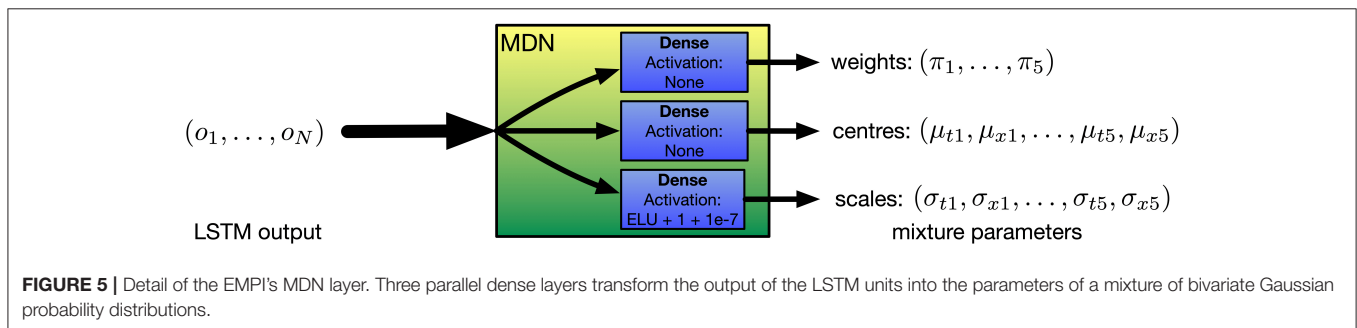
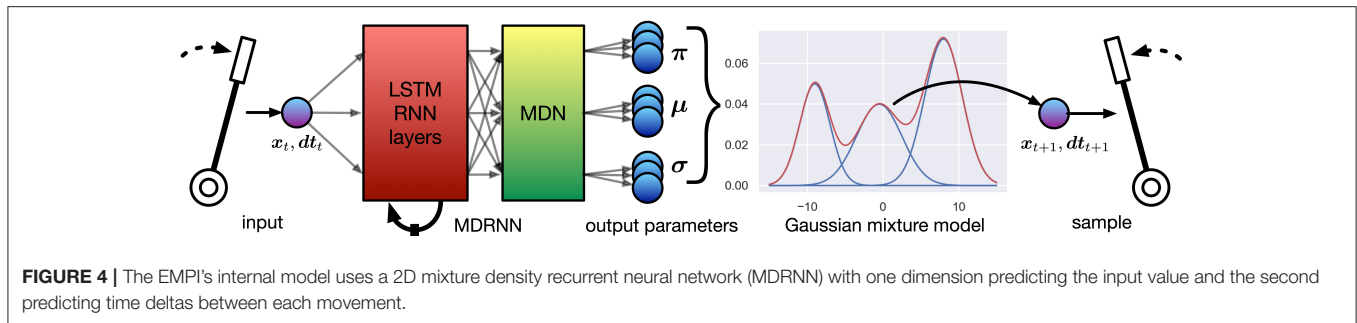
condition the MDRNN's memory state. If they stop playing (after a threshold of 3 s), the MDRNN attempts to continue where they left off, generating more interactions until the performer plays again. The EMPI's hardware design and software, including trained models, are open source and can be found online (Martin, 2019a).

3.1. Predictive Model

The EMPI uses a mixture density recurrent neural network to predict future input on the lever. This architecture combines a recurrent neural network with a mixture density network (MDN) (Bishop, 1994) that transforms the output of a neural network to the parameters of a mixture-of-Gaussians distribution. Real-valued samples can be drawn from this distribution, and the number of mixture components can be chosen to represent complex phenomena. The probability density function (PDF) of this distribution is used as an error function to optimize the neural network. In contrast, the softmax layer used in many music RNNs parameterizes a categorical distribution between a set number of discrete classes.

The expressive capacity of MDRNNs has been previously exploited to generate creative data, such as handwriting (Graves, 2013) and sketches (Ha and Eck, 2017). This architecture has only recently been applied to musical interaction data, for instance in *RoboJam* to continue musical touchscreen interactions (Martin and Torresen, 2018), and in *IMPS* as a general model for musical interaction data (Martin and Torresen, 2019). For the EMPI interface, an MDRNN model has the advantage of delivering real-valued samples for lever position and time, as well as a tuneable learning capacity in terms of the RNN configuration (width and number of LSTM layers) and the number of mixture components. This allows us to generate movements in absolute time and to potentially learn complex behaviors from the lever movements.

EMPI's MDRNN is a special case of the one described in *IMPS* (Martin and Torresen, 2019), and is illustrated in **Figure 4**. The neural network has two inputs. One input is for the current lever position (x_t), and the other for the time since the previous movement (dt_t). These inputs are fed through two layers of long short-term memory (LSTM) units and into the MDN layer which outputs the mixture parameters. Each of the K components of the mixture is a bivariate Gaussian distribution with a diagonal covariate matrix with centers (μ_{xk}, μ_{tk}) and scales (σ_{xk}, σ_{tk}). A set of mixing parameters (π_1, \dots, π_K), forms a categorical distribution between the mixture components. In our case, we



set the number of mixture components $K = 5$ following previous work (Martin and Torresen, 2019).

The MDN layer is provided by the Keras MDN Layer (v0.2.1) library (Martin, 2019b). This layer transforms the outputs of the LSTM layers into appropriate parameters to form the mixture distribution (see **Figure 5**). The outputs of the LSTM layers are fed into parallel dense layers that output the centers, scales, and weights of the mixture distribution, respectively. No activation function is used for the centers and weights. The exponential linear unit (ELU) activation function (Clevert et al., 2016) is used for the scales, with the output offset by $1 + 10^{-7}$. This ensures that the scales are positive and non-zero while providing gradients at very small values (as recommended by Brando, 2017). To train this neural network, the PDF of the mixture model is constructed using Mixture and MultivariateNormalDiag distributions from the TensorFlow Probability library (Dillon et al., 2017) to provide a likelihood function that the training target was drawn from the mixture distribution predicted by the neural network. The negative log of this likelihood can be used as a loss value for gradient descent to optimize the neural network's weights. Further discussion of this procedure can be found in Bishop's work (Bishop, 1994).

To sample from the parameters output by the MDRNN, first, a mixture component is chosen by sampling from the categorical distribution. Then, this chosen mixture component is sampled to produce an output value. Similarly to other generative RNNs, the sampling diversity, or temperature, can be altered to draw more or less conservative choices. The π_k form a categorical model that can be adjusted with the usual temperature modification in the softmax function (Hinton et al., 2015, see Equation 1). The covariance matrix can also be scaled to produce a similar effect. This process yields a sample (x_{t+1}, dt_{t+1}) , representing

a prediction of the next lever position and time at which it could occur. By feeding this sample back into the MDRNN, a continuous stream of lever movements can be generated.

3.2. Sound Design

The digital synthesis routine for EMPI runs in Pure Data so a variety of mappings between lever motion and output sound are possible. In our configuration, Pure Data receives one value from the input lever (its position as a MIDI continuous control value), and one from the predictive model's virtual lever. This data is only sent when either lever's position changes, this is similar to the implementation of a fader on a MIDI control surface. We chose to use the lever positions to control pitch. The amplitude of the sound is controlled by an envelope that is only sustained as long as the lever continues to move. This means that rhythmic performance is possible (albeit with small glissandi) by tapping the lever slightly while allowing the sound to diminish in between each movement.

We experimented with controlling a variety of sounds from the levers, such as simple tones, plucked strings (reminiscent of a harp glissando), sample playback, and formant synthesis. For this research, we settled on a simple 4-operator FM synthesis routine with a slight change to the tone controlled by having separate envelopes on modulation and carrier oscillators. Similarly, while it is possible to have dramatically different sounds on the input and output levers, we used the same synth routine (separate voices), with the EMPI's virtual lever tuned one octave lower. This arrangement allows the sounds to be distinguished as different voices, but recognized as coming from the same source.

3.3. Data

We have experimented with models based on three sources of training data: (1) a collection of solo improvised recordings using

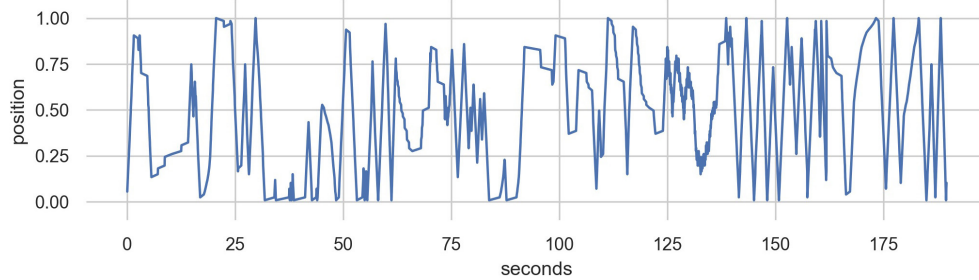


FIGURE 6 | Excerpt from a 10-min human-sourced improvisation with the input lever. This performance was part of the training data for the EMPI's MDRNN model.

the EMPI; (2) synthetic data generated from simple waveforms; and (3) uniform noise. The human-sourced data was collected on the EMPI hardware in “human only” mode where the human input was directly linked to a synthesized sound with no input from the internal model. The improvised performances were completely unconstrained and included data from the entire input range of the lever, periods of no interaction (rests), as well as sweeps and movements in different speeds and rhythms. The improvisation was performed by the first author and an excerpt example from the data is shown in **Figure 6**. This training dataset is available as part of the EMPI source code (Martin, 2019a).

The synthetic data was generated to represent plausible lever motions in repetitive patterns. To generate these, a sequence of time-steps was drawn stochastically from a normal distribution with mean and standard deviation identical to the human-sourced improvisation¹. This sequence of time-steps was then fed through sine, square, and triangle wave functions with frequencies at five steps between 0.1 and 1.1 Hz to generate the input values. In total, 10,000 datapoints were generated for each function and frequency resulting in 150,000 total datapoints. The noise data associated a uniformly sampled random number (between 0 and 1) for each of 30,000 time-steps drawn by the same method. Excerpts from the data generated by sine, square, and triangle waves, as well as noise, are shown in **Figure 7**.

The three sources of data were used to train separate models for the EMPI that are used in the experiments described in section 4. The rationale for using three different models was to explore the creative utility of models based on both human-sourced and synthetically generated data. While the synthetic data is a simple behavior it could potentially represent an appealing and recognizable movement to a performer. In contrast, the noise dataset was not intended to be appealing, rather it was intended to have no recognizable behavior.

4. EVALUATION

Our evaluation of EMPI is focused on the generative potential of the ML models embedded in the device, and the experience of human performers who interact with it. We first discuss the ML models in the abstract and then describe the results

of a human-centered experiment with the EMPI where twelve participants each perform six improvisations under different experimental conditions.

4.1. Machine Learning Models

In this section we evaluate the performance of the mixture density RNN architecture and three models applied in the EMPI system. We performed a small training experiment to ascertain an appropriate size of model for the datasets that we used, and generated unconstrained performances from each model to observe what its behavior might be like in performances.

4.1.1. Training

Previous research has suggested that smaller MDRNNs—i.e., with 64 or even 32 LSTM units in each layer, might be most appropriate for modeling small amounts of musical data for integration into an interactive music system (Martin and Torresen, 2019). We trained EMPI's MDRNN models with 32, 64, 128, and 256 units in each LSTM layer to ascertain the best accuracy in terms of reproducing held-out examples from the dataset. Each candidate model used two layers of LSTM units and was trained on sequences that were 50 datapoints in length. Training was conducted using the Adam optimizer with a batch size of 64 and with 10% of training examples held out for validation. For each model, the number of mixture components was held static at 5.

The human dataset contained 75,262 interaction events, corresponding to 65 min of interaction with the EMPI system. The noise dataset included 30,000 interaction events, and the synth dataset included 150,000 interaction events to allow for 10,000 points with each of the 15 signal variations.

The training and validation set loss over this training process for the human dataset are shown in **Figure 8**. Over the 100 epochs of training on human-sourced data, the 32-unit MDRNN produced the lowest validation loss. For this reason, and also out of concern for speed of computation on the Raspberry Pi, this size of MDRNN was chosen for our experiments below. The noise and synth models used the same size MDRNN. To avoid overfitting, for each dataset we selected the model with the lowest validation loss achieved during these 100 epochs of training. These models were used for the generation experiments below and in our performer study.

¹The human data above was found to have a mean time-delta of 0.045 s with S.D. 0.184.

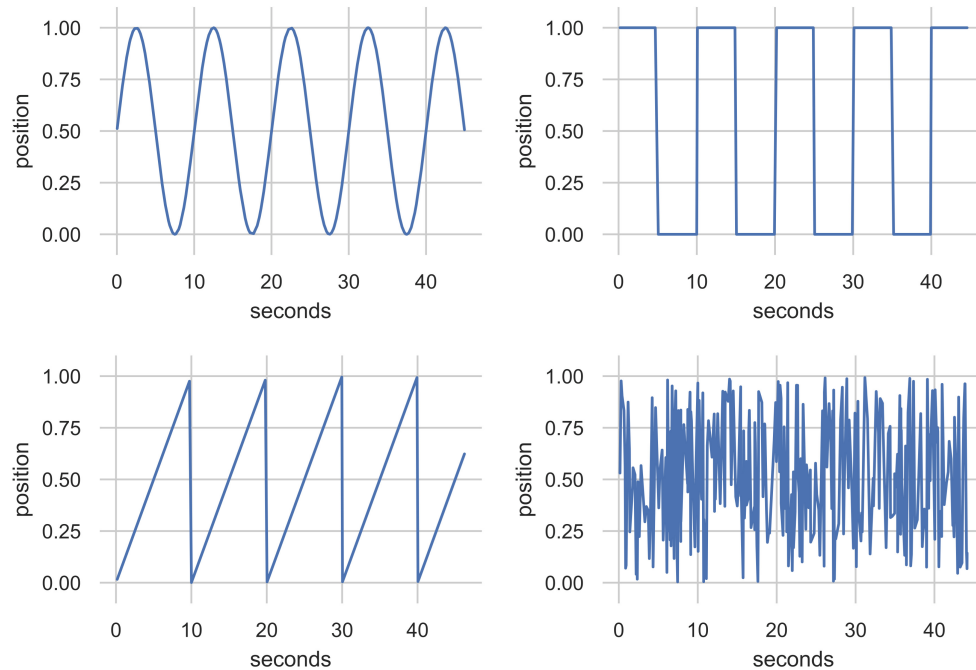


FIGURE 7 | Excerpts from a synthesized data corpus created using stochastically sampled time steps. The function generators are sine, square, and triangle at 0.1 Hz and uniform noise. These data were used as an alternative training data source for the EMPI's MDRNN model.

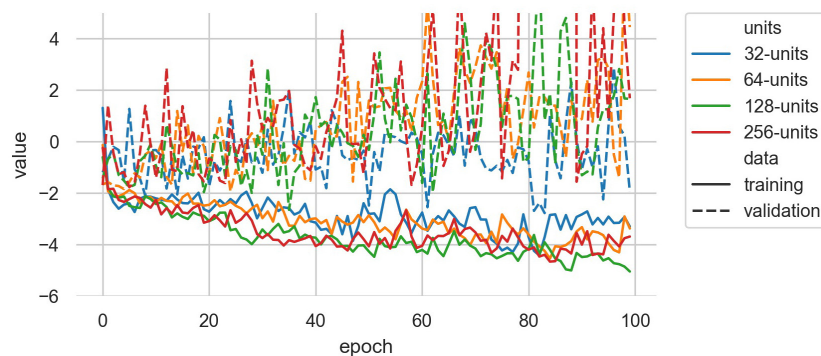


FIGURE 8 | Training data loss and validation data loss while training the human-sourced EMPI model with different size MDRNN architectures. The 32-LSTM-unit MDRNN produced the lowest validation loss and this architecture was used for all EMPI models.

4.1.2. Generation

To demonstrate the potential output of the RNN models we generated sample performances in an unconstrained manner—starting with an uninitialized memory state and random first value, and linking output to input for 500 prediction steps. Temperature settings of 1.1 for the categorical distribution and 0.1 for the multivariate Gaussian's covariate matrix were chosen by trial-and-error. The results of this experiment are shown for each of the three models (human, synthetic, and noise) in **Figure 9**.

The output of the human model seems comparable with the human-sourced dataset (see **Figure 6**). The MDRNN captures a mix of behaviors, such as full back-and-forth motions,

small fast movements, and stepping motions with pauses in between movements. The synth model produced output that, similarly to the training data, moves back-and-forth through the whole range of motion with the ability to change its rate of movement. The wave shape seems to change somewhat, but does not deviate from a roughly sinusoidal pattern. The noise model produces unpredictable patterns as expected. Rather than generate uniformly random outputs over the range of the motion, it seems to alternate between the upper and lower extremes with random movements around the middle.

One notable difference between the models is that the human model produces movements at a finer temporal granularity. While 500 samples yields 70 s of movement from the noise and

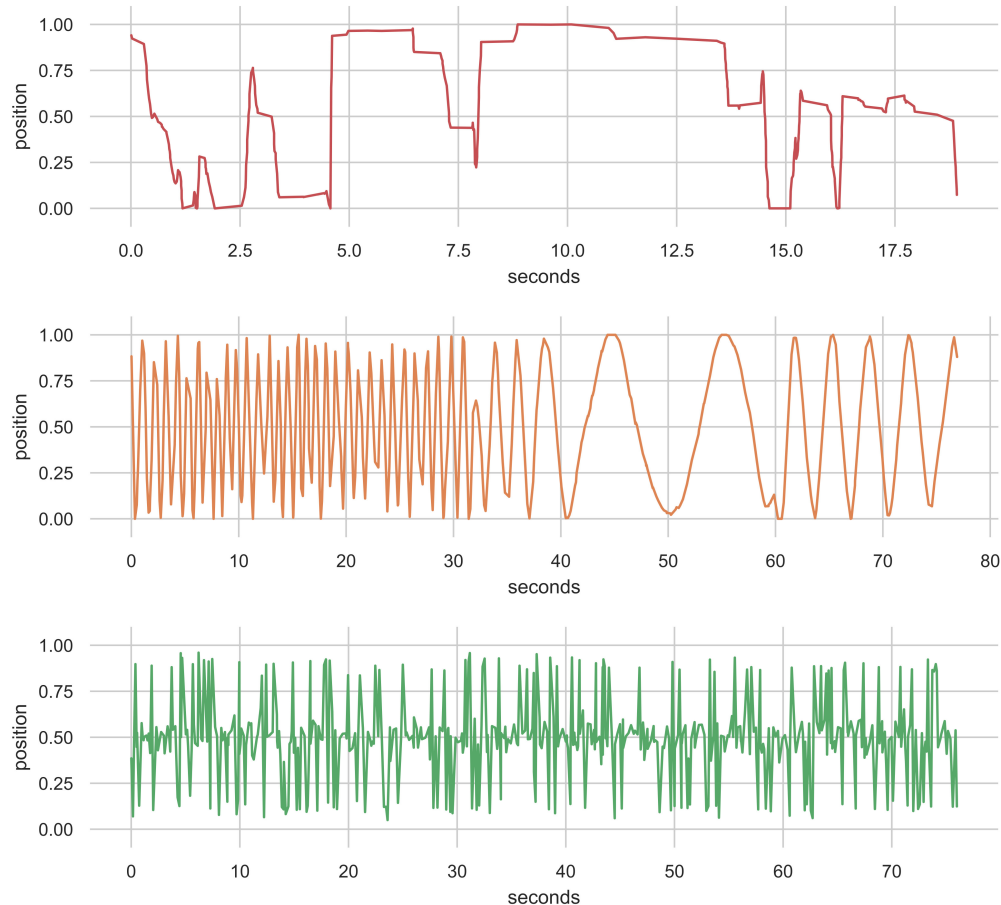


FIGURE 9 | 500 Datapoints from the 32-unit MDRNN models in generation mode starting with an uninitialized memory state and a random starting point. The human-, synthetic-, and noise-based models are shown from top to bottom.

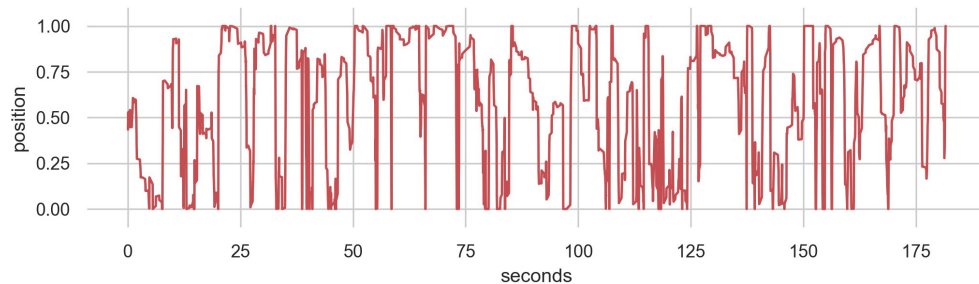


FIGURE 10 | 4,500 Datapoints from the 32-unit MDRNN trained on human data resulting in 180 s of performance.

synth models, only 20 s is produced from the human model. This difference becomes apparent in performance with these models as the human model moves much more smoothly than the other two. A longer performance with the human model, produced by sampling 4,500 datapoints, is shown in **Figure 10**. This shows that the model often focuses on particular areas of the control range for around 10 s before changing to back-and-forth behaviors or moving to a different location. While the long-term

structure of the real human performance is not represented, the local structure seems to be reasonably convincing even with this small MDRNN model.

Performance with the three models (see video in **Supplementary Material**) shows that the noise model produces a consistent but unpredictable pattern, unaffected by any user input. The synth model starts where the user stops, and continues back-and-forth motion. This model can be controlled somewhat

by feeding in particularly fast or slow movements, which are matched by the MDRNN model. The human model generates smoother movements that sounds most like normal user inputs. Although it starts in the same location as the user, it seems more difficult to control with different styles of playing than the synth model. All three models appear to be stable and computationally tractable for extended performances on the EMPI's Raspberry Pi.

4.2. Performer Study

A study with performers was undertaken to ascertain the effects of the three different models and the absence or presence of physical feedback on their perception of the musical interaction experience. The study took the form of a structured improvisation session where participants performed six short improvisations with the EMPI system under different conditions.

Two independent factors were explored in this study. The first was the *model* that the EMPI device used to make predictions; the three models tested were trained with either human-, synthetic-, or noise-sourced data. The second factor was the *feedback* with the physically-actuated arm either enabled or disabled. These conditions were combined leading to six instrument states and each participant improvised under each of these. The study can be characterized as a two-factor within-groups experiment.

4.2.1. Participants

Participants for the study were recruited from the music and computer science communities at the Australian National University. Twelve respondents (six female, six male) were chosen to participate based on availability and experience with musical performance.

4.2.2. Procedure

The study sessions took the structure of research rehearsals (Martin and Gardner, 2019) in that the participants were asked to perform six short improvisations with each one followed by a written survey and the whole session concluded with an interview. The study environment is shown in **Figure 11**. The improvisations were finished when the performer determined that they wanted to stop by signaling the researcher, or at a maximum length of 5 min. Each participant's six improvisations was performed with one of the instrument states. The exposure to different states was ordered following a Williams (1949) design to ensure balance with respect to first-order carryover effects. This required six different orderings, each of which was replicated with two different participants.

The collected data consisted of audio, video, and interaction data recordings of the session, a semi-structured interview at the end of the session, and a short written Likert-style survey after each improvisation. The written surveys had 8 questions with each recorded on a 9-point rating scale with labels only on the extremes and midpoint: "Strongly Disagree" (1), "Neutral" (5), "Strongly Agree" (9). The survey questions were as follows:

1. I understood the ML model's responses (*understood*).
2. The responses were related to my performance (*related*).
3. The responses had a high musical quality (*quality*).
4. The responses showed musical creativity (*creativity*).
5. The responses influenced my playing (*inf-play*).

6. My playing influenced the responses (*inf-resp*).
7. The ML model enhanced the performance (*enh-perf*).
8. The ML model enhanced my experience (*enh-exp*).

4.2.3. Survey Results

The distributions of responses to each question are shown in **Figure 12** and the data can be found in the **Supplementary Material**. Responses to the survey questions were analyzed with an aligned rank transform (ART) and two-way mixed-effects ANOVA procedure. This procedure was used to establish significance of main and interaction effects due to the two factors (*model* and *feedback*). The ART-ANOVA was performed in R using the ARTool library v0.10.6 (Kay and Wobbrock, 2019). This procedure was used as it has been recommended as appropriate for factorial HCI studies with non-parametric data (Wobbrock and Kay, 2016), such as this experiment. *Post-hoc* testing via Holm-corrected paired *t*-tests were performed to establish significant differences between responses to individual conditions.

The ART-ANOVA procedure revealed that the ML model had a significant effect on responses to five of the eight questions; these are shown in **Table 1**. The model had a significant effect on how participants rated the relation between responses in their performance, the musical creativity of responses, whether responses influenced their playing and vice-versa, and whether the ML model enhanced the performance.

The presence or absence of the servo-actuated lever did not have any significant effects on the survey results. For Question 6, "My playing influenced the responses," a minor effect [$F_{(1,55)} = 2.93$, $p < 0.1$] was observed. The distribution of responses here (see **Figure 12**) show that participants seemed to perceive that they had more influence over the response when the physical actuation was present.

As we detected significant effects of the ML model using the ART-ANOVA procedure, *post-hoc* Holm-corrected paired *t*-tests were used between the results for each ML model to reveal which had led to significantly different responses to these questions. For Question 2, participants reported that the responses were more related to their performance with the human model than the synth model and that the noise model was least related. The differences were significant ($p < 0.05$) for all three models for this question with the human model rated as most related, then synth, then noise. The musical creativity (Q4) of responses was rated significantly higher with the human model than for the other two ($p < 0.05$). The participants reported significantly more influence (Q5) from the human model than from the synth model ($p < 0.01$), but the noise model's influence was not rated significantly differently to the other two. The performers rated their own degree of influence over the human model (Q6) significantly more highly than both the synth and noise models. The noise model was also rated as providing significantly less enhancement (Q7) to the performances than with the human model ($p < 0.05$).

The survey results tell us that performers perceived the ML model as making significant impacts on their performances while the physical feedback only had a minor effect on the participants perception of influence over the responses. The *post-hoc* tests



FIGURE 11 | A participant performing with the EMPI during a study session.

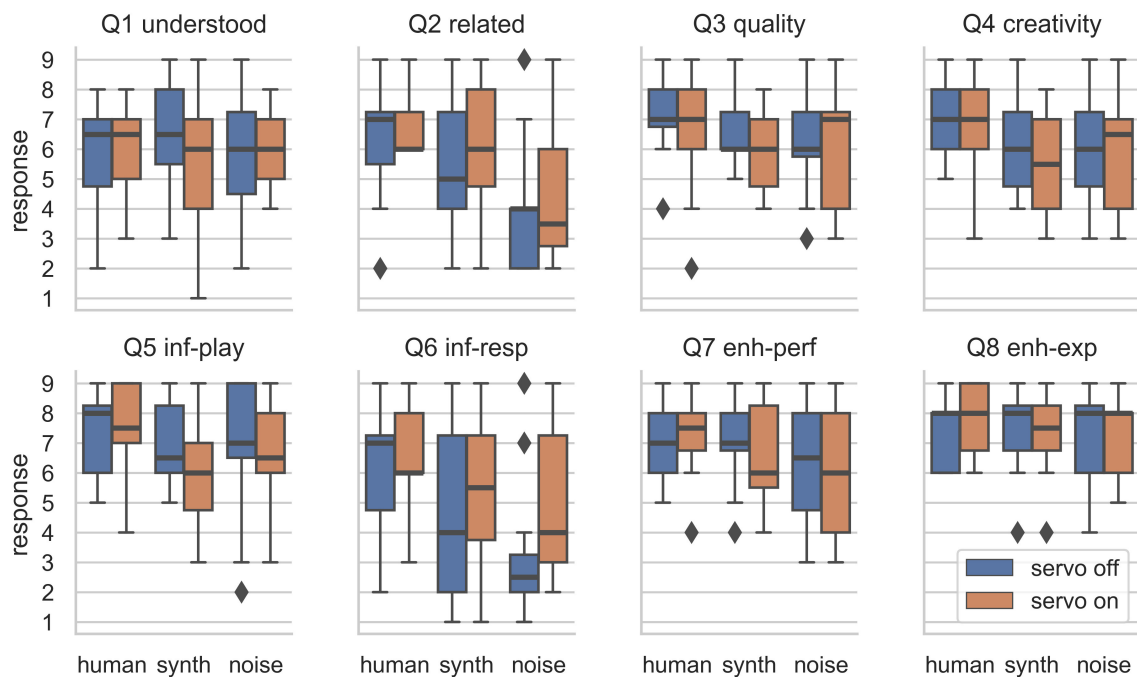


FIGURE 12 | Distribution of responses to the eight survey questions divided by ML model and the presence or absence of the physical lever movement. Outliers are shown as diagonal markers.

TABLE 1 | Survey questions with significant effects due to the ML model.

| Question | F | Significance |
|---|-------|--------------|
| 2. The responses were related to my performance | 12.42 | $p < 0.001$ |
| 4. The responses showed musical creativity | 6.87 | $p < 0.01$ |
| 5. The responses influenced my playing | 6.23 | $p < 0.01$ |
| 6. My playing influenced the responses | 6.51 | $p < 0.01$ |
| 7. The ML model enhanced the performance | 3.66 | $p < 0.05$ |

showed that the human ML model's performances were rated as significantly more related to the performers' actions, significantly more creative, and significantly more able to be influenced than the other models. It also influenced the performers' playing significantly more than the synth (but not noise) model. This suggests that the human model had learned enough human-like behavior to interact with the human performers in a natural way. The synth model was rated as performing significantly less related actions than the human model, but was significantly better than the noise model. While the noise model was rated as providing significantly less enhancement to the performances, it did draw some positive ratings, and in particular, was not significantly more or less influential over the player's performance than the other two models.

4.2.4. Interview Results

The interviews following each session were structured around the performers favorite/least favorite condition, whether they preferred the servo on or off, which model they preferred, how they found the interface, and whether they had suggestions for improvement.

Almost all of the participants identified one of the human or synth conditions as their favorite, with physical actuation either on or off. They often reported that these conditions had felt most responsive to their different inputs. Two participants seemed to favor the noise model due to its interesting rhythmic pattern and the fact that it was consistent. Six of the participants indicated that one of the noise conditions had been their least favorite; their main complaint was that they couldn't figure out what the noise model was doing. The other participants chose a human or synth condition as their least favorite. One mentioned disliking the smooth movement of the human model and others disliked the repetitive gestures of the synth model.

Six of the twelve participants preferred to have physical actuation, three preferred not to have actuation, and three had no preference. Some participants preferred to have the visual reinforcement of the model's responses, one noted that it was fun to have it moving, and another that it was similar to eye contact in an ensemble. The servo-detractors felt that it drew their attention away from the sound. One participant even closed their eyes when the servo was turned on.

In general, the participants were able to identify the three distinct models in performance without having been told explicitly during the session. They commented on the idea of exploring the influence they had over the responses as well as taking influence from it. Several participants attempted to lead the models and commented that the synth model seemed to respond most clearly to different kinds of inputs. Some participants were frustrated that the models were most influenced by their training data, rather than the current performance. One suggested implementing something more like a looper. While several participants noticed that the noise model did not respond to their performances, some enjoyed the distinct sound of its performance. Several noted that the human model was distinguished by its "slidy" sound, and one participant thought this made it more expressive than the other models.

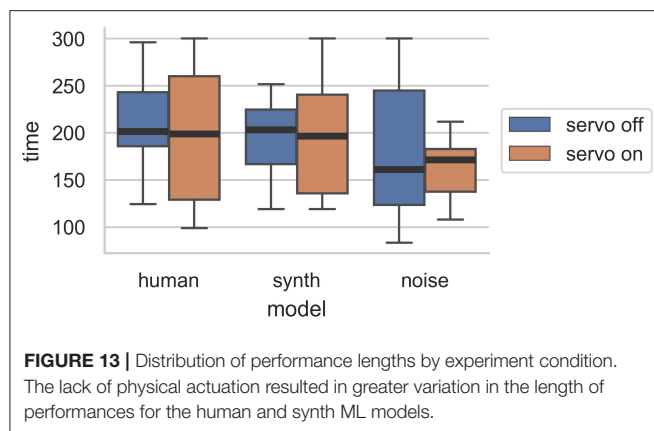
In general, participants seemed to enjoy using the EMPI, and several noted that it was "cute" and fun to interact with. Most of the participants commented that they could only "glide" between notes with the lever, rather than skip pitches. In general, this was seen as a limitation when compared with the ability of the ML model to skip between notes. One participant, however, mentioned that they felt they had improved over the session. The participants also saw the focus on pitch control as a limitation and one envisaged controlling other parameters by moving the input lever in other directions. Others suggested extra sounds or controls to fine-tune their input. Although the EMPI was generally seen as unique, one participant compared the EMPI to a flex-a-tone (a novelty percussion instrument) and another to a hurdy gurdy (a string instrument with a crank-driven friction wheel). Several participants saw the strict call-and-response interaction as a limitation, and wanted responses that could overlap with their performance. One suggested reducing the gap between their input and the response to allow for continuous sound.

4.3. Discussion

The results of our study reveal variations in how performers perceive the EMPI's machine learning models and interface. The ML model used in each performance had a significant effect on responses to five of the eight survey questions covering the relationship between performance and response, the musical creativity of responses, the amount of influence between the participants' performance and the responses, and the extent to which responses enhanced performances. The human model seemed to produce responses that were most related to the participants' performance and were most creative. This model seemed to influence the performers and receive their influence most readily. On the other hand, several participants reported that the synth model was their favorite in interviews. One participant even favored the noise model.

A complication of this comparison is that the synth and noise models sounded distinct from the participants' performances, primarily due to their quite different temporal behavior. In contrast, the human model sounded more similar to what the performers played. As a result, the human model may have been less memorable at the end of the session. In terms of interaction with the ML models, some participants were concerned with exploring responses, discovering ways to exert control over what the model would do. Others reported drawing inspiration from the ML model's performances, particularly those based on the noise and synth models.

Several participants expressed a desire for the responses to be more directly related to their own performances, perhaps more like a looper, or reflexive instrument (Pachet et al., 2013). In contrast, our MDRNN model (similarly to other RNN-based music systems) has only limited capacity to reflect the performer's input material, and the relationship to the training dataset is much more clear. These participants may have been more interested in ML-systems with on-line training capability. Our study seems to have shown that the performers distinguish between the three models, and see advantages of each one, so a compromise may be to give them control over which ML model



is active, emphasizing the strong role of the training data in what they will hear.

The presence or absence of the servo-actuated lever did not have a significant effect on any of the survey questions. The interviews revealed that although half of the participants liked having the servo turned on, the others preferred it off, or had no preference. This split opinion could explain the negative result in the surveys for this effect. It could be that for performers in control of a solo instrument, the physical embodiment of gestures are less important than for an audience watching a performance.

One objective measure of these performances, the length (shown in **Figure 13**), does show some interesting results related to the servo. For both the human and synth performance, the interquartile range of the length is wider with the servo on than off. For noise, the interquartile range is wider without the servo. An interpretation of these results is that for the more widely favored models, the presence of the servo encouraged some performers, who played for longer, and discouraged others, who stopped performances sooner. The random and unyielding nature of the noise model's performance may have been more apparent with the servo turned on, resulting in shorter performances. It seems that there may yet be an effect due to physical representation of the ML model's behavior in terms of how quickly performers recognize and understand boring responses. A further study could isolate this effect while controlling for differing opinions on physical actuation.

The participants were broadly positive about the EMPI's interface design and interacting with the ML models. They agreed in almost all performances that the ML models had enhanced their experiences, and that the responses showed musical quality and creativity. Although some were frustrated by constraints of the single lever interface, they often overcame these to some extent during performance while attempting to match the behaviors of the ML models. Although the performers generally tried to influence the model's responses, they may have been more influenced themselves. This suggests that the choice of model in EMPI may be more important in terms of suggesting different ways to play the instrument than in picking up the performer's pre-existing musical gestures. Future experiments with EMPI could apply other RNN model architectures or datasets to examine the musical styles they might afford performers.

5. CONCLUSIONS

In this work, we have examined musical AI through a novel, machine-learning-enabled musical instrument, the embodied musical prediction interface (EMPI). The EMPI system is consciously constrained. This design choice focuses attention toward embodied predictive interaction, where a performer creates music in a call-and-response improvisation with an ML model that can predict physical musical gestures. We use this interface to investigate how different recurrent neural network models are understood and exploited by performers. We also ask whether the physical representation of predictions helps or hinders the performer. While we have examined the generative potential of our ML models, our focus has been on how this system might be used in genuine musical performance. To this end, we conducted a formal, controlled experiment where 12 participants created 72 improvised pieces of music.

Through this study, we found evidence that the ML model's training dataset affects how performers perceive the model's responses, the extent to which they are able to influence it and use it as a source of inspiration. We found that the different performers appreciated different models and that their interest was often drawn to models that were distinct from their playing. Although the survey results often favored the human model, some performers expressed preferences for the model trained on synthetic data and even the model trained on noise. We found that the performers were split on their preference for the physically actuated lever although analysis of the length of the improvised performances suggests that it affects how long the EMPI performance might hold their interest.

These findings suggest that the presence of different ML models can change how users perform with a musical interface. The use of an MDRNN to predict embodied gestural data, rather than musical notes, seems to have added a new dimension of flexibility to our instrument in terms of creating models from synthetic data. The human model sounded most related to the performer's playing, but the two models based on computer-generated data also led to satisfying improvisations. It is entirely feasible to add more custom-designed models to EMPI and to allow musicians to choose which they would like to use, even during the same performance. Our study results suggest that this could lead to new kinds of performances both from the ML response, and the performers' interactions.

While the use of physical actuation was not universally appreciated, overall, the performers reacted positively to the EMPI instrument. Many participants continued to perform and explore the interface and the ML responses up to the 5-min limit of the experimental improvisations. This finding suggests that constrained and gesture-focussed musical instruments can benefit from generative ML interactions that, so far, have often been limited to keyboard-style interfaces. Constrained and self-contained electronic instruments could be an effective way to deploy musical AI systems into broader use by musicians. Physically actuated indicators may be controversial but have the potential to encourage users to explore new ways of operating an interactive music system.

Our work has demonstrated that although simple, EMPI supports a range of musical interactions afforded by the presence of multiple ML models. We also found that while physical actuation of embodied predictions can serve as both an aid and a distraction to different performers, interacting with embodied predictions can enhance a performer's musical experience. Overall, this work contributes new understandings of how musicians use generative ML models in performance backed up by experimental evidence. Our embodied predictive instrument is also a contribution as an open hardware and software system. This research has demonstrated that EMPI can produce compelling music experiences within a lab setting. We argue that EMPI, and future embodied predictive instruments, hold substantial potential for enhancing and enabling musical creativity.

DATA AVAILABILITY STATEMENT

The survey data and performance durations are available in the **Supplementary Material** and a video showing the six experimental conditions is available online: <https://doi.org/10.5281/zenodo.3521178>. The interface and machine learning code for this project is open source and available online: <https://doi.org/10.5281/zenodo.3451729>.

ETHICS STATEMENT

The studies involving human participants were reviewed and approved by The ANU Human Research Ethics Committee, The Australian National University, Telephone: +61 2 6125 3427, Email: Human.Ethics.Officer@anu.edu.au. The participants provided their written informed consent to participate in this study.

REFERENCES

- Ananthanarayanan, G., Bahl, P., Bodik, P., Chintalapudi, K., Philipose, M., Ravindranath, L., et al. (2017). Real-time video analytics: the killer app for edge computing. *Computer* 50, 58–67. doi: 10.1109/MC.2017.3641638
- Berdahl, E., and Ju, W. (2011). "Satellite CCRMA: a musical interaction and sound synthesis platform," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '11, eds A. R. Jensenius, A. Tveit, R. I. Godøy, and D. Overholt (Oslo: University of Oslo), 173–178.
- Berdahl, E., Salazar, S., and Borins, M. (2013). "Embedded networking and hardware-accelerated graphics with Satellite CCRMA," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '13, eds W. S. Yeo, K. Lee, A. Sigman, H. Ji, and G. Wakefield (Daejeon: KAIST), 325–330.
- Bishop, C. M. (1994). *Mixture Density Networks*. Technical Report NCRG/97/004, Neural Computing Research Group, Aston University.
- Brando, A. (2017). *Mixture density networks (MDN) for distribution and uncertainty estimation* (Master's thesis), Universitat Politècnica de Catalunya, Barcelona, Spain.
- Bretan, M., and Weinberg, G. (2016). A survey of robotic musicianship. *Commun. ACM* 59, 100–109. doi: 10.1145/2818994
- Briot, J.-P., Hadjeres, G., and Pachet, F.-D. (2020). "Deep learning techniques for music generation," in *Computational Synthesis and Creative Systems* (Cham: Springer). doi: 10.1007/978-3-319-70163-9

AUTHOR CONTRIBUTIONS

CM designed the EMPI interface and machine learning system and conducted the experiments in this work. TN and CM collaborated on the hardware design of the EMPI interface. KG encouraged CM to investigate the system from a self-aware cybernetic system perspective. JT supervised the project and contributed to the research design. All authors provided the critical feedback and helped to shape the research and manuscript.

FUNDING

This work was supported by the Research Council of Norway through the EPEC project (#240862), and its Centres of Excellence scheme (#262762).

ACKNOWLEDGMENTS

We wish to thank Akhsarbek Gozoev for contributing to the EMPI enclosure design, as well as Vegard Søyseth and Yngve Hafting for helpful suggestions for hardware improvements. We thank our study participants for their important contribution to this research.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/frai.2020.00006/full#supplementary-material>

Video S1 | Overview video of EMPI: the embodied musical predictive interface.

Data Sheet 1 | Survey results and durations of performances.

- Broughton, M., and Stevens, C. (2008). Music, movement, and marimba: an investigation of the role of movement and gesture in communicating musical expression to an audience. *Psychol. Music* 37, 137–153. doi: 10.1177/0305735608094511
- Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). "Fast and accurate deep network learning by exponential linear units (ELUs)," in *International Conference on Learning Representations* (San Juan).
- Davies, S. (2005). *Themes in the Philosophy of Music*. Oxford: Oxford University Press.
- Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., et al. (2017). TensorFlow distributions. *arXiv [Preprint]*. arXiv:1711.10604.
- Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. (2019). "GANSynth: adversarial neural audio synthesis," in *7th International Conference on Learning Representations, ICLR 2019* (New Orleans, LA).
- Fiebrink, R. (2017). "Machine learning as meta-instrument: human-machine partnerships shaping expressive instrumental creation," in *Musical Instruments in the 21st Century: Identities, Configurations, Practices*, eds T. Bovermann, A. de Campo, H. Egermann, S. I. Hardjowirogo, and S. Weinzierl (Singapore: Springer Singapore), 137–151.
- Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S. M. A., and O. Vinyals. (2018). "Synthesizing programs for images using reinforced adversarial learning," in *Proceedings of the 35th International Conference on Machine Learning, Vol. 80*, eds J. Dy and A. Krause (Stockholm), 1666–1675.
- Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv [Preprint]*. arXiv:1308.0850v5.

- Gurevich, M., Marquez-Borbon, A., and Stapleton, P. (2012). Playing with constraints: stylistic variation with a simple electronic instrument. *Comput. Music J.* 36, 23–41. doi: 10.1162/COMJ_a_00103
- Ha, D., and Eck, D. (2017). A neural representation of sketch drawings. *arXiv [Preprint]*. arXiv:1704.03477v4.
- Hinton, G., Vinyals, O., and Dean, J. (2015). “Distilling the knowledge in a neural network.” In *NIPS 2014 Deep Learning Workshop* (Montreal, QC).
- Huang, C.-Z. A., Cooijmans, T., Roberts, A., Courville, A., and Eck, D. (2017). “Counterpoint by convolution,” in *Proceedings of ISMIR 2017* (Suzhou), 211–218.
- Jensenius, A. R., Wanderley, M. M., Godøy, R. I., and Leman, M. (2010). “Musical gestures: concepts and methods in research,” in *Musical Gestures: Sound, Movement, and Meaning*, eds B. I. Godøy and M. Leman (New York, NY: Routledge), 12–35.
- Kay, M., and Wobbrock, J. O. (2019). *ARTool 0.10.6: Aligned Rank Transform for Nonparametric Factorial ANOVAs*. Geneva: Zenodo. doi: 10.5281/zenodo.594511
- Leman, M., Maes, P.-J., Nijs, L., and Van Dyck, E. (2018). “What is embodied music cognition?” in *Springer Handbook of Systematic Musicology*, ed R. Bader (Berlin; Heidelberg: Springer Berlin Heidelberg), 747–760.
- Lewis, P. R., Chandra, A., and Glette, K. (2016). “Self-awareness and self-expression: Inspiration from psychology,” in *Self-aware Computing Systems: An Engineering Approach*, eds R. P. Lewis, M. Platzner, B. Rinner, J. Tørrsen, and X. Yao (Cham: Springer International Publishing), 9–21.
- Martin, C. (2019a). *EMPI v0.3*. Geneva: Zenodo. doi: 10.5281/zenodo.3451730
- Martin, C. (2019b). *Keras MDN Layer v0.2.1*. Geneva: Zenodo. doi: 10.5281/zenodo.3376850
- Martin, C. P., and Gardner, H. (2019). “Free-improvised rehearsal-as-research for musical HCI,” in *New Directions in Music and Human-Computer Interaction*, Springer Series on Cultural Computing, eds S. Holland, T. Mudd, K. Wilkie-McKenna, A. McPherson, and M. M. Wanderley (Cham: Springer), 269–284.
- Martin, C. P., and Tørrsen, J. (2018). “RoboJam: a musical mixture density network for collaborative touchscreen interaction,” in *Computational Intelligence in Music, Sound, Art and Design*, eds A. Liapis, J. J. Romero Cardalda, and A. Ekárt (Cham: Springer International Publishing), 161–176.
- Martin, C. P., and Tørrsen, J. (2019). “An interactive musical prediction system with mixture density recurrent neural networks,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME ’19, eds M. Queiroz, and A. X. Sedó (Porto Alegre: UFRGS), 260–265.
- McNamee, D. (2009). *Hey, What’s That Sound: Stylophone*. The Guardian. Available online at: <https://www.theguardian.com/music/2011/jun/16/korg-monotribe-monotron>
- McPherson, A., Jack, R., and Moro, G. (2016). “Action-sound latency: are our tools fast enough?” in *Proceedings of the International Conference on New Interfaces for Musical Expression, Volume 16 of 2220–4806* (Brisbane, QLD: Queensland Conservatorium Griffith University), 20–25.
- Moro, G., Bin, A., Jack, R. H., Heinrichs, C., and McPherson, A. P. (2016). “Making high-performance embedded instruments with Bela and Pure Data,” in *International Conference on Live Interfaces* (Brighton: University of Sussex).
- Næss, T. R., and Martin, C. P. (2019). “A physical intelligent instrument using recurrent neural networks,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, eds M. Queiroz, and A. X. Sedó (Porto Alegre: UFRGS), 79–82.
- Nymoen, K., Chandra, A., and Tørrsen, J. (2016). “Self-awareness in active music systems,” in *Self-aware Computing Systems: An Engineering Approach*, eds R. P. Lewis, M. Platzner, B. Rinner, J. Tørrsen, and X. Yao (Cham: Springer International Publishing), 279–296.
- Pachet, F. (2003). The continuator: musical interaction with style. *J. New Music Res.* 32, 333–341. doi: 10.1076/jnmr.32.3.333.16861
- Pachet, F., Roy, P., Moreira, J., and d’Inverno, M. (2013). “Reflexive loopers for solo musical improvisation,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’13 (New York, NY: ACM), 2205–2208.
- Pressing, J. (1990). Cybernetic issues in interactive performance systems. *Comput. Music J.* 14, 12–25.
- Reus, J. (2011). “Crackle: a mobile multitouch topology for exploratory sound interaction,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME ’11, eds A. R. Jensenius, A. Tveit, R. I. Godøy, and D. Overholt (Oslo: University of Oslo), 377–380.
- Roberts, A., Engel, J., Mann, Y., Gillick, J., Kayakic, C., Nørly, S., et al. (2019). “Magenta studio: augmenting creativity with deep learning in Ableton Live,” in *Proceedings of the International Workshop on Musical Metacreation (MUME)* (Charlotte, NC).
- Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. (2018). “A hierarchical latent vector model for learning long-term structure in music,” in *Proceedings of the 35th International Conference on Machine Learning, Volume 80 of Proceedings of Machine Learning Research*, eds J. Dy and A. Krause (Stockholm: Stockholmsmässan; PMLR), 4364–4373.
- Rowe, R. (1993). *Interactive Music Systems: Machine Listening and Composing*. Cambridge, MA: The MIT Press.
- Sturm, B. L., and Ben-Tal, O. (2017). Taking the models back to music practice: evaluating generative transcription models built using deep learning. *J. Creative Music Syst.* 2, 1–29. doi: 10.5920/JCMS.2017.09
- Waisvisz, M. (2004). *The CrackleBox* (’75). Retrieved from: <http://www.crackle.org/CrackleBox.htm>
- Williams, E. J. (1949). Experimental designs balanced for the estimation of residual effects of treatments. *Aust. J. Chem.* 2, 149–168.
- Wobbrock, J. O., and Kay, M. (2016). “Nonparametric statistics in human-computer interaction,” in *Modern Statistical Methods for HCI*, Chapter 7, eds J. Robertson and M. Kaptein (Cham: Springer International Publishing), 135–170. doi: 10.1007/978-3-319-26633-6_7
- Zappi, V., and McPherson, A. (2014). “Design and use of a hackable digital instrument,” in *Proceedings of the International Conference on Live Interfaces* (Lisbon).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Martin, Glette, Nygaard and Tørrsen. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Computational Creativity and Music Generation Systems: An Introduction to the State of the Art

Filippo Carnovalini* and Antonio Rodà

Department of Information Engineering, CSC - Centro di Sonologia Computazionale, University of Padova, Padua, Italy

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Fabio Aurelio D'Asaro,
University of Naples Federico II, Italy
Rinkaj Goyal,
Guru Gobind Singh Indraprastha
University, India

*Correspondence:

Filippo Carnovalini
filippo.carnovalini@dei.unipd.it

Specialty section:

This article was submitted to
Machine Learning and Artificial
Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 30 October 2019

Accepted: 09 March 2020

Published: 03 April 2020

Citation:

Carnovalini F and Rodà A (2020)
Computational Creativity and Music
Generation Systems: An Introduction
to the State of the Art.
Front. Artif. Intell. 3:14.
doi: 10.3389/frai.2020.00014

Computational Creativity is a multidisciplinary field that tries to obtain creative behaviors from computers. One of its most prolific subfields is that of Music Generation (also called Algorithmic Composition or Musical Metacreation), that uses computational means to compose music. Due to the multidisciplinary nature of this research field, it is sometimes hard to define precise goals and to keep track of what problems can be considered solved by state-of-the-art systems and what instead needs further developments. With this survey, we try to give a complete introduction to those who wish to explore Computational Creativity and Music Generation. To do so, we first give a picture of the research on the definition and the evaluation of creativity, both human and computational, needed to understand how computational means can be used to obtain creative behaviors and its importance within Artificial Intelligence studies. We then review the state of the art of Music Generation Systems, by citing examples for all the main approaches to music generation, and by listing the open challenges that were identified by previous reviews on the subject. For each of these challenges, we cite works that have proposed solutions, describing what still needs to be done and some possible directions for further research.

Keywords: computational creativity, music generation, survey, meta-review, algorithmic composition, musical metacreation, automatic composition, computer music

1. INTRODUCTION

What is Creativity?

While the term is of fairly common use in everyday life, giving a precise definition of this concept is not a trivial task. The general idea is that it relates to the ability that some human individuals possess to create something that did not exist before. Upon further reflection, one can notice that most of the times these “creations” start from concepts that already existed, or at least that could already have existed, but that nobody had already explicitly linked in a fixed product. This kind of “novel linkage” is what brought us works of art such as Dals The Persistence of Memory: clocks had been painted before, and everybody has experienced that things can melt, but nobody had yet linked these two concepts in a painting.

There is another question relating to creativity that raises even more problematic considerations: can computers be creative? The usual experience with machines is that we humans give a set of instructions to the machine along with some initial data (the input), and we expect the machine to behave in a way that is fully deterministic, always giving the same output when the same input is given. Moreover, we expect that the output should be something that can be fully expected and

computed even without the help of a computer, albeit the computation of the output could be extremely time-consuming (otherwise we would not have resorted to computers in the first place).

The word deterministic seems to be the exact opposite of our understanding of the concept of creativity, and yet the idea of obtaining creative behaviors from computers has inspired the writing of a notable amount of scientific publications, that can be collected under the field of *Computational Creativity* (CC), defined as:

“The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviors that unbiased observers would deem to be creative.”
Colton and Wiggins (2012).

Practitioners of this field share the interest in gaining a better understanding of how creativity works, and to what extent it can be replicated via a computer system. The above definition underlines the diversity in background of the people researching CC. Such a diverse community is for sure source of many interesting insights, but there is also room for many different goals and perspectives that sometimes can make it hard to understand what is the current general direction of the research, and what directions should be explored for the advancement of the field (Lamb et al., 2018).

One of this field's goals is for sure answering to the above question can computers be creative?, that is most interesting to computer scientists and engineers that wish to create advanced models of artificial intelligence with creative capabilities. Yet, this is only one of the possible goals: artists could be more interested in finding out how computers can help express their own creativity, while psychologists and philosophers are more interested in using computer models of creativity to better understand creative processes that happen in humans (Pearce et al., 2002). The diversity of the goals is reflected in the diversity of the literature concerning CC. Some works are devoted to the definition or the assessment of creativity itself. A notable amount of contributions focus on the design of systems that are meant to be creative, sometimes designing them starting from some definition of creativity, and thus trying to actually obtain output that is creative according to a certain evaluation method. Other times, the system is simply designed to tackle tasks that are commonly considered to involve creativity (usually artistic tasks like painting or composing music), but the process involved in the generation of such output does not necessarily involve creativity. More often than not, the creativity of the creative systems is not evaluated in any formal way, either having just summary evaluations of the quality of the output or not having any evaluation at all (Jordanous, 2012). This might be fine for the artistic goal of empowering the creativity of humans through computational means, but is not acceptable for AI practitioners trying to understand whether computers can exhibit creative behaviors.

One especially prolific research task of CC is that of music generation, that has interested computer scientists even before the birth of the term Computational Creativity. Ever since

the early days of computing, scientists and engineers have used computers for musical task, creating digital synthesizers, developing engraving software, and also writing procedures that generate musical scores, to be performed either by computers or by humans. This task was called Algorithmic Composition, and as the name suggests was related to a well defined procedure (an algorithm, once again a concept distant from creativity Papadopoulos and Wiggins, 1999; Nierhaus, 2009; Fernández and Vico, 2013). A name that is more common today is Musical Metacreation, that suggests the fact that the programmer creates a system that in turn can create some kind of music Pasquier et al., 2017; Bodily and Ventura, 2018). Throughout this article, we will use the more neutral term “Music Generation Systems” (MGSs). Music is especially interesting for the investigation in CC because of the broad possibilities that it offers in terms of mathematical and computational representations, and because it does not need explicit semantics like other forms of art such as poetry or non-abstract painting (Wiggins, 2018). This might be some of the reasons why there is so much research on MGSs, and also many reviews of the literature. Those reviews usually focus on the technical approaches used for music generation rather than the contributions given to the understanding of creativity in computers and humans. What this survey tries to accomplish is to give a broad introduction to the field of CC with a focus on MGSs, first reviewing the literature on the definition and evaluation of creativity and then focusing on the systems proposed in literature, describing the current approaches and the challenges that are still not fully addressed, and what kind of solutions have been tried or proposed to overcome those problems.

2. FORMALIZING CREATIVITY

2.1. Defining Creativity

We opened the introduction to this paper asking what is creativity. This is a question that many researchers have faced before, especially after Guilford's speech to the American Psychological Association in 1950 advocating for psychological studies on creativity (Rhodes, 1961). Since that year, psychological research on creativity exploded, exploring many facets of what defines and stimulates creativity in humans. Some works were focused on the study of what are the personality traits of the creative person (Rhodes, 1961; Getzels and Jackson, 1962), as well as what external factors can positively or negatively influence creativity (Amabile, 1983a,b). Other researchers were more interested on the mental processes that happen in the creation of something creative, and finally many works were dedicated to the definition of creativity itself.

2.1.1. Creativity as Novelty and Value

The newfound abundance of research led to having hundreds of definitions of creativity in literature. In their works, Sarkar and Chakrabarti analyzed over 200 of those (Sarkar and Chakrabarti, 2008, 2011; Ranjan et al., 2018), finding that the factors that have been used as indicators for creativity can be grouped in two main categories: *Novelty* (or unusualness, unexpectedness, surprise, originality) and *Value* (or usefulness, quality, appropriateness,

meaningfulness). This subdivision is not new at all, as already Stein (1953) had proposed a definition of a creative work as novel and useful or satisfying. Novelty is usually considered the defining characteristic of a creative artifact, but value is also necessary: it is easy to think of something that has never been built before, like a car with fifteen wheels, but while such car would be novel, it would have higher maintenance costs, with little or no increase in performance. This kind of novelty lacks value: creativity (that includes value) introduces innovations useful to the purpose of the created object, possibly leading to a general advancement in its own field. One example of creativity in the field of car manufacturing could be the introduction of hybrid cars: the idea of using two different energy sources was novel, but hybrid cars are now common because of the advantages they bring to their owners in terms of efficiency. On the contrary, it is highly unlikely that our fifteen-wheeled car could become an industrial standard.

While Novelty and Values are surely important features of creativity, these give only a vague description of creativity. From their study of the literature, Sarkar and Chakrabarti (2008) reached a somewhat more complete definition of creativity:

“Creativity occurs through a process by which an agent uses its ability to generate ideas, solutions or products that are novel and valuable.”

2.1.2. The Four Perspectives of Creativity

The above definition points out that creativity is a concept that cannot be ascribed to the final artifact, but must consider its creation. In particular, it underlines the existence of a process, used by an agent, to create a product. These represent three of the four “P’s” of creativity, first identified by Rhodes (1961): Person, Process, Product, Press. Rhodes was interested in the educational aspects of creativity and in educating children to be creative, but later the focus shifted toward the study of what makes something be considered creative. This shift is probably partly due to a paper by Anna Jordanous, who revisited the concept of the four P’s under the light of the evaluation of creativity (Jordanous, 2016), but the definition of the four P’s had already started changing soon after the original paper by Rhodes (Golann, 1963), showing that while the general idea of the four P’s was immediately utilized by the scientific community, it took some time to reach widely accepted definitions. Lamb et al. (2018) describe the four terms as follows:

- **Person:** is the human (or non-human agent) who is seen as creative. Person theories study what it is about the agent that makes them creative.
- **Process:** is the set of internal and external actions the agent takes when producing a creative artifact. Process theories study what sort of actions are undertaken when creative work is done.
- **Product:** is an artifact, such as an artwork or a mathematical theorem, which is seen as creative or as having been produced by creativity. Product theories study what it is about the product that makes it worthy of being called creative.

- **Press:** is the surrounding culture which influences people, processes, and products and which judges them as creative or uncreative. Press theories study what it is that leads a culture to view something as creative.

This useful subdivision into four perspectives helps frame the various contributions on creativity, as often each work focuses on only one or two of the above perspectives. For example, the definitions of creativity as Novelty and Value are focused on the Product, even if Sarkar and Chakrabarti’s definition encompasses almost all four P’s. In the following sections we review some contributions that focus on the other three perspectives: Person, Process and Press.

2.1.3. Person

Regarding the Person perspective, the study of the personality traits of creative people has unsurprisingly interested many psychologists: already in the first years after Guilford’s speech many works emerged (those early works were reviewed by Golann, 1963), and soon was found out that creativity is not directly related to intelligence (Getzels and Jackson, 1962), and a relationship between creativity and humor was also noted (Treadwell, 1970). Guilford himself underlined that creatives emerge for their sensitivity to problems, mental flexibility, and divergent thinking (Guilford, 1957, 1967). The importance of this last trait was exploited by Torrance, who designed the Tests of Creative Thinking (Torrance, 1965) that give an effective measure for the individuation of creative people (Torrance, 1988). Simonton (2000) gives a review of psychological studies on creativity in terms of personal and developmental traits, as well as the socio-cultural influence of creativity (connecting the Person and the Press perspectives).

Within the field of CC, one could argue that any Turing-complete machine is equivalent in what it can achieve, thus making every computer system equal under the Person perspective. Nonetheless, the Person remains an insightful perspective at a more abstract level, for example when a software system can be viewed as an agent or as a group of agents collaborating together. In this case, the (virtual) personality of each agent could give a different contribution to the system, making it useful to consider psychological personality aspects such as motivation (Guckelsberger et al., 2017) or curiosity (Schmidhuber, 2012), or to try and model in software cognitive aspects of creativity (Wiggins and Forth, 2015; Wiggins and Sanjekdar, 2019).

2.1.4. Process

The Process perspective has interested CC the most, as someone who wishes to obtain a creative behavior from a computer must know how to describe creativity in algorithmic terms. While there is no such thing as a fixed procedure to obtain something creative, it is possible to gain insights on how to obtain creativity from the study of the creative processes of people that have shown great creativity throughout history (and wrote how they reached that idea). This is in part what Margaret Boden did in her book, *The Creative Mind* (Boden, 2004) (for a shorter introduction to the same ideas, see Boden, 1998, 2009). The description of creativity

she provides in that book has become extremely influential to the field of CC, also because she used computer models of creativity to discuss her ideas, explaining what was obtained and what was still to be achieved by machines. One major contribution she gave was the introduction of the idea of “Conceptual Space,” i.e., a space where the possible concepts exist, some of which have been explored and some are yet to be discovered. This idea allows the distinction of many levels of creativity:

- **Combinational Creativity:** two already explored ideas from a concept space are joined, thus creating an association that is novel;
- **Exploratory Creativity:** some kind of method for the free exploration of the concept space is used, to find regions in the space that have not been yet explored, but are valuable;
- **Transformational Creativity:** the highest level of creativity is reached when a new idea is found that was not part of the original conceptual space, thus changing the shape of the concept space itself.

The idea of obtaining creative ideas from the union of two known ideas, that Boden called Combinational Creativity, is at the basis of other theories of creativity, although with different names: Koestler (1964) called the same idea *Bisociation*, while Fauconnier and Turner (2008) used the term *Conceptual Blending*. The novelty of Boden’s theory lies in the introduction of conceptual spaces, necessary for the definition of the other two levels of creativity. Wiggins (2006, 2019) mathematically formalized these ideas, also showing that Transformational Creativity is equal to Exploratory Creativity on a meta-level.

Another useful notion introduced by Boden is the distinction between *H-Creativity* (historical creativity) and *P-Creativity* (personal creativity). In order for something to be H-Creative, it must be the first time it has appeared in the history of mankind, while to be P-Creative it is enough to be new to the one creating it. As an example, Boden mentions that if a child can prove Pythagoras’ theorem without any help, we would find this deed an impressive example of mathematical creativity even if that theorem was demonstrated millennia ago. H-Creativity is what is usually considered novel and/or creative, but Boden argues that P-Creativity is just as important as it originates from the same creative Process.

2.1.5. Press

The Press perspective is most interesting to the evaluation and assessment of creativity. This is not just an appendix to the concept of creativity: the definition we gave for CC seeks behaviors that are deemed to be creative by an unbiased observer, making it necessary to have an external appraisal of the Product before calling something creative. The works of Amabile have underlined both the importance of the environment for the development of creativity (Amabile, 1983b; Amabile et al., 1996) and the importance of the assessment of creativity, proposing one of the first formalized methods for the evaluation of creativity, using expert judges (Amabile, 1983a). Moreover, Csikszentmihalyi (2013) pointed out the proactive function that field’s experts can have in increasing the rate of

creativity in a particular domain. We will discuss the problems relating to the evaluation of creativity later (see section 2.3).

Even if someone tried to directly assess the creativity of a Product, of the Process behind it, or of the Person, he needs to pass through the lens of human perception (and thus the Press perspective) to be really understood (Colton, 2008), making the Press perspective the most ubiquitous. On the other hand, the Press perspective is not enough to give an indication of creativity, since commercial success or reach of a Product is influenced by a variety of factors that go beyond creativity, or even just its Value (Fraiberger et al., 2018).

2.1.6. Dimensions of Creativity

Another interesting contribution to the definition of Creativity comes from Jordanous and Keller (Jordanous, 2012, 2013, 2019; Jordanous and Keller, 2012, 2016), who used a statistical language processing techniques to identify fourteen main components of creativity, as described by scientific research on the topic. This study resulted in an unordered list of components, that should be seen as different dimensions of the concept of creativity rather than a systematic description (Jordanous and Keller, 2016):

- Active Involvement and Persistence;
- Dealing with Uncertainty;
- Domain Competence;
- General Intellectual Ability;
- Generation of Results;
- Independence and Freedom;
- Intention and Emotional Involvement;
- Originality;
- Progression and Development;
- Social Interaction and Communication;
- Spontaneity/Subconscious Processing;
- Thinking and Evaluation;
- Value;
- Variety, Divergence and Experimentation.

The notion of Novelty (here called Originality) and Value are kept, but using all 14 components gives a much broader definition of creativity, that considers all the four Ps: for example *General Intellectual Ability* is related to the Person, *Progression and Development* to the Process, *Value* to the Product, and *Social Interaction and Communication* is connected to the Press perspective. Jordanous and Keller (2012) explain that not all the components listed above will be as important in all possible creative deeds, so this list also offers the possibility to categorize different kinds of creativity required by different activities.

To our knowledge, there is no work in literature that has given a short definition or a model of creativity based on these fourteen dimensions.

2.2. Computers and Creativity

The above definitions of creativity were general enough to be applied both to humans and machines alike (although we sometimes focused on the implication of those theories on computers). It is now time to face the second question we posed in the introduction: can computers be creative?

This is a question that seems to be as old as computer science: Lady Lovelace, while commenting the Analytical Engine, mentioned that computers do not have the ability to originate anything on their own (Lovelace, 1843). As paraphrased by Bringsjord et al. (2003), her statement reads:

“Computers can’t create anything. For creation requires, minimally, *originating* something. But computers originate nothing; they merely do that which we order them, via programs, to do.”

The Countess leaves no room whatsoever for creativity, but other important scientists disagreed with her. Alan Turing, who argued that artificial intelligence should have creative abilities, responded to Lady Lovelace’s objection pointing out that she had no real experience in programming, while we now know that a computer can often surprise us by doing the exact opposite of what we intended, until a program is thoroughly checked for bugs (Turing, 1950). This response is somewhat unsatisfying, since it seems that the only accountability for creativity from computers would come from human errors, but in the rest of the article Turing argues that intelligent machines should be able to learn, thus gaining abilities beyond those envisioned by the original programmer.

Another strong argument against computer creativity is that of the “Chinese Room” introduced by Searle (1980). He argues against artificial intelligence in general, but the argument applies to creativity as well. He imagines to be locked inside a closed room, that can accept questions and give answers written on paper, either in English or in Chinese. For the English questions, he would answer normally using his own intelligence, while for the Chinese ones he would use a special script telling him, for any combination of Chinese symbols that he sees, what symbols to write as answer. Supposedly, the English answers would be as good as the Chinese ones to the eyes of the people outside the room (if the Chinese script is good enough), but the person inside would not gain any knowledge of Chinese in this way. Searle argues that computers work in this way, manipulating symbols without having a real understanding of those.

Searle’s objection is rather convincing, unless we suppose that the manipulations of symbols that happen in computers are in reality not different from those that happen in our brains, if not because of less “computational power” (Minsky, 1982). This vision basically reduces human brains to extremely powerful computers, so that an artificial computer could recreate all of their functions. This is of course far from being a proven truth, and does not fully account for things we experience everyday, such as consciousness, free will, and subjectivity (Chalmers, 1995; Hameroff and Penrose, 2014; Ceroni and Prosperi, 2018).

There is room for a long lasting debate on the possibility of computers being “*really*” creative, but fortunately CC is not ultimately interested in this debate. According to the definition of CC, we want computer systems that have behaviors that an unbiased observer would deem to be creative, and not necessarily behaviors that are *actually* creative. This means that we aim at simulating creativity well enough to trick observers into thinking that the product they are seeing is actually creative.

It is nonetheless important to understand what creativity is, and possibly to incorporate the definitions of creativity in the generation process, because the unbiased observer will judge creativity in the same way as it would with a human, thus implicitly applying some of the concepts relating to creativity that we illustrated above. The problem of the evaluation of creativity thus becomes central: if the goal is to recreate what an observer would deem creative, we need to give metrics of how creative something would be perceived by an observer.

2.3. Evaluating Creativity

Despite the importance of the evaluation of creativity, most of the scientific publications on evaluation only came about in the last 20 years (Jordanous, 2013). In this section we will describe some of the most common creativity evaluation methods. To read some more extensive reviews on this subject, we suggest: Jordanous (2012, 2013, 2014), Lamb et al. (2018), Pease and Corneli (2018), and Ritchie (2019).

2.3.1. Turing Test-Like Approaches

The definition of CC that we gave suggests that creativity needs to be assessed via human judgement, leading to evaluation techniques based on the concept of “Turing Test” (Turing, 1950): ideally, if a human cannot distinguish computer creativity from human creativity, the computer has achieved a satisfying level of creativity.

Amabile (1983a) proposed the Consensual Assessment Technique, which has become the standard evaluation of human creativity (Baer and McKool, 2009). This technique requires a pool of experts independently evaluating a set of artifacts. An artifact can be considered creative if it receives good evaluations and the interrater reliability is high enough (for example having a Cronbach’s alpha higher than 0.7). While this method was not originally conceived for CC, it is easy to insert one or more computer generated artifacts along some human made ones, to get a comparison between human and computer creativity. The judges only have access to the artifact, not knowing anything about its author or background (including whether the author is a computer). This means that they only evaluate the Product perspective in a non interactive way. This is rather different from the original Turing Test, but it was included in this section because it operates a comparison between human and computers carried out by a human evaluator.

Pearce and Wiggins (2001) propose a machine composition framework that includes in its final phase an evaluation inspired by the Turing Test (although the authors underline the major difference of not having interaction). While it was initially defined for music generation, it can be applied to CC in general. This framework supposes that a corpus is available to the software, and that some sort of learning is applied to create a “critic” for that corpus. Once new compositions are generated that satisfy the learnt critic, some generated pieces are presented a group of subjects along with composition coming from the corpus. The evaluators are asked to tell whether the compositions they hear are human or machine made (similarly to Turing’s imitation game). If their evaluation cannot be statistically distinguished from a random selection, the system

is considered effective. This approach, being entirely based on learning a corpus, is arguably not really an evaluation of creativity but rather one of quality in imitating human products.

Ariza (2009) underlines this and other limitations of Turing Test approaches to the evaluation of creativity, showing how sometimes these tests are implemented in a way that he calls “toy Tests,” failing to understand that interactivity between human and computers was the main feature of the “Imitation Game,” as it was meant to assess intelligence, that is experienced through interaction. Another critic to this kind of tests comes from Soldier (2002), who raises a more fundamental doubt on the capability of non-experts to act as evaluators. This is not surprising (indeed, the Consensual Assessment Technique requires experts), but often Turing Test approaches only require the evaluator to be human.

Bringsjord et al. (2003) propose to go beyond the Turing Test with the “Lovelace Test” (inspired by her statement reported in section 2.2). The authors argue that Turing’s game could be beat with simple manipulation of symbols without the need of any intelligence (as Searle described with his Chinese Room example). On the contrary, an agent passes the Lovelace Test if and only if it is capable of creating an output of some kind through a repeatable process, and this output cannot be fully explained by the knowledge-base, the architecture, and the core functionalities of the agent. Unluckily, this test is not easy to perform in real-life situations, and arguably a machine could never pass this test, as every output of a machine is the result of its architecture and functionalities. This might be a good abstract test for *real* creativity, but is not very useful to evaluate CC systems.

A more manageable version of the Lovelace Test was proposed by Riedl (2014), that requires the machine to be able to generate an output that satisfies a set of requirements chosen by a human. The generated output is then evaluated in terms of how well it meets the requirements and if it is “not unrealistic for an average human.” This proposal is somewhat unsatisfying, because by losing the strong requirements of the original Lovelace Test it basically falls back to a standard Turing Test, in a way that Ariza (2009) described as “Directive toy Test,” meaning a Turing Test where the interaction is only limited to giving initial directives for the generation.

2.3.2. Self-Assessment Frameworks

Another popular approach is to have the author of the system describe the way it works and how it can be considered creative or not, and to what degree. These assessments try to frame the chosen Process in some kind of creativity scale, for example distinguishing if the used process is combinational, explorational or transformative, using Boden’s categories. Indeed, this kind of evaluation is reminiscent of how Boden investigated creativity in her book (Boden, 2004).

Colton (2008) introduced these assessments with a reflection on how the evaluation of the Product alone is not enough to evaluate the creativity of a system. He proposes an example, where the same object is obtained through different processes. This can lead to different perceptions of creativity, but obviously only if the process is known to the observer. In that paper, he

introduced the concept of the “Creative Tripod,” a tripod having *Skill*, *Appreciation* and *Imagination* as legs, saying that all three must be extended to some degree in order for the tripod to stand.

The tripod framework had little success, possibly because it was not formalized enough, but it remained influential on literature on creativity evaluation. Colton et al. (2011) and Pease and Colton (2011a,b) described another framework for self-assessment: the FACE and IDEA models. The FACE model can be used to describe the creative capabilities of a system through a set of symbols that tell if the evaluated system *possesses* or is *capable of generating* Expressions (i.e., products), Concepts, Aesthetic measurements, and Framing information (read backwards, the initials spell FACE). The IDEA model describes instead the impact of the system during its lifecycle, starting from the developmental stage and ideally reaching a stage where it can perform some kind of transformational creative processes.

These assessment frameworks are limited in the possibilities they offer, and a common criticism is that the assessment comes from the author of the system, making it biased. Nonetheless it is useful to frame the capabilities of a system and to reflect on the degree of automation in creativity it has reached, even just for development purposes. Indeed, an extension to the FACE/IDEA framework was proposed to consider the creative abilities of different versions of a same software (Colton et al., 2014) to make it easier for a developer to understand how the creativity of the system is progressing.

2.3.3. Quantitative Metrics

In order to compare the results of different systems in terms of creativity, and to give more scientific indications of the effectiveness of CC applications, it is desirable to have objective metrics that can indicate how creative a system is. Designing such metrics is not an easy task, but many efforts have been made toward this goal.

Ritchie (2001) proposed a set of criteria for the evaluation of creativity based on the Product perspective, judged according to *Value* and *Typicality*. The latter is a concept strongly related to Novelty, but is based on the fact that an “inspiration set” (the corpus used by the system) is available, and used to define what is more or less typical. These two basic features must be measured according to some *rating scheme*, and can then be used to compute a set of parametrized criteria, that are basically functions over the Value and Typicality. In his proposal Ritchie described these criteria as either satisfied or not satisfied (if a certain threshold is reached), but often these were applied as a continuous scale rather than a boolean one. Extending this evaluation framework (Pease et al., 2001) focused on the measurement of Value and Typicality, while Colton et al. (2001) investigated the effects of fine-tuning the input knowledge. Ritchie (2007) presented an updated version of his criteria, commenting the works that have used it as a means of evaluation, but the presence of many parameters to be tuned makes it difficult to use for comparisons between different systems.

While Ritchie’s criteria are the main metrics for the evaluation of creativity, there are other metrics in literature that can be relevant for CC systems, although they do not evaluate directly the creativity of the systems. Galanter (2012) made a review on

metrics and methods to evaluate aesthetic value of computer generated artifacts, that is a vital part of many CC systems. Within the field of computer vision, the use of fuzzy logic applied to visual features was suggested for the automatic evaluation of complexity, as well as interestingness and aesthetic value (Cardaci et al., 2009; Tabacchi and Termini, 2011; Constantin et al., 2019). Shaker et al. (2016) focus on procedural content generation, and describe how it is possible to give a visual indication of the capabilities of a system in terms of the variety of products it can generate. To the best of our knowledge, this system has never been used for CC systems, despite the fact that the representation of the space of possible outputs generated by a system has strong links to Boden's theories (which in part inspired Shaker's work). Possibly, these graphical representations could give a good indication of whether a system uses mere combinational creativity or is capable of going beyond that limit.

2.3.4. Evaluation of Generated Music

The evaluation methods that we presented in the previous paragraphs are general enough to be applied to musical generation as well as to other CC applications. The following methods focus instead solely on the evaluation of MGSs.

Eigenfeldt et al. (2012) used a concert setting to evaluate a variety of MGSs, and a similar event is described by Sturm et al. (2019). In both cases, the evaluation in itself was performed via a questionnaire given to the audience of the concert. This approach can be extended by turning the concerts into music competitions, as has been done for computer-generated expressive performances of human composed music (Katayose et al., 2012; Schubert et al., 2017). If the program includes both human and computer generated music, this approach becomes similar to the ones inspired by the Turing Test, but a concert setting is one of the most natural ways to experience music, and could fatigue the evaluators less than a laboratory setting. Two major limitations of this approach is that the audience will evaluate music according to their personal taste, rather than assessing creativity, and that this evaluation method can only be used to compare the pieces that are included in the concert: comparing different concerts could induce unwanted bias due to different performers, venue and setting in general.

Another useful contribution is that of Yang and Lerch (2018), that argue that while creativity cannot be assessed without a human evaluator, it is useful to use *formative* metrics to describe how well computer generated music fits a musical genre, in order to help the development of the system toward "human-like" music generation. To that goal, many quantitative metrics are presented, and data visualization techniques are suggested. While this does not solve the ultimate goal of the evaluation of creativity, it is nonetheless an useful addendum to the evaluation toolbox.

An overview of the current methods for the evaluation of MGSs is present in Agres et al. (2016), that provides both motivations and tools to evaluate in different manners systems that are merely generative, systems that allow for feedback, and systems that are capable of some kind of self-reflection. Moreover, a distinction is presented between internal and external evaluation, the first being necessary for the functioning

of the system and the latter being the usual *a posteriori* evaluation to understand the effectiveness of the system.

3. MUSIC GENERATION SYSTEMS

3.1. Meta-Review

This is not the first review on Music Generation Systems (MGSs), and the goal of this work is not to give a comprehensive review of every contribution to the field, but rather an introduction through examples from literature. To this goal, we searched on Scopus and Google Scholar for reviews on MGSs that have been published over the last 10 years (2009–2019), by searching "computational creativity music," "musical metacreation," "algorithmic composition," and "music generation" followed by "review" or "survey," limiting to first 50 results. From the results, only the papers written in English after 2009 were kept. Of those, the abstract was read to select those that were actually reviews of artificial intelligence techniques for music generation. The selected ones are listed in **Table 1**. Two results (Williams et al., 2013; Briot et al., 2017) were excluded because they were prior versions of the reviews we included by the same authors.

It is important to notice that not all the works on MGSs have the goal of CC in mind. Sometimes the goal of a MGS is to create a formalization of a certain musical style, or to test certain composition rules or assumptions by generating music that satisfies those rules. Other times, the generation of music itself is the only goal of those systems. This is also reflected by the reviews on MGSs, that are not always concerned with the creativity of the reviewed systems: we used search tools to count the number of occurrences of the stem "creativ" in the main body of the reviews, that we reported in the **Table 1** under the column "Mentions to Creativity" to show that some reviews on music generation hardly acknowledge the problem of creativity at all. The goal of the reviews varies as well. Older reviews made comprehensive lists of methods for MGSs, while newer reviews tend to focus on more specific subsets of the literature. A brief description of the aim of the included reviews is listed under "Focus." Since every review tends to group works in clusters, we listed the criterion for the subdivision of the reviewed works under the column "Taxonomy." Finally, for each review we included the number of pages and the amount of references in their bibliography.

Some reviews also include sections or chapters are not directly related to MGSs: Nierhaus (2009) includes a chapter narrating the history of Algorithmic Composition; Williams et al. (2015) gives a brief review of studies that investigate emotional correlates of musical features; Herremans et al. (2017) gives an introduction both to the history of Algorithmic Composition and to the problem of Evaluation that we discussed in section 2.3. Finally, Briot et al. (2020), gives an introduction both to the ways in which musical data can be encoded and to deep learning in general. Moreover, to exemplify some deep learning techniques that were not yet used in MGSs, the authors cited some *visual* generation systems.

TABLE 1 | Summary of the reviews used as a starting point for the present survey.

| | Focus | Taxonomy | References | Mentions to creativity | Pages |
|----------------------------|--|----------------------|------------|------------------------|-------|
| Nierhaus (2009) | Broad review of all the methods for algorithmic composition in literature. | Method | 328 | 24 | 294 |
| Fernández and Vico (2013) | Broad (yet condensed) review of all the methods for algorithmic composition in literature. | Method | 337 | 38 | 70 |
| Williams et al. (2015) | Affective/emotional assessment integrated in algorithmic composition. | Expressive features | 123 | 0 | 24 |
| Herremans et al. (2017) | An objective-based taxonomy to better understand the state of the art in music generation. | Objective and method | 165 | 4 | 30 |
| Lopez-Rincon et al. (2018) | Brief introduction on a variety of AI methods used for music generation. | Method | 32 | 2 | 7 |
| Tatar and Pasquier (2019) | Generative musical agents. | Typology of agents | 205 | 122 | 50 |
| Briot et al. (2020) | Music generation using deep learning techniques. | Method | 212 | 37 | 303 |

For each, the focus for the choice of works to be reviewed is reported, as well as what the classification of the systems is based upon (Taxonomy), the number of references present in the reviews, the number of times creativity is mentioned, and finally the page count.

3.2. Methods for Music Generation

Many algorithms and techniques were applied to music generation, but it is possible to group those in some main categories. The subdivision we use is the one used by Fernández and Vico (2013), which is in turn based on prior reviews (Papadopoulos and Wiggins, 1999; Nierhaus, 2009). More recent reviews have either used this taxonomy or expanded specific subsets of its six classes. We decided to add a seventh category for Agents based systems, which is a meta-approach that has gained a lot of popularity and deserves to be treated separately. The seven categories are:

- Markov Chains;
- Formal Grammars;
- Rule/Constraint based systems;
- Neural Networks/Deep Learning;
- Evolutionary/Genetic algorithms;
- Chaos/Self Similarity;
- Agents based systems.

In the following sections, we will describe each of these approaches by citing works that implemented MGSs using techniques that fall in those categories, also briefly discussing how these approaches can be seen under the Process perspective using Boden's categories of creativity (see section 2.1.4).

3.2.1. Markov Chains

A Markov chain is a special stochastic process, i.e., a sequence of random events dependant on a time variable, that has a finite number of states, and the probability of the next state is only dependant on the current state (Brémaud, 2013). In practice, a Markov chain is described by a transition table, where rows and columns represent the states, and every cell (x, y) represents the probability of going from the state x to the state y . Since each row represents a probability distribution, the sum of all the cells in a row must be equal to 1.

If the last n states are used to determine the probability of the next state instead of just the last one, this is called n -th order Markov chain. These can be represented with a single transition

matrix as well, by constructing an equivalent first order Markov chain having A^n rows, where A is the number of states in the n -th order chain.

Due to their sequential nature, Markov chains are well fit to describe melodies, seen as a sequence of notes. The simplest way to implement a melody-generating Markov chain is to use a set of notes as the possible states, and to compute the transition probabilities between these notes by counting the occurrences of each transition in a given corpus to create a first order Markov chain.

This is what was done in one of the first MGSs ever described. Pinkerton (1956) created the “Banal Tune Maker” by analyzing the transitions of 39 nursery tunes by hand to create a transition matrix. The states used were the seven notes of the diatonic scale of C major (only one octave was considered), plus one extra symbol to indicate rests or notes that are prolonged over a beat. In this case the states of the chain only contain pitch information, requiring the use of other strategies to implement the rhythm. In this case, all the notes were kept to the same duration, and the extra symbol was used to introduce rests in the generated music. Of course, other approaches are possible, including implementing another Markov chain to handle durations.

The basic assumption underlying this simple approach, i.e., that the next note is only dependant on the previous note, is very flawed and only lead to musical results of little interest. Pachet (2002) used a more refined approach in the “Continuator.” He implemented a variable order Markov chains using prefix-trees to handle sequences of varying length (as opposed to n -th order Markov chains that will always consider n states) and also used a hierarchy of reductions: the system analyzed in a single chain pitch, duration and velocity, but was able to ignore some information when analyzing new input and comparing it to the learnt sequences. This was especially important in the Continuator because, as the name suggests, it was meant to listen to a musical input and continue it in real time. Being able to ignore part of the learnt information allowed the system to interact with previously unmet input, and to consider musical structures at various levels of detail.

Hiller and Isaacson (1958) used a different approach in their “Illiac Suite.” In their fourth experiment, they used Markov chains to generate sequences of motions and progressions rather than sequences of pitches and durations themselves, thus using the model to organize the notes at a higher level. The same idea of organizing higher structural levels via Markov chains was used more recently in the GEDMAS system (Anderson et al., 2013), whose goal was to generate Electronic Dance Music. To do so, a series of Markov chains were used to choose the general form of the song (i.e., a sequence of sections, each section being 8 bars long), to fill each section with a chord sequence, and finally to generate melodic patterns.

From the viewpoint of the creative process, Markov chains risk to reuse a lot of material from the learnt corpus non creatively, even plagiarizing when the order of the chain is too high (Papadopoulos et al., 2014). However, they can also result in novel combinations of smaller sections such as motifs, in a way that can be considered Combinational Creativity, but will hardly go beyond that limit unless other techniques are also employed. They also remain useful at higher structural levels (like the above example of GEDMAS), where high levels of creativity are not usually as important as when the melodic material is being generated.

3.2.2. Formal Grammars

Chomsky (1957) introduced the concept of Generative Grammars, a tool for the analysis of natural language that became extremely influential in linguistic studies. The same idea was applied to musical studies, most notably by Lerdahl and Jackendoff (1985), who tried to design a Generative Grammar for the description of music starting from music analysis concepts introduced by Heinrich Schenker in his book “Free Composition” (Schenker, 1979), that well fit the concept of *rewriting rules*, that is at the basis of Chomsky’s grammars.

A Generative Grammar is composed of two alphabets: terminal symbols and non-terminal symbols (or variables). A set of rewriting rules is given over the union of these two alphabets, that allow to transform variables into other symbols (both variables and terminals). The generated language is the set of all the strings of terminal symbols that can be obtained starting from a special variable chosen as starting point (usually called *S*) and applying any number of rewriting rules in sequence.

Grammars can be seen both as an analysis tool and as a generative tool. For example, Steedman (1984) compiled a Generative Grammar to describe Jazz chord sequences: Pachet (2000) describes a system that is in part inspired by Steedman’s analysis to tell apart blues songs and non-blues songs, while Chemillier (2004) implemented Steedman’s grammar creating a software for music generation.

Chord sequences can be very easily encoded as symbols, but, if an adequate alphabet is given, it is possible to use Grammars to generate any kind of musical information. Hamanaka et al. (2007) describe a system for the automatic analysis of scores based on Lerdahl and Jackendoff’s Generative Theory of Tonal Music, formalizing in details a grammar to describe musical material. This was then used to create variations on melodies by altering the derivation trees (a graphical representation of the

applied rewriting rules) (Hamanaka et al., 2008). Quick (2011) implemented a software to generate three voice harmonies using a Grammar derived from Schenkerian theory.

L-systems (Lindenmayer Systems) are a variant to Generative Grammars that has been used for music generation. Their main difference from Grammars is that they implement parallel rewriting, thus applying all the rewriting rules at once instead of only one at a time. This characteristic makes these system less apt to sequential data, like simple melodies, and have been used to generate stunning visual effects. When applied to music generation, the most common approach was to map visual data generated by L-systems either to score information (Prusinkiewicz, 1986; Mason and Saffle, 1994; Nelson, 1996) or to arrange a sequence of musical segments (Langston, 1989; Supper, 2001).

Formal grammars can be seen as a precise definition of a conceptual space, which is then explored when generating music. In this sense, the compilation of the rewriting rule can be seen as Transformational Creativity, but this is usually performed by a human rather than a computer. An exploration of the conceptual space of the possible rules can be seen as a meta-level creativity, which as Wiggins (2019) showed is indeed a form of Transformational Creativity, but this can be extremely hard to implement effectively in a CC system, since the compiling of a formal grammar requires careful study even when done by a human to ensure valuable results.

Another related approach is that of Transition Networks: finite state automata that can parse languages similarly to what Generative Grammars do. The most notable example of Transition Networks applied to MGS is that of David Cope’s Experiments in Musical Intelligence (Cope, 1991, 1992). His approach was to use pattern-matching algorithms to analyze “signatures,” short musical sequences that define the style being analyzed, and to determine when and how to use those signatures. After the analysis phase, the collected information is encoded in a Transition Network that is then used to generate new music in the style of the composer that was analyzed. While the results are sometimes impressive, they are arguably not very creative, since they just reuse material taken from the learnt corpus in a way that can be at most be seen as Combinational Creativity (Wiggins, 2007). Possibly, this is one of the reasons why there is not much research on Transition Networks for music generation beside Cope’s works.

3.2.3. Rule/Constraint Based Systems

Music theory traditionally describes rules that help to guide the compositional process. While composers regularly break those rules, it should come to no surprise that those rules have been used to implement MGSs since the early days of Algorithmic Composition, like in the first two movements of the Illiac Suite (Hiller and Isaacson, 1958). Generative Grammars can be seen as an implementation of such rules, but the systems we refer to in this section are usually unable to generate musical material from scratch, and either start from some input material (like in the case of harmonization software) or use other methods, sometimes even random generation, to have a starting point that is then refined through rules.

The inclusion of rules can be implemented in many ways, for example as a final validation step, or to refine intermediate results. One natural way to implement rules in a MGSs is to use *Constraint Programming*, whose declarative nature is well fit to describe music theory rules. A survey on works that have used Constraint Programming to model music theory (not only with the goal of generation) can be found in Anders and Miranda (2011).

One of the most influential researchers within the scope of music generation through constraint is Ebcioglu, who first implemented rules of fifth-species counterpoint into a Lisp program, and later implemented a custom logic language that he used to create CHORAL, a system for the generation of Bach-like chorales that uses some 350 rules for the generation of melodies and harmonization (Ebcioglu, 1988, 1990). The difficulty of designing such a system lies in the complexity of explicitly coding a sufficient amount of rules, many of which often do not have a formal definition in musicology literature. Moreover, there is a tradeoff between adding more rules to obtain results that better fit the style that is being modeled and leaving less constraints to be more open to different styles of music.

Constraints can be used to model more abstract features, rather than explicit music theory rules: Herremans and Chew (2016b) defines a way to describe tension in musical pieces based on a geometric model of tonality called the *Spiral Array* (Chew, 2014). Herremans and Chew (2017) used that tension model in a MGS that is capable of generating new music following the tension pattern of an input piece, by first generating random notes and then applying optimization methods (in particular, Variable Neighborhood Search) to change the notes in order to satisfy constraints defined by the chosen tension model.

Techniques for optimization such as integer programming can be useful as a selection technique when more than one possibility is available. For example, Cunha et al. (2018) describe a MGS that creates guitar solos by concatenating guitar licks. This approach is somewhat similar to a transition network, but in their implementation the concatenation of any two licks had a defined transition cost, and through a branch-and-cut algorithm it was possible to compute the optimal solo. The computation of transition costs was in itself another example of integration of rules: in that work eight rules were described to assign the transition cost between licks.

The integration of rules and constraints in a creative Process can be seen in two ways: the first is considering those rules as bounding and reshaping the conceptual space, the second is to see rules and constraints as a guidance in the exploration of the conceptual space. Either way, the use of rules can result in a more efficient Exploratory Creativity, although they might reduce the size of the conceptual space (or limit the explored areas) thus limiting the variety of the output.

3.2.4. Neural Networks/Deep Learning

The increased computational power of computers and the widespread of general purpose GPU programming recently made deep learning techniques extremely popular, with applications that span from natural language processing, to image and video editing, to, of course, music generation. The survey by Briot

et al. (2020) is specifically focused on these techniques, and gives an exhaustive overview of how machine learning has been used in MGSs.

While the interest in these algorithms grew exponentially in the last decade, the first MGS to use Artificial Neural Networks is that of Todd (1989), who used a three-layered Recurrent Neural Network (RNN) to generate monophonic melodies. Recurrent Networks reuse the results of the computations from previous steps when new input is given, allowing them to encode temporal sequences. This is of vital importance when generating melodies, making them a typical approach for MGSs that use deep learning (unlike, for example, Convolutional Neural Networks that are more apt for the elaboration of images). Nonetheless, there is also room for standard feed-forward networks: Lewis (1991) trained a network with musical patterns ranging from random to well-constructed, to learn a measure of “musicality” that is then used by his MGS to select pleasant compositions.

As already mentioned, RNNs are a popular choice for music generation. In particular, LSTMs (Long-Short Term Memory networks) Hochreiter and Schmidhuber (1997) are a special variant of recurrent networks that use special gates to decide the amount of information that is taken from novel input and what is maintained from older inputs. This control over the data flow allowed LSTMs to be both more efficient and effective than standard RNNs in a wide range of applications, and have been used for music generation as well. The first music generation LSTM was applied to blues improvisation (Eck and Schmidhuber, 2002a,b). Traditional music was instead the focus of *folk-rnn* (Sturm et al., 2016), that analyzed over 20000 pieces in textual (abc) notation. A more advanced approach is used by DeepBach, Hadjeres et al. (2017) that generates chorales in the style of Bach (whose chorales made the training set for the software) using two LSTMs, one going forward and one going backwards in time, together with one feed-forward network to consider contemporaneous notes. The results of these networks is then handled by a final feed-forward network that joins the results in final piece. The rationale behind this choice is explained by the goal of generating *counterpoint*, which requires knowledge of both the previous and the following notes. This gives an example of how it is possible to design complex architectures using many layers of Neural Networks, but the complexity comes with a price in terms of computational time.

Another deep learning approach that is of great interest to CC is that of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). The idea behind this method is to train two networks at the same time, one that generates artifacts imitating what is learnt from real-world examples, and the other trying to discriminate between real and imitated artifacts. As one gets better, the other must get better as well in order to “beat” the other network (thus making them “Adversarial”). The two networks can be simple feed-forward networks, but these are not the usual choice for music generation. For example, the eloquently called C-RNN-GAN (Mogren, 2016) uses recurrent networks (in particular LSTMs) in an adversarial architecture to generate polyphonic music. MidiNet (Yang et al., 2017) uses convolutional layers instead: Convolutional Networks are trained to reduce the dimension of the input, usually starting from

bidimensional input. This approach is often used on images, so when applied to MGSs the input of the Network is often some graphical representation of music, such as piano rolls. It is important to be aware that while images have two dimensions that are equivalent (both represent displacement in space), graphical representations of music show two non equivalent dimensions, usually pitch and time, possibly leading to less reasonable results (Briot and Pachet, 2018).

Machine learning has strong implications for what concerns creativity: Turing advocated that learning machines would have been the key to beating the Imitation Game (Turing, 1950), and the very concept of learning is strongly related to the possibility of expanding and changing both the conceptual space and the means of exploring it, possibly reaching Transformational Creativity. However, Boden (2004) argues that connectionist systems cannot reach human levels of creativity, and Bringsjord et al. (2003) argues that the “learning” involved in a neural network is not enough to pass the “Lovellace Test” (see section 2.3.1). Both those works seem to only consider classic feed-forward neural networks rather than state-of-the-art deep learning approaches that are able to learn the representation and encoding of raw data (Briot et al., 2020), which can be seen as the definition of the conceptual space for these systems. From this viewpoint, these systems are the closest we have come to implementing Transformational Creativity, but the black box nature of these systems make it hard to pinpoint exactly how the generation process works and how the corpus information is used, making it also hard to control the output of the system (as we will discuss in section 4.1), a capability that is considered fundamental by certain definitions of creativity (Riedl, 2014; Jordanous and Keller, 2016).

3.2.5. Genetic/Evolutionary Algorithms

The general idea behind Genetic (or Evolutionary) Algorithms is that, starting from a population of random solutions to a problem, it is possible to combine those solutions to obtain new solutions, and by selecting the ones that better answer the problem it is possible to get closer and closer to the optimal solution to the original problem. Thus, to solve a problem via Genetic Algorithms, it is necessary to have (Sivanandam and Deepa, 2008):

1. The ability to generate random but suitable solutions to the problem as a starting population;
2. A way to evaluate the “fitness” of a solution;
3. The ability to mutate and recombine those solutions.

In the field of music generation, the points 1 and 3 are for sure available (once a representation of musical material is chosen), but it is hard to evaluate how good a solution is (as already discussed in Section 2.3). It might be difficult even just giving a precise definition of what the problem is. Nonetheless, Genetic Algorithms have often been used to implement MGSs.

Possibly, the most famous Genetic MGS is GenJam, designed by Biles (1994). The system is meant for Jazz improvisation, where a human player interacts with the software that outputs both the pre-made musical base and solos generated on-the-fly by evolving the human improvisation it has just

listened to. Originally, the fitness function was implemented by having a human decide if the output was good or bad, an approach that is usually referred to as “Interactive Genetic Algorithm.” This generates a bottleneck for the system, as a lot of human intervention is required. A successive version (Biles et al., 1996) used an Artificial Neural Network as a fitness function, but it led to unsatisfactory results. In the end, the author resolved to completely eliminate the fitness function (Biles, 2001). Basically, the algorithm retains the ability to mutate and compose licks, an ability that is used to respond to musical input in a way that incorporates the human improvisation without being a mere copy, but since there is no more evaluation of the fitness, GenJam is no more a genetic algorithm.

GenJam passed, through his versions, some of the most common approaches to the definition of a fitness function. Another approach is to use rules taken from music theory to design a fitness function. This is the approach chosen by Phon-Amnuaisuk et al. (1999). In that case the goal was the harmonization of a given melody, and the fitness function incorporated rules of harmony describing forbidden and preferred intervals and motions. In this case, the use of genetic algorithms becomes a way to explore a space of possibilities described by the chosen rules. One might wonder if this is better or not than just generating samples following those rules, as described in the previous section. Indeed, Phon-Amnuaisuk and Wiggins (1999) found that their genetic implementation was outperformed by a rule-based system using the same set of rules that were incorporated in the fitness function. The authors argue that having explicit control over a system’s knowledge will lead to better results and more powerful means of exploration: while the authors do not scorn genetic algorithms in general, it seems that this approach cannot give such explicit control over the knowledge of the system, and thus other systems should be preferred when explicit knowledge is available.

Genetic Algorithms offer many other forms of hybridization, since the representation used by other algorithms can be evolved genetically. For instance, it is possible to evolve the rules of a grammar (de la Puente et al., 2002), or to evolve the parameters of a Markov chain (Werner and Todd, 1997; Bell, 2011) or of a Cellular Automaton (Lo, 2012). We already mentioned that rules, Neural Networks and human assessments can be incorporated in the fitness function for a Genetic algorithm. It is worth mentioning that Markov chains have been used for the same goal (Lo and Lucas, 2006). Markov chains can also generate the initial population, obtaining starting point that is better than random, possibly leading to convergence to good solutions with fewer generations (Manaris et al., 2011).

The evolutionary approach is in itself an exploratory process: the combination of two individuals from the population pool is a combinational process, but the use of a fitness function guides the exploration toward promising areas of the conceptual space, which is bounded and defined by the genetic encoding of the individuals. Losing the fitness function, or having one that is unable to effectively guide the exploration, reverts the mechanism to pure combinational creativity, where elements of

the conceptual space are joined and mutated hoping to find interesting unexplored combinations.

3.2.6. Chaos/Self Similarity

Musical compositions show some degree of self similarity, both in the musical structures and in its spectral density (Hsü and Hsü, 1991), roughly following a $1/f$ distribution, at least for pieces that are deemed pleasant to listen to (as opposed to random compositions) (Voss and Clarke, 1978).

Starting from these considerations, fractals and other self-similar systems have been used to generate musical material. The results of such systems are usually not regarded as a final output, but rather as an inspiration for human composers (Bidlack, 1992). Another approach is to generate self similar structures rather than directly generating self similar melodies: Leach and Fitch (1995) generated tree structures like those described by Lerdahl and Jackendoff (1985), by tracing the orbit of a chaotic system, and mapping the computed values to different hierarchical levels of the tree.

Another approach is to use Cellular Automata, dynamic systems composed of many cells, whose states are updated at discrete times using a set of transition rules. Famous examples include “Game of Life” by Conway (1970), and the systems studied in “A New Kind of Science” by Wolfram (2002)¹. Like other fractal systems, Cellular Automata tend to generate melodies that are not too pleasing, and often need further human intervention. CAMUS is a MGS that is based on two different Cellular Automata, whose cells were mapped to sequences of notes and to different instruments (Miranda, 1993). A later version used a Markov chain to specify rhythm, but despite the effort to create a full MGS, the authors still admit that the results can often be not very pleasing, but can become interesting “for the composer who is prepared to put a little effort into the system” (McAlpine et al., 1999). Miranda (2007) later argued that Cellular Automata are more effective for sound synthesis, rather than for MGSs.

Since the decision making of these systems is based upon chaotic and random processes it is difficult to describe them using Boden’s categories, and the usual lack of aesthetic value of the results suggests that this is not a good example of CC but rather a way to explore unusual melodies. For these reasons, these systems are arguably less interesting to AI practitioners, but were included for completeness. Nierhaus (2009) provides a good review of these approaches, that are given less consideration by later surveys.

3.2.7. Agents Based Systems

A software agent is an autonomous piece of software with perception and action capabilities. Any software with such capabilities can be seen as an agent (including many of the systems described in the previous sections), but the definition becomes especially interesting when multiple agents cooperate within a single software, that can be referred to as a Multi Agent

System. This is not a specific algorithm for music generation, but rather a meta-technique that has gained popularity among researchers, as testified by Tatar and Pasquier (2019).

The use of agents in MGSs makes it easy to model certain musical behaviors. Voyager (Lewis, 2000) uses 64 player agents that generate melodies according to one of various pitch generation algorithms written by the author, according to his own taste, and a behavior model that describes the general timbre, tempo, pitch range and other features that regulate the development of the piece. This models a band where everybody is improvising, but still follows some general agreement. Lewis has played together with Voyager, both in recordings and live: in this setting one can also consider the human performer as one additional agent of the system.

MASs are also useful to model social interactions: once each agent is given specific characteristics (one could say, a personality), the interaction between different agents can take into account the difference in their characteristics, either in a conflict or in an agreement. For example, Kirke and Miranda (2011) introduces a system (later called MASC; Kirke and Miranda, 2015) where each agent has a specific “emotion” and the ability to express it by “singing” to another agent. The other agent will be affected by the mood expressed by the singer, adapting his own internal state. Moreover, their internal state also defines if the listener will “like” the song, incorporating it into his own song.

Taking further the same idea, the agents can implement cognitive models that regulates their interaction with the others. One such model is the Belief-Desire-Intention Architecture. For instance, Navarro et al. (2014, 2016) describe a system with the goal of generating harmonic sequences, where two particular agents, the *composer* and the *evaluator*, have beliefs based on music theory and desires (one to compose and the other to evaluate the generated composition). The intentions are represented by the algorithms implemented to apply and verify the theoretic rules that form their beliefs, and are influenced by the communication between the two roles.

Since the use of agents is a meta-technique rather than a specific algorithm, it is not possible to frame them from the Process perspective, but it is useful to consider the Person and Press perspective. The use computational means to give a “personality” is important to obtain results that are affectively relatable for humans, possibly making it easier to pass Turing-like tests. Moreover, the influence of other individuals is an important factor in human creativity (Amabile, 1983b), and is thus an interesting direction for research in CC (Saunders, 2019).

4. OPEN CHALLENGES FOR MUSIC GENERATION SYSTEMS

One of the goals of this review is to give pointers to any reader who is approaching Computational Creativity (CC) in general and Music Generation Systems (MGSs) in particular some pointers on what still needs to be addressed and the open challenges in the field. To do so, we extracted a list of problems and challenges that were identified in the reviews

¹A web application developed by Wolfram Research, Inc. that allows users to generate music using Cellular Automata is available at <http://tones.wolfram.com/generate/>.

listed in **Table 1**, especially looking at those sections were the reviewers gave indications for future directions. The different surveys used variable terminology, often without giving precise definitions for the challenges they mentioned. To group them, we first tried to cluster those problems that we believed to be similar, and then gave precise definition for each cluster. We then re-read the problems descriptions in the reviews and marked as “mentioned” the clusters that was the closest to the descriptions. The clusters that were never mentioned were removed, and the remaining ones were given fitting names and are listed in **Table 2** as the challenges we identified. The table also lists for each challenge which reviews mentioned it. The precise definition of each challenge is given in the next paragraphs. It is worth noticing that all the reviews mentioned Evaluation as an open challenge, and nearly half also mentioned Creativity (as opposed to mere imitation) as still lacking in most systems. Since we already widely discussed Evaluation, this won’t be further treated. Creativity will be instead treated in each paragraph, as we want to make this review useful for CC as well as MGS research. To do so, we will try to categorize these challenges using the dimensions of creativity described by Jordanous and Keller (see section 2.1.6). For each of the other challenges, we will give precise formulations of what is the problem to be addressed, citing examples of works published in the last 10 years that have faced these problems and gave insights to what solutions could be used to overcome those problems and to achieve higher creativity.

4.1. Control

Control refers to having the possibility to choose specific features that the output of the MGS will exhibit.

Having control over certain features of the output of a MGS can be, depending on the used algorithm, trivial. But, with

more data-driven approaches like machine learning, it becomes less obvious what can be done to affect the output. It is not surprising that this issue was only mentioned in a review focused on deep learning.

Since data-driven approaches are meant to learn features from their input, one simple way to influence the features of the output is the selection of the training set. This approach is to some extent used by every corpus-based system, knowing that learning on Folk music will be very different from learning on Bach chorales. The problem with this approach is that it does not allow a good granularity of control, and any change on the input would require retraining the system, a task that can be very time consuming.

The same idea is applied in a slightly different fashion by Ekeus et al. (2012). Their approach was to generate a set of randomly sampled Markov chains, which were evaluated with an approach based on Information Theory. These were employed in a MGS that allows the users to select a point in a triangular space where the vertices represent periodicity, repetition, and noise. The chosen point is mapped to the features that were evaluated for each Markov chain, and the most appropriate one is selected and used for melody generation.

The same approach can be used in Neural Networks by altering the parameters that make up the network, but this can be much more intimidating, due to the excessive number of parameters involved and the difficulty of understanding their meaning (Sturm, 2018). A way to obtain this is proposed by Kaliakatsos-Papakostas et al. (2018), who used a recurrent network trained on a small dataset (made of only three pieces) that was augmented specifically to address the features the authors wanted the user to be able to manipulate, in order to study how the parameters are affected, and to be able to alter them accordingly in the generation phase.

Control is related to the creative dimension of “Active involvement & persistence” which suggests that the creative agent is in control of the generation process. Using deep learning to achieve this can be extremely hard, although many advancements in this direction are being made. We suggest to use techniques that allow for simpler tuning over the features one wishes to control, by either using appropriate representations (see section 4.5) or by explicitly limiting those features with rules. Machine learning can be used in conjunction with these approaches to ensure other creative features, such as “Variety, divergence & experimentation.”

TABLE 2 | For each of the identified challenges, an X is added under every review that mentions it.

| | Nierhaus (2009) | Fernández and Vico (2013) | Williams et al. (2015) | Herremans et al. (2017) | Lopez-Fincon et al. (2018) | Tatar and Pasquier (2019) | Briot et al. (2020) |
|---------------|-----------------|---------------------------|------------------------|-------------------------|----------------------------|---------------------------|---------------------|
| Control | | | | | | | X |
| Narrative | | | | | | | |
| Adaptability | | | X | | X | | |
| Emotion | | X | X | | | | |
| Hybridization | X | | | | | | X |
| Rendering | X | | X | | | | |
| Structure | X | X | X | | | | X |
| Mapping | X | | | | | | |
| Playing | | | | | | | |
| Difficulty | | | X | | | | |
| Evaluation | X | X | X | X | X | X | X |
| Creativity | X | X | | | | | X |

4.2. Narrative Adaptability and Emotion

Narrative Adaptability refers to the capability of the MGS to convey a sense of development (Narrative) in the generated music, giving a more complex meaning to the piece. *Emotion* refers to the capability of the MGS to convey specific emotions with the generated music.

These two are treated together because it is possible to convey different emotions in different sections of the piece, one of the main aspects of *Narrative Adaptability*. Both of these can be seen as a special instance of *Control*, where the features that are being controlled relate to emotional aspects or to specific events of the narration. This is especially relevant in non-linear media (like

video games) where the Narrative must adapt in real-time to the events in the media.

The study of Emotion in music has a long history (Juslin, 2010) and, as can be seen in the review by Williams et al. (2015), has often been considered in MGSs. Narrative Adaptability is less commonly found, despite the fact that such adaptability is something that human composers could never achieve without the help of a computer, making it an interesting field of investigation. Ventura et al. (2009) present an installation implementing a typical architecture for emotion-aware systems: an emotion (expressed as values in the valence/arousal plane; Hunter and Schellenberg, 2010) is detected (in this case by analysing the movements of the users via webcam) and then used as the input for the MGS. To do so, some features that are known to be related to emotional expression are manipulated, such as tempo, pitch range and loudness (Oliveira and Cardoso, 2007). A similar architecture is used by Scirea et al. (2018) to add music to Checkers: the MGS analyzes the board to understand how risky the situation is for the player, and then generates music that emotionally expresses the level of risk.

Mezzo by Brown D. (2012), Brown D. L. (2012) uses a different approach that takes its roots in classical music: the use of *Leitmotifs*. In a video game setting, some characters and situations are given a theme (composed by a human), and when those are encountered in the game, a message is sent to *Mezzo*. This will use the themes triggered by the messages, blending them together to generate a music that expresses the current situation. Similarly, when music is used within human-computer interaction, is useful to detect musical features in the human interaction to generate music that matches the emotional content as a feedback (Carnovalini and Rodà, 2019b; Carnovalini et al., 2019).

Another approach that does not necessarily involve a full MGS, but can be used to increase its adaptive capabilities, is that of automatically generating transitions between pre-composed sections, to be able to connect sections without knowing a priori when one will end and one will start (Horner and Goldberg, 1991; Gillespie and Bown, 2017). This is usually applied to human-made compositions, but it could be easily applied to a MGS, as long as it is capable of generating the next musical piece in advance, since it is needed for the transition generation.

These challenges are especially important for creativity, as they address “Intention & emotional involvement” and “Progression & development” and “Social interaction and communication,” and in general are fundamental for the affective perception of the machine, which in turn is important to pass Turing-like tests. One research direction we suggest is to study how different expressive features can influence each other, and how to select one specific expressive technique to convey certain feelings rather than altering all the features that are linked to that emotion, so that systems could be able to generate, for example, a sad piece which is also fast-paced. This can also improve “Variety, divergence & experimentation” of the generated pieces and possibly lead to more “Originality.”

4.3. Hybridization

Hybridization refers to the use of more than one technique for music generation in a single MGS.

This is the only point of this list that does not concern a quality of the output, but rather a characteristic of the system itself. The need to go beyond a single method for the generation of music was already noted 20 years ago (Papadopoulos and Wiggins, 1999), but the call for hybridization is relevant to this day. The rationale behind this idea is that since there is not a single method that has been proved to be more effective than the others, nor to be capable of addressing all the issues that a MGS must face, it is important to take advantage of different approaches. Nonetheless, using multiple algorithms is obviously expensive for the development, and in general it is hard to witness in the early stages of any project. Moreover, researchers are often more interested in applying a specific technique for music generation rather than creating a complete MGS that would benefit from hybridization.

Some approaches are more prone to being used in an hybrid context than others: we have already discussed how Genetic algorithms and rules or constraints are often coupled with other algorithms, but other approaches are possible. Eigenfeldt and Pasquier (2009) describes how the various versions of the *Kinetic Engine* have used different algorithms for designing agents capable of generating rhythm, melodies, and harmony. In the later versions, agents with different roles interacted with each other to generate both rhythm and melody, and also a Markov chain was used to influence the harmonic progression Eigenfeldt and Pasquier (2010). This gives an idea of how, in an agents-based system, it is possible to delegate different tasks to different agents, that can each implement a different strategy when generating their respective content.

A somewhat similar subdivision of tasks is proposed by Carnovalini and Rodà (2019a), where the process of composing a melodic phrase is divided in successive steps: generation of pitch succession, generation of rhythm, and finally generation of expressive variations of intensity and timing. Each of these steps follows a different algorithm, but some information is passed on through each step. In particular, all of the steps use information about the importance of each of the generated notes, dividing them with a Schenkerian approach (Simonetta et al., 2018). The authors argue that this idea can be further extended to other tasks (such as form generation and harmony), including both deep learning and classical AI algorithms, trying to find the optimal combination for each task (Carnovalini, 2019).

The use of expressive musical performance generation systems (Widmer and Goebel, 2004; Canazza et al., 2012, 2015), that are sometimes embedded in MGSs as the one just cited above, can also be seen as a form of hybridization, but will be better discussed in the next section.

We believe that systematic use of Hybridization could be one of the most prolific research direction for CC, since it could help researchers expand different dimensions of creativity using different techniques for each. Moreover, giving a variety of compositional approaches to a software could be seen as giving it a better “Domain competence,” and being able to choose between techniques can improve “Variety, divergence & experimentation.” Explicitly modeling into the software what different techniques are more apt for and changing behavior according to user’s requests could be an interesting research direction.

4.4. Rendering

Rendering refers to the quality of the audio (meant as waveform) that is generated by the MGS.

This might seem redundant, as the quality of the generation is obviously important in a MGS. But in many cases, MGSs only handle symbolic music generation, usually as MIDI or MusicXML files, and the audio is generated with simple software MIDI synthesizers, which are far from giving good renditions of any musical composition.

We already mentioned that it is possible to add expressive performance to a generated piece in order to improve its audio rendering. This is usually done through existing algorithms that are applied to the music after it is generated. A review of existing algorithms can be found in Kirke and Miranda (2009).

Another way to improve the musical rendering is to use automatic orchestration techniques: rather than having a predetermined instrument to play the generated piece (piano seems to be a popular choice) it is possible to generate musical material that is then assigned to different instruments (Handelman et al., 2012) or to have a set of possible instruments from which to choose from and that can intervene at different moments of the composition (Anderson et al., 2013).

Brunner et al. (2018) describe a system that uses both expressivity and orchestration to perform Style Transfer through Variational Autoencoders (Kingma and Welling, 2013). Style Transfer tries to apply a certain style (for example, defined by a certain composer or genre) to an existing musical piece that was not originally meant for that style. This can of course be applied to computer-generated music as well, although we are not aware of any work in literature that has yet tried this approach.

All these approaches generate some sort of variations after the score generation process is over. This excludes any possibility to render audio in real-time, as the generation phase must be over for these algorithms to function. A different approach that has been less explored is to generate music and its expressive variation at the same moment: one example is PerformanceRNN (Oore et al., 2018).

A completely different approach is to model music directly at the audio level, thus implicitly generating the rendering as well. This approach is challenging for many reasons, including computational complexity and the difficulty to capture semantic structures from raw audio. Nonetheless, Dieleman et al. (2018) found that using Autoencoders it is possible to obtain realistic results that remain consistent for tens of seconds, meaning that local structures can be understood and modeled directly from the audio.

One could argue that the creative task we are interested in is the composition, while the rendering is delegated to musicians. While it is true that in some cases computers generate music that is meant to be played by humans, it is more often the case that computers directly play the generated music themselves. Moreover, in the context of evaluation of CC, having a good audio rendering can influence human evaluators, so it should not be overlooked (Oore et al., 2018; Carnovalini and Rodà, 2019a). More generally, Rendering can be seen as part of “Generating results”: while scores are results in themselves, the fruition of music is through sound. Therefore, to add “Value” to the output,

Rendering must be considered. We are not aware of any research comparing user preference of computer generated music that is emotionally rendered vs. “deadpan” executions, but that would certainly be an useful contribution to CC research.

4.5. Structure and Mapping

Structure refers to generating longer pieces, containing reasonable repetitions and subdivision of different sections, usually recreating some kind of musical form. *Mapping* refers to the problem of handling different representations of music and choosing the most appropriate one for the generation of musical content.

While the first is notoriously difficult for MGSs, the latter is an issue that is often not considered, as usually a certain representation is chosen a priori. There are instead notable proposals in literature that further the possibilities for MGSs using specific representations of music.

Herremans and Chew (2016a, 2017) used a specific data structure, the *Spiral Array* (Chew, 2014), to compute the *tension profile* of a musical piece. This profile is used to generate a new piece that follows the same profile, through constraint programming. Starting from a specific representation for tension structures, the MGS is able to create longer pieces with convincing structure. One might argue that the structure is simply being copied, but a possible extension to this work could possibly generate novel tension patterns using the same ideas.

Representations based on Schenker's or on Lerdahl and Jackendoff's theories are studied, since these can capture different levels of structural information. Most works only have the aim of automatic analysis of musical pieces (Marsden, 2010; Marsden et al., 2013; Hamanaka et al., 2016, 2017), but others have used this approach to generate music that follows a defined structure (Groves, 2016; Carnovalini and Rodà, 2019a).

Other systems approached the problem of structure without any specific representation. GEDMAS (Anderson et al., 2013) explicitly generates structure, seen as successions of 8-bar segments, through a Markov chain. Medeot et al. (2018) describe StructureNet, a neural network that studies occurrences of repeats (either of rhythm or of interval sequences), and that can be embedded in a larger MGS influencing the generation process according to the learnt structures of repeats.

Structure is strongly linked with the creative dimension of “Progression & development,” and can be linked to the challenge of Narrative Adaptability as well. Once again, we suggest to hybridize different approaches, possibly using different techniques at every level of representation to consider the development of a piece at a macro level before considering the local melodic and harmonic content.

4.6. Playing Difficulty

Playing Difficulty refers to the ability of an MGS to regulate the difficulty for a human to play the generated music.

This can be seen as a specific instance of *Control*, where the feature that must be controlled is the technical difficulty of the output. This problem only becomes relevant when the output of the MGS is meant to be played by a human, which is often not

the case: this might be the reason why this issue has hardly been acknowledged in literature.

The review by Herremans et al. (2017), that is the only one to mention Playing Difficulty, only cites a couple recent works that have considered the issue. One is Sabastien et al. (2012), that designed seven criteria used to estimate the difficulty of a piano piece, in order to suggest pieces to learn to students. A similar approach for guitar is presented in Xambó et al. (2018), based on known chords. Both these systems are not MGSs, but could be implemented as a constraint or as a fitness function in a MGS. Another work is that of McVicar et al. (2014), that generates guitar solos in tablature form. This is indeed a MGS, but it does not really consider the difficulty of the generated solo, but rather uses an algorithm to minimize fingering difficulty, without affecting the generation of the piece. Extending on the same idea, Ariga et al. (2017) created a guitar solo generator that considers the fingerings as a way to measure and control the difficulty of the generated solos. Nakamura and Yoshii (2018) describes a system that creates piano reduction of ensemble scores, capable of generating reductions with different levels of difficulty based on fingering and tempo information.

On the opposite side of the difficulty spectrum, Pachet (2012) describes a system that can generate virtuoso solos, using variable-order Markov chains trained on a dataset of virtuoso solos. Arguably, increasing the Playing Difficulty of a generated piece is easier than lowering it (without losing musicality), but the work by Pachet was motivated by a study of creativity in solos and not of difficulty itself.

To be able to change the playing difficulty of a piece, one needs to increase the “Domain competence” considering for example the physical characteristics of the instruments that will be used to perform the piece, making this challenge also relevant to the perception of CC. Choosing an appropriate level of difficulty can also improve the “Social interaction and communication,” since if one wishes to create a MGS to interact with humans, it is important to tune the difficulty to the end user’s ability. One possibility to deepen this relatively unexplored branch is to use published exercises books for the learning of instruments to extract features correlated to the difficulty level.

5. CONCLUSIONS

We presented a broad introduction to the field of Computational Creativity, and to Music Generation Systems in particular. In the first part (section 2), we described the main concepts needed to understand research on creativity (both human and

computational), analyzing a variety of definitions and studies on what makes People, Processes, Products be perceived as creative (to the Press). We reviewed some works on the evaluation of creativity, which is one of the main challenges for computational creativity, and is strongly intertwined with trying to give a formal definition of creativity. In the second part (section 3) we focused on the specific task of Music Generation, starting from existing reviews on the subject. We described the main approaches to Music Generation in the literature, giving examples for each. Finally (section 4), we listed a set of issues that need further development as identified by the reviews we analyzed. For each, we listed possible approaches used to face these issues that have been proposed by papers published in the last 10 years, discussing how facing these challenges can also lead to an improvement in the perceived creativity.

What hope that this review can serve as an introduction to the research on Music Generation that can give all the necessary bases to any researcher who wants to approach Music Generation from a Computational Creativity point of view. As we already underlined, the main problems in this field derive from the fact that often researchers fail to clearly state the goals of their research, and consequently cannot give a good evaluation of their work. This review can help frame new research within the scope of Computational Creativity, and give an indication of what still needs to be done. For instance, we believe that often researchers have chosen a specific method or algorithm and developed Music Generation Systems with the goal of using that method rather than trying to create a complete Music Generation System that could benefit from the use of different approaches to face different issues. To this goal, we advocate that a well studied hierarchical hybridization could give means to face many of the open challenges listed above, and possibly also allow for easier comparison between different methods, thus opening new possibilities for evaluation.

AUTHOR CONTRIBUTIONS

FC selected and studied the sources, designed the structure of the manuscript, and wrote the first draft of the manuscript. AR contributed with supervision over the study of the literature and the writing of the manuscript. All authors contributed to manuscript revision, read and approved the submitted version.

FUNDING

FC was funded by a doctoral grant by University of Padova.

REFERENCES

- Agres, K., Forth, J., and Wiggins, G. A. (2016). Evaluation of musical creativity and musical metacreation systems. *Comput. Entertainment* 14:33. doi: 10.1145/2967506
- Amabile, T. M. (1983a). “A consensual technique for creativity assessment,” in *The Social Psychology of Creativity*, Springer Series in Social Psychology, ed T. M. Amabile (New York, NY: Springer), 37–63.
- Amabile, T. M. (1983b). The social psychology of creativity: a componential conceptualization. *J. Pers. Soc. Psychol.* 45, 357–376. doi: 10.1037/0022-3514.45.2.357
- Amabile, T. M., Conti, R., Coon, H., Lazenby, J., and Herron, M. (1996). Assessing the work environment for creativity. *Acad. Manage. J.* 39, 1154–1184. doi: 10.5465/256995
- Anders, T., and Miranda, E. R. (2011). Constraint programming systems for modeling music theories and composition. *ACM Comput. Surv.* 43, 1–38. doi: 10.1145/1978802.1978809

- Anderson, C., Eigenfeldt, A., and Pasquier, P. (2013). "The generative electronic dance music algorithmic system (GEDMAS)," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE'13) Conference* (Palo Alto, CA), 4.
- Ariga, S., Fukayama, S., and Goto, M. (2017). "Song2guitar: a difficulty-aware arrangement system for generating guitar solo covers from polyphonic audio of popular music," in *ISMIR* (Suzhou), 568–574.
- Ariza, C. (2009). The interrogator as critic: the turing test and the evaluation of generative music systems. *Comput. Music J.* 33, 48–70. doi: 10.1162/comj.2009.33.2.48
- Baer, J., and McKool, S. S. (2009). "Assessing creativity using the consensual assessment technique," in *Handbook of Research on Assessment Technologies, Methods, and Applications in Higher Education* (Hershey, PA: IGI Global), 65–77.
- Bell, C. (2011). Algorithmic music composition using dynamic Markov chains and genetic algorithms. *J. Comput. Sci. Coll.* 27, 99–107.
- Bidlack, R. (1992). Chaotic systems as simple (but complex) compositional algorithms. *Compu. Music J.* 16, 33–47. doi: 10.2307/3680849
- Biles, J., Anderson, P., and Loggi, L. (1996). "Neural network fitness functions for a musical IGA," in *Proceedings of the Soft Computing Conference* (Reading, UK), 11.
- Biles, J. A. (1994). "GenJam: a genetic algorithm for generating jazz solos," in *ICMC*, Vol. 94 (Ann Arbor, MI), 131–137.
- Biles, J. A. (2001). "Autonomous GenJam: eliminating the fitness bottleneck by eliminating fitness," in *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program* (San Francisco, CA), 7.
- Boden, M. A. (1998). Creativity and artificial intelligence. *Artif. Intell.* 103, 347–356. doi: 10.1016/S0004-3702(98)00055-1
- Boden, M. A. (2004). *The Creative Mind: Myths and Mechanisms*. London: Routledge.
- Boden, M. A. (2009). Computer models of creativity. *AI Mag.* 30:23. doi: 10.1609/aimag.v30i3.2254
- Bodily, P. M., and Ventura, D. (2018). "Musical metacreation: past, present, and future," in *Mume 2018* (Salamanca: University of Salamanca), 5.
- Brémaud, P. (2013). *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, Vol. 31. New York, NY: Springer Science & Business Media.
- Bringsjord, S., Bello, P., and Ferrucci, D. (2003). "Creativity, the turing test, and the (better) Lovelace test," in *The Turing Test*, ed J. H. Moor (Dordrecht: Springer), 215–239.
- Briot, J.-P., Hadjeres, G., and Pachet, F.-D. (2017). Deep learning techniques for music generation - a survey. *arXiv:1709.01620*.
- Briot, J.-P., Hadjeres, G., and Pachet, F.-D. (2020). *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Basel: Springer International Publishing.
- Briot, J.-P., and Pachet, F. (2018). Deep learning for music generation: challenges and directions. *Neural Comput. Appl.* 32, 981–993. doi: 10.1007/s00521-018-3813-6
- Brown, D. (2012). "Mezzo: an adaptive, real-time composition program for game soundtracks," in *Musical Metacreation: Papers from the 2012 AIIDE Workshop* (Palo Alto, CA: AAAI), 5.
- Brown, D. L. (2012). *Expressing narrative function in adaptive, computer-composed music*. (Ph.D. thesis), UC Santa Cruz, Santa Cruz, CA, United States.
- Brunner, G., Konrad, A., Wang, Y., and Wattenhofer, R. (2018). "MIDI-VAE: modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018* (Paris), 747–754.
- Canazza, S., De Poli, G., and Rodà, A. (2015). Caro 2.0: an interactive system for expressive music rendering. *Adv. Hum. Comput. Interact.* 2015:850474. doi: 10.1155/2015/850474
- Canazza, S., Rodà, A., Poli, G., and Vidolin, A. (2012). *Expressiveness in Music Performance: Analysis, Models, Mapping, Encoding*. Hershey, PA: IGI Global.
- Cardaci, M., Di Gesù, V., Petrou, M., and Tabacchi, M. E. (2009). A fuzzy approach to the evaluation of image complexity. *Fuzzy Sets Syst.* 160, 1474–1484. doi: 10.1016/j.fss.2008.11.017
- Carnovalini, F. (2019). "Open challenges in musical metacreation," in *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good*, GoodTechs '19 (New York, NY: ACM), 124–125.
- Carnovalini, F., and Rodà, A. (2019a). "A multilayered approach to automatic music generation and expressive performance," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)* (Milan: IEEE), 41–48.
- Carnovalini, F., and Rodà, A. (2019b). "A real-time tempo and meter tracking system for rhythmic improvisation," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound, AM'19* (New York, NY: Association for Computing Machinery), 24–31.
- Carnovalini, F., Rodà, A., and Caneva, P. (2019). "A musical serious game for social interaction through augmented rhythmic improvisation," in *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good*, GoodTechs 19 (New York, NY: Association for Computing Machinery), 130–135.
- Ceroni, M., and Prosperi, G. M. (2018). Free will, subjectivity and the physics of the nervous system. *Open J. Philos.* 8, 317–341. doi: 10.4236/ojpp.2018.83023
- Chalmers, D. J. (1995). Facing up to the problem of consciousness. *J. Conscious. Stud.* 2, 200–219.
- Chemillier, M. (2004). Toward a formal study of jazz chord sequences generated by Steedman's grammar. *Soft Comput.* 8, 617–622. doi: 10.1007/s00500-004-0386-3
- Chew, E. (2014). "The spiral array," in *Mathematical and Computational Modeling of Tonality: Theory and Applications*, International Series in Operations Research & Management Science, ed E. Chew (Boston, MA: Springer), 41–60.
- Chomsky, N. (1957). *Syntactic Structures*. The Hague: Janua Linguarum. Mouton & Co.
- Colton, S. (2008). "Creativity versus the perception of creativity in computational systems," in *AAAI Spring Symposium: Creative Intelligent Systems* Vol. 8 (Palo Alto, CA), 7.
- Colton, S., Charnley, J. W., and Pease, A. (2011). "Computational creativity theory: the FACE and IDEA descriptive models," in *ICCC* (Mexico City), 90–95.
- Colton, S., Pease, A., Corneli, J., Cook, M., and Llano, T. (2014). "Assessing progress in building autonomously creative systems," in *ICCC* (Ljubljana), 137–145.
- Colton, S., Pease, A., and Ritchie, G. (2001). "The effect of input knowledge on creativity," in *Proceedings of the ICCBR'01 Workshop on Creative Systems* (Vancouver, BC), 7.
- Colton, S., and Wiggins, G. A. (2012). "Computational creativity: the final frontier?," in *ECAI*, Vol. 2012 (Montpellier: University of Montpellier), 21–16.
- Constantin, M. G., Redi, M., Zen, G., and Ionescu, B. (2019). Computational understanding of visual interestingness beyond semantics: literature survey and analysis of covariates. *ACM Computing Surveys (CSUR)* 52, 25:1–25:37.
- Conway, J. (1970). The game of life. *Sci. Am.* 223:4.
- Cope, D. (1991). Recombinant music: using the computer to explore musical style. *Computer* 24, 22–28. doi: 10.1109/2.84830
- Cope, D. (1992). Computer modeling of musical intelligence in EMI. *Comput. Music J.* 16:69. doi: 10.2307/3680717
- Csikszentmihalyi, M. (2013). *Creativity: The Psychology of Discovery and Invention*. New York, NY: Perennial.
- Cunha, N. d. S., Subramanian, A., and Herremans, D. (2018). Generating guitar solos by integer programming. *J. Operat. Res. Soc.* 69, 971–985. doi: 10.1080/01605682.2017.1390528
- de la Puente, A. O., Alfonso, R. S., and Moreno, M. A. (2002). "Automatic composition of music by means of grammatical evolution," in *ACM SIGAPL APL Quote Quad*, Vol. 32 (New York, NY: ACM), 148–155.
- Dieleman, S., van den Oord, A., and Simonyan, K. (2018). "The challenge of realistic music generation: modelling raw audio at scale," in *Advances in Neural Information Processing Systems 31*, eds S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Red Hook, NY: Curran Associates, Inc.), 7989–7999.
- Ebcioğlu, K. (1988). An expert system for harmonizing four-part chorales. *Comput. Music J.* 12, 43–51. doi: 10.2307/3680335
- Ebcioğlu, K. (1990). An expert system for harmonizing chorales in the style of J.S. Bach. *J. Logic Progr.* 8, 145–185. doi: 10.1016/0743-1066(90)90055-A
- Eck, D., and Schmidhuber, J. (2002a). "Finding temporal structure in music: blues improvisation with LSTM recurrent networks," in *Proceedings of the 12th IEEE*

- Workshop on Neural Networks for Signal Processing (New York, NY: IEEE), 747–756.
- Eck, D., and Schmidhuber, J. (2002b). “Learning the long-term structure of the blues,” in *International Conference on Artificial Neural Networks* (Berlin: Springer), 284–289.
- Eigenfeldt, A., Burnett, A., and Pasquier, P. (2012). “Evaluating musical metacreation in a live performance context,” in *Proceedings of the 3rd International Conference on Computational Creativity, ICC3 2012* (Dublin), 140–144.
- Eigenfeldt, A., and Pasquier, P. (2009). “A realtime generative music system using autonomous melody, harmony, and rhythm agents,” in *XIII Internationale Conference on Generative Arts* (Milan), 67–76.
- Eigenfeldt, A., and Pasquier, P. (2010). “Realtime generation of harmonic progressions using controlled markov selection,” in *Proceedings of ICC3-X-Computational Creativity Conference* (Lisbon), 16–25.
- Ekeus, H., Abdallah, S. A., Plumbley, M. D., and McOwan, P. W. (2012). “The melody triangle: exploring pattern and predictability in music,” in *Musical Metacreation: Papers from the 2012 AIIDE Workshop* (Palo Alto, CA), 8.
- Fauconnier, G., and Turner, M. (2008). *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. New York, NY: Basic Books.
- Fernández, J. D., and Vico, F. (2013). AI methods in algorithmic composition: a comprehensive survey. *J. Artif. Intell. Res.* 48, 513–582. doi: 10.1613/jair.3908
- Fraiberger, S. P., Sinatra, R., Resch, M., Riedl, C., and Barabási, A.-L. (2018). Quantifying reputation and success in art. *Science* 362, 825–829. doi: 10.1126/science.aau7224
- Galanter, P. (2012). “Computational aesthetic evaluation: past and future,” in *Computers and Creativity*, eds J. McCormack and M. d'Inverno (Berlin: Heidelberg: Springer), 255–293.
- Getzels, J. W., and Jackson, P. W. (1962). *Creativity and Intelligence: Explorations With Gifted Students*. Creativity and intelligence: Explorations with gifted students. Oxford: Wiley.
- Gillespie, S., and Bown, O. (2017). “Solving adaptive game music transitions from a composer centred perspective,” in *Proceedings of the 5th International Workshop on Musical Metacreation* (Atlanta, GA: Association for Computational Creativity), 8.
- Golann, S. E. (1963). Psychological study of creativity. *Psychol. Bull.* 60, 548–565. doi: 10.1037/h0041573
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, eds Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Red Hook, NY: Curran Associates, Inc.), 2672–2680.
- Groves, R. (2016). “Towards the generation of melodic structure,” in *The Fourth International Workshop on Musical Metacreation, MUME 2016* (Paris), 8.
- Guckelsberger, C., Salge, C., and Colton, S. (2017). “Addressing the ‘why?’ in computational creativity: a non-anthropocentric, minimal model of intentional creative agency,” in *Proceedings of the 8th International Conference on Computational Creativity* (Goldsmiths: University of London), 8.
- Guilford, J. (1967). *The Nature of Human Intelligence*. The nature of human intelligence. New York, NY: McGraw-Hill.
- Guilford, J. P. (1957). Creative abilities in the arts. *Psychol. Rev.* 64, 110–118. doi: 10.1037/h0048280
- Hadjeres, G., Pachet, F., and Nielsen, F. (2017). “Deepbach: a steerable model for bach chorales generation,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (Sydney, NSW), 1362–1371.
- Hamanaka, M., Hirata, K., and Tojo, S. (2007). “Fatta: full automatic time-span tree analyzer,” in *ICMC* (Ann Arbor, MI: Michigan Publishing), 153–156.
- Hamanaka, M., Hirata, K., and Tojo, S. (2008). “Melody morphing method based on GTTM,” in *ICMC* (Ann Arbor, MI: Michigan Publishing), 155–158.
- Hamanaka, M., Hirata, K., and Tojo, S. (2016). “Implementing methods for analysing music based on Ierdlahl and Jackendoff's generative theory of tonal music,” in *Computational Music Analysis*, ed M. David (Cham: Springer), 221–249.
- Hamanaka, M., Hirata, K., and Tojo, S. (2017). “deepgttm-III: multi-task learning with grouping and metrical structures,” in *International Symposium on Computer Music Multidisciplinary Research* (Cham: Springer), 238–251.
- Hameroff, S., and Penrose, R. (2014). Consciousness in the universe: a review of the ‘Orch OR’ theory. *Phys. Life Rev.* 11, 39–78. doi: 10.1016/j.plrev.2013.08.002
- Handelman, E., Sigler, A., and Donna, D. (2012). “Automatic orchestration for automatic composition,” in *Musical Metacreation: Papers from the 2012 AIIDE Workshop* (Palo Alto, CA: AAAI), 6.
- Herremans, D., and Chew, E. (2016a). “Morpheus: automatic music generation with recurrent pattern constraints and tension profiles,” in *Region 10 Conference (TENCON), 2016 IEEE* (New York, NY: IEEE), 282–285.
- Herremans, D., and Chew, E. (2016b). “Tension ribbons: quantifying and visualising tonal tension,” in *Proceedings of the Second International Conference on Technologies for Music Notation and Representation (TENOR)* (Cambridge), 10.
- Herremans, D., and Chew, E. (2017). Morpheus: generating structured music with constrained patterns and tension. *IEEE Trans. Affect. Comput.* 16.
- Herremans, D., Chuan, C.-H., and Chew, E. (2017). A functional taxonomy of music generation systems. *ACM Comput. Surv.* 50:69. doi: 10.1145/3108242
- Hiller, L. A. Jr., and Isaacson, L. M. (1958). Musical composition with a high-speed digital computer. *J. Audio Eng. Soc.* 6, 154–160.
- Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735
- Horner, A., and Goldberg, D. E. (1991). “Genetic algorithms and computer-assisted music composition,” in *ICMC*, Vol. 91 (Ann Arbor, MI), 479–482.
- Hsü, K. J., and Hsü, A. (1991). Self-similarity of the “1/f noise” called music. *Proc. Natl. Acad. Sci. U.S.A.* 88, 3507–3509.
- Hunter, P. G., and Schellenberg, E. G. (2010). “Music and emotion,” in *Music Perception*, eds R. J. Mari, R. R. Fay, and A. N. Popper (New York, NY: Springer), 129–164.
- Jordanous, A. (2012). A standardised procedure for evaluating creative systems: computational creativity evaluation based on what it is to be creative. *Cogn. Comput.* 4, 246–279. doi: 10.1007/s12559-012-9156-1
- Jordanous, A. (2014). “Stepping back to progress forwards: setting standards for meta-evaluation of computational creativity,” in *Proceedings of the Fifth International Conference on Computational Creativity* (Ljubljana: Jožef Stefan Institute), 8.
- Jordanous, A. (2016). Four PPPerspectives on computational creativity in theory and in practice. *Connect. Sci.* 28, 194–216. doi: 10.1080/09540091.2016.1151860
- Jordanous, A. (2019). “Evaluating evaluation: assessing progress and practices in computational creativity research,” in *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, Computational Synthesis and Creative Systems, eds T. Veale and F. A. Cardoso (Cham: Springer International Publishing), 211–236.
- Jordanous, A., and Keller, B. (2012). What makes musical improvisation creative? *J. Interdiscip. Music Stud.* 6, 151–175. doi: 10.4407/jims.2014.02.003
- Jordanous, A., and Keller, B. (2016). Modelling creativity: identifying key components through a corpus-based approach. *PLoS ONE* 11:e0162959. doi: 10.1371/journal.pone.0162959
- Jordanous, A. K. (2013). *Evaluating computational creativity: a standardised procedure for evaluating creative systems and its application*. (Ph.D. Thesis). University of Sussex, Brighton, United Kingdom.
- Juslin, P. N. (2010). *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford, UK: Oxford University Press.
- Kaliakatsos-Papakostas, M., Gkiokas, A., and Katsouros, V. (2018). “Interactive control of explicit musical features in generative LSTM-based systems,” in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion, AM'18* (New York, NY: ACM), 29:1–29:7.
- Katayose, H., Hashida, M., De Poli, G., and Hirata, K. (2012). On evaluating systems for generating expressive music performance: the rencon experience. *J. New Music Res.* 41, 299–310. doi: 10.1080/09298215.2012.745579
- Kingma, D. P., and Welling, M. (2013). Auto-encoding variational bayes. *arXiv:1312.6114*, 14.
- Kirke, A., and Miranda, E. (2015). A multi-agent emotional society whose melodies represent its emergent social hierarchy and are generated by agent communications. *J. Artif. Soc. Soc. Simulat.* 18:16. doi: 10.18564/jasss.2679
- Kirke, A., and Miranda, E. R. (2009). A survey of computer systems for expressive music performance. *ACM Comput. Surv.* 42:3. doi: 10.1145/1592451.1592454
- Kirke, A., and Miranda, E. R. (2011). “Emergent Construction of melodic pitch and hierarchy through agents communicating emotion without melodic intelligence,” in *ICMC* (Ann Arbor, MI), 8.
- Koestler, A. (1964). *The Act of Creation*. London, UK: Hutchinson & Co.

- Lamb, C., Brown, D. G., and Clarke, C. L. A. (2018). Evaluating computational creativity: an interdisciplinary tutorial. *ACM Comput. Surv.* 51, 1–34. doi: 10.1145/3167476
- Langston, P. (1989). “Six techniques for algorithmic music composition,” in *Proceedings of the International Computer Music Conference*, Vol. 60 (Ann Arbor, MI), 59.
- Leach, J., and Fitch, J. (1995). Nature, music, and algorithmic composition. *Comput. Music J.* 19, 23–33. doi: 10.2307/3680598
- Lerdahl, F., and Jackendoff, R. S. (1985). *A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press.
- Lewis, G. E. (2000). Too many notes: computers, complexity and culture in voyager. *Leonardo Music J.* 10, 33–39. doi: 10.1162/096112100570585
- Lewis, J. P. (1991). *Music and Connectionism, chap. Creation by Refinement and the Problem of Algorithmic Music Composition*. Cambridge: The MIT Press.
- Lo, M., and Lucas, S. M. (2006). “Evolving musical sequences with n-gram based trainable fitness functions,” in *2006 IEEE International Conference on Evolutionary Computation* (New York, NY: IEEE), 601–608.
- Lo, M. Y. (2012). *Evolving cellular automata for music composition with trainable fitness functions*. (Ph.D. Thesis). University of Essex, Colchester, United Kingdom.
- Lopez-Rincon, O., Starostenko, O., and Martin, G. A. (2018). “Algorithmic music composition based on artificial intelligence: a survey,” in *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)* (New York, NY), 187–193.
- Lovelace, A. (1843). Notes on L. Menabreas sketch of the analytical engine invented by Charles Babbage, Esq. *Taylor’s Sci. Mem.* 3:1843.
- Manaris, B., Hughes, D., and Vassilandonakis, Y. (2011). “Monterey mirror: combining Markov models, genetic algorithms, and power laws,” in *Proceedings of 1st Workshop in Evolutionary Music, 2011 IEEE Congress on Evolutionary Computation (CEC 2011)* (New York, NY: IEEE), 33–40.
- Marsden, A. (2010). Schenkerian analysis by computer: a proof of concept. *J. New Music Res.* 39, 269–289. doi: 10.1080/09298215.2010.503898
- Marsden, A., Hirata, K., and Tojo, S. (2013). “Towards computable procedures for deriving tree structures in music: context dependency in GTTM and Schenkerian theory,” in *Proceedings of the Sound and Music Computing Conference 2013* (Stockholm), 360–367.
- Mason, S., and Saffle, M. (1994). L-Systems, melodies and musical structure. *Leonardo Music J.* 4, 31–38. doi: 10.2307/1513178
- McAlpine, K., Miranda, E., and Hoggar, S. (1999). Making music with algorithms: a case-study system. *Comput. Music J.* 23, 19–30. doi: 10.1162/014892699559733
- McVicar, M., Fukayama, S., and Goto, M. (2014). “AutoLeadGuitar: automatic generation of guitar solo phrases in the tablature space,” in *2014 12th International Conference on Signal Processing (ICSP)* (New York, NY: IEEE), 599–604.
- Medeot, G., Cherla, S., Kosta, K., McVicar, M., Abdallah, S., and Selvi, M. (2018). “StructureNet: INDUCING STRUCTURE IN GENERATED MELODIES,” in *19th International Society for Music Information Retrieval Conference* (Paris), 7.
- Minsky, M. L. (1982). Why people think computers can’t. *AI Mag.* 3:3.
- Miranda, E. R. (1993). Cellular automata music: an interdisciplinary project. *J. New Music Res.* 22, 3–21. doi: 10.1080/09298219308570616
- Miranda, E. R. (2007). “Cellular automata music: from sound synthesis to musical forms,” in *Evolutionary Computer Music*, eds E. R. Miranda and J. A. Biles (London, UK: Springer), 170–193.
- Mogren, O. (2016). C-RNN-GAN: continuous recurrent neural networks with adversarial training. *arXiv: 1611.09904*.
- Nakamura, E., and Yoshii, K. (2018). Statistical piano reduction controlling performance difficulty. *APSIPA Trans. Sig. Informat. Process.* 7:e13. doi: 10.1017/ATSIP.2018.18
- Navarro, M., Corchado, J., and Demazeau, Y. (2014). “A musical composition application based on a multiagent system to assist novel composers,” in *5th International Conference on Computational Creativity* (Ljubljana), 4.
- Navarro, M., Corchado, J. M., and Demazeau, Y. (2016). MUSIC-MAS: modeling a harmonic composition system with virtual organizations to assist novice composers. *Exp. Syst. Appl.* 57, 345–355. doi: 10.1016/j.eswa.2016.01.058
- Nelson, G. L. (1996). Real time transformation of musical material with fractal algorithms. *Comput. Math. Appl.* 32, 109–116. doi: 10.1016/0898-1221(96)00094-6
- Nierhaus, G. (2009). *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer Science & Business Media.
- Oliveira, A. P., and Cardoso, A. (2007). “Towards affective-psychophysiological foundations for music production,” in *International Conference on Affective Computing and Intelligent Interaction* (Berlin: Springer), 511–522.
- Oore, S., Simon, I., Dieleman, S., Eck, D., and Simonyan, K. (2018). This time with feeling: learning expressive musical performance. *Neural Comput. Appl.* 32, 955–967. doi: 10.1007/s00521-018-3758-9
- Pachet, F. (2000). “Computer analysis of jazz chord sequence: is solar a blues?” in *Readings in Music and Artificial Intelligence*, ed E. R. Miranda (Reading, UK: Harwood Academic Publishers), 85–114.
- Pachet, F. (2002). “Interacting with a musical learning system: the continuator,” in *Music and Artificial Intelligence* (Berlin: Springer), 119–132.
- Pachet, F. (2012). “Musical virtuosity and creativity,” in *Computers and Creativity*, eds J. McCormack and M. d’Inverno (Berlin; Heidelberg: Springer), 115–146.
- Papadopoulos, A., Roy, P., and Pachet, F. (2014). “Avoiding plagiarism in Markov sequence generation,” in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI’14* (Quebec City, QC: AAAI Press), 2731–2737.
- Papadopoulos, G., and Wiggins, G. (1999). “AI methods for algorithmic composition: a survey, a critical view and future prospects,” in *AISB Symposium on Musical Creativity*, Vol. 124 (Edinburgh), 110–117.
- Pasquier, P., Eigenfeldt, A., Bown, O., and Dubnov, S. (2017). An introduction to musical metacreation. *Comput. Entertain.* 14, 2:1–2:14. doi: 10.1145/2930672
- Pearce, M., Meredith, D., and Wiggins, G. (2002). Motivations and methodologies for automation of the compositional process. *Musicae Scientiae* 6, 119–147. doi: 10.1177/102986490200600203
- Pearce, M., and Wiggins, G. (2001). “Towards a framework for the evaluation of machine compositions,” in *Proceedings of the AISB’01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences* (York, UK), 22–32.
- Pease, A., and Colton, S. (2011a). “Computational creativity theory: inspirations behind the FACE and the IDEA models,” in *ICCC* (Mexico City: Association for Computational Creativity), 72–77.
- Pease, A., and Colton, S. (2011b). “On impact and evaluation in computational creativity: a discussion of the Turing test and an alternative proposal,” in *Proceedings of the AISB Symposium on AI and Philosophy*, Vol. 39 (York, UK), 8.
- Pease, A., and Corneli, J. (2018). “Evaluation of creativity,” in *Concept Invention: Foundations, Implementation, Social Aspects and Applications*, Computational Synthesis and Creative Systems, eds R. Confalonieri, A. Pease, M. Schorlemmer, T. R. Besold, O. Kutz, E. Maclean, and M. Kaliakatos-Papakostas (Cham: Springer International Publishing), 277–294.
- Pease, A., Winterstein, D., and Colton, S. (2001). “Evaluating machine creativity,” in *Workshop on Creative Systems, 4th International Conference on Case Based Reasoning* (Vancouver, BC), 129–137.
- Phon-Amnuaisuk, S., Tuson, A., and Wiggins, G. (1999). “Evolving musical harmonisation,” in *Artificial Neural Nets and Genetic Algorithms*, eds A. Dobnikar, N. C. Steele, D. W. Pearson, and R. F. Albrecht (Vienna: Springer), 229–234.
- Phon-Amnuaisuk, S., and Wiggins, G. A. (1999). “The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system,” in *Proceedings of the AISB’99 Symposium on Musical Creativity* (London: AISB), 28–34.
- Pinkerton, R. C. (1956). Information theory and melody. *Sci. Am.* 194, 77–87. doi: 10.1038/scientificamerican0256-77
- Prusinkiewicz, P. (1986). “Score generation with L-systems,” in *ICMC* (Ann Arbor, MI), 455–457.
- Quick, D. (2011). *Generating Music Using Concepts from Schenkerian Analysis and Chord Spaces*. Technical report, Yale University.
- Ranjan, B. S. C., Siddharth, L., and Chakrabarti, A. (2018). A systematic approach to assessing novelty, requirement satisfaction, and creativity. *AI EDAM* 32, 390–414. doi: 10.1017/S0890060418000148
- Rhodes, M. (1961). An analysis of creativity. *The Phi Delta Kappan* 42, 305–310.
- Riedl, M. O. (2014). The lovelace 2.0 test of artificial creativity and intelligence. *arXiv: 1410.6142*.
- Ritchie, G. (2001). “Assessing creativity,” in *Proceedings of AISB’01 Symposium* (York, UK: University of Edinburgh, Department of Artificial Intelligence), 10.
- Ritchie, G. (2007). Some empirical criteria for attributing creativity to a computer program. *Minds Mach.* 17, 67–99. doi: 10.1007/s11023-007-9066-2
- Ritchie, G. (2019). “The evaluation of creative systems,” in *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, Computational Synthesis and Creative Systems, eds T. Veale and F. A. Cardoso (Cham: Springer International Publishing), 159–194.

- Sabastien, V., Ralambondrainy, H., Sabastien, O., and Conruyt, N. (2012). "Score analyzer: automatically determining scores difficulty level for instrumental e-learning," in *13th International Society for Music Information Retrieval Conference (ISMIR 2012)* (Porto), 571–576.
- Sarkar, P., and Chakrabarti, A. (2008). "Studying engineering design creativity: developing a common definition and associated measures," in *Proceedings of the NSF Workshop on Studying Design Creativity* (Aix-en-Provence), 20.
- Sarkar, P., and Chakrabarti, A. (2011). Assessing design creativity. *Design Stud.* 32, 348–383. doi: 10.1016/j.destud.2011.01.002
- Saunders, R. (2019). "Multi-agent-based models of social creativity," in *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, Computational Synthesis and Creative Systems, eds T. Veale and F. A. Cardoso (Cham: Springer International Publishing), 305–326.
- Schenker, H. (1979). *Free Composition (Der freie Satz)*. Longman Music Series. New York, NY: Longman.
- Schmidhuber, J. (2012). "A formal theory of creativity to model the creation of art," in *Computers and Creativity*, eds J. McCormack and M. d'Inverno (Berlin; Heidelberg: Springer), 323–337.
- Schubert, E., Canazza, S., De Poli, G., and Rodà, A. (2017). Algorithms can mimic human piano performance: the deep blues of music. *J. New Music Res.* 46, 175–186. doi: 10.1080/09298215.2016.1264976
- Scirea, M., Eklund, P., Togelius, J., and Risi, S. (2018). "Evolving in-game mood-expressive music with metaCompose," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion, AM'18* (New York, NY: ACM), 8:1–8:8.
- Searle, J. R. (1980). Minds, brains, and programs. *Behav. Brain Sci.* 3, 417–424. doi: 10.1017/S0140525X00005756
- Shaker, N., Smith, G., and Yannakakis, G. N. (2016). "Evaluating content generators," in *Procedural Content Generation in Games*, Computational Synthesis and Creative Systems, eds N. Shaker, J. Togelius, and M. J. Nelson (Cham: Springer International Publishing), 215–224.
- Simonetta, F., Carnovalini, F., Orio, N., and Rodà, A. (2018). "Symbolic music similarity through a graph-based representation," in *Audio Mostly 2018: Sound in Immersion and Emotion (AM'18)*, September 12–14, 2018, Wrexham, United Kingdom (New York, NY: ACM), 7.
- Simonton, D. K. (2000). Creativity: cognitive, personal, developmental, and social aspects. *Am. Psychol.* 55:151. doi: 10.1037/0003-066X.55.1.151
- Sivanandam, S. N., and Deepa, S. N. (2008). "Genetic algorithms," in *Introduction to Genetic Algorithms* (Berlin: Springer), 15–37.
- Soldier, D. (2002). Eine Kleine Naughtmusik: how nefarious Nonartists cleverly imitate music. *Leonardo Music J.* 12, 53–58. doi: 10.1162/096112102762295142
- Steedman, M. J. (1984). A generative grammar for jazz chord sequences. *Music Percept.* 2, 52–77. doi: 10.2307/40285282
- Stein, M. I. (1953). Creativity and culture. *J. Psychol.* 36, 311–322. doi: 10.1080/00223980.1953.9712897
- Sturm, B. L. (2018). "What do these 5,599,881 parameters mean? An analysis of a specific LSTM music transcription model, starting with the 70,281 parameters of its softmax layer," in *Proceedings of the 6th International Workshop on Musical Metacreation* (Salamanca), 8.
- Sturm, B. L., Ben-Tal, O., Monaghan, V., Collins, N., Herremans, D., Chew, E., et al. (2019). Machine learning research that matters for music creation: a case study. *J. New Music Res.* 48, 36–55. doi: 10.1080/09298215.2018.1515233
- Sturm, B. L., Santos, J. F., Ben-Tal, O., and Korshunova, I. (2016). Music transcription modelling and composition using deep learning. *arXiv: 1604.08723*.
- Supper, M. (2001). A few remarks on algorithmic composition. *Comput. Music J.* 25, 48–53. doi: 10.1162/014892601300126106
- Tabacchi, M., and Termini, S. (2011). "Measures of fuzziness and information: some challenges from reflections on aesthetic experience," in *World Conference on Soft Computing* (San Francisco, CA), 8.
- Tatar, K., and Pasquier, P. (2019). Musical agents: a typology and state of the art towards Musical Metacreation. *J. New Music Res.* 48, 56–105. doi: 10.1080/09298215.2018.1511736
- Todd, P. M. (1989). A connectionist approach to algorithmic composition. *Comput. Music J.* 13, 27–43. doi: 10.2307/3679551
- Torrance, E. P. (1965). Scientific views of creativity and factors affecting its growth. *Daedalus* 94, 663–681.
- Torrance, E. P. (1988). "The nature of creativity as manifest in its testing," in *The Nature of Creativity: Contemporary Psychological Perspectives*, ed R. J. Sternberg (Cambridge, UK: Cambridge University Press), 43–75.
- Treadwell, Y. (1970). Humor and creativity. *Psychol. Rep.* 26, 55–58. doi: 10.2466/pr0.1970.26.1.55
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind* 59, 433–460. doi: 10.1093/mind/LIX.236.433
- Ventura, F., Oliveira, A., and Cardoso, A. (2009). "An emotion-driven interactive system," in *Portuguese Conference on Artificial Intelligence (Aveiro)*, 12.
- Voss, R. F., and Clarke, J. (1978). 1/f noise in music: music from 1/f noise. *J. Acoust. Soc. Am.* 63, 258–263. doi: 10.1121/1.381721
- Werner, G. M., and Todd, P. M. (1997). "Too many love songs: sexual selection and the evolution of communication," in *Fourth European Conference on Artificial Life* (Cambridge, MA: MIT Press, Bradford Books), 434–443.
- Widmer, G., and Goebel, W. (2004). Computational models of expressive music performance: the state of the art. *J. New Music Res.* 33, 203–216. doi: 10.1080/0929821042000317804
- Wiggins, G. A. (2006). A preliminary framework for description, analysis and comparison of creative systems. *Knowl. Based Syst.* 19, 449–458. doi: 10.1016/j.knosys.2006.04.009
- Wiggins, G. A. (2007). Computer models of musical creativity: a review of computer models of musical creativity by David cope. *Literary Linguist. Comput.* 23, 109–116. doi: 10.1093/llc/fqm025
- Wiggins, G. A. (2018). Creativity, information, and consciousness: the information dynamics of thinking. *Phys. Life Rev.* doi: 10.1016/j.plrev.2018.05.001. [Epub ahead of print].
- Wiggins, G. A. (2019). "A framework for description, analysis and comparison of creative systems," in *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, Computational Synthesis and Creative Systems, eds T. Veale and F. A. Cardoso (Cham: Springer International Publishing), 21–47.
- Wiggins, G. A., and Forth, J. (2015). "IDyOT: a computational theory of creativity as everyday reasoning from learned information," in *Computational Creativity Research: Towards Creative Machines*, Atlantis Thinking Machines, eds T. R. Besold, M. Schorlemmer, and A. Smaill (Paris: Atlantis Press), 127–148.
- Wiggins, G. A., and Sanjekdar, A. (2019). Learning and consolidation as re-representation: revising the meaning of memory. *Front. Psychol.* 10:802. doi: 10.3389/fpsyg.2019.00802
- Williams, D., Kirke, A., Miranda, E. R., Roesch, E., Daly, I., and Nasuto, S. (2015). Investigating affect in algorithmic composition systems. *Psychol. Music* 43, 831–854. doi: 10.1177/0305735614543282
- Williams, D., Kirke, A., Miranda, E. R., Roesch, E. B., and Nasuto, S. J. (2013). "Towards affective algorithmic composition," in *The 3rd International Conference on Music & Emotion, Jyväskylä, Finland, June 11–15, 2013* (Jyväskylä: University of Jyväskylä, Department of Music), 8.
- Wolfram, S. (2002). *A New Kind of Science*, Vol. 5. Champaign, IL: Wolfram Media.
- Xambó, A., Pauwels, J., Roma, G., Barthet, M., and Fazekas, G. (2018). "Jam with jamendo: querying a large music collection by chords from a learner's perspective," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion, AM'18* (New York, NY: ACM), 30:1–30:7.
- Yang, L.-C., Chou, S.-Y., and Yang, Y.-H. (2017). MidiNet: a convolutional generative adversarial network for symbolic-domain music generation. *arXiv: 1703.10847*.
- Yang, L.-C., and Lerch, A. (2018). On the evaluation of generative models in music. *Neural Comput. Appl.* doi: 10.1007/s00521-018-3849-7. [Epub ahead of print].

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Carnovalini and Rodà. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Designing and Evaluating the Usability of a Machine Learning API for Rapid Prototyping Music Technology

Francisco Bernardo^{1,2*}, Michael Zbyszyński², Mick Grierson^{2,3} and Rebecca Fiebrink^{2,3}

¹ EMuTe Lab, School of Media, Film and Music, University of Sussex, Brighton, United Kingdom, ² EAVI, Department of Computing, Goldsmiths, University of London, London, United Kingdom, ³ Creative Computing Institute, University of the Arts, London, United Kingdom

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Deepak P.,
Queen's University Belfast,
United Kingdom
Chetan Tonde,
Amazon, United States

*Correspondence:

Francisco Bernardo
f.bernardo@sussex.ac.uk

Specialty section:

This article was submitted to
Machine Learning and Artificial
Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 30 October 2019

Accepted: 09 March 2020

Published: 03 April 2020

Citation:

Bernardo F, Zbyszyński M, Grierson M and Fiebrink R (2020) Designing and Evaluating the Usability of a Machine Learning API for Rapid Prototyping Music Technology.
Front. Artif. Intell. 3:13.
doi: 10.3389/frai.2020.00013

To better support creative software developers and music technologists' needs, and to empower them as machine learning users and innovators, the usability of and developer experience with machine learning tools must be considered and better understood. We review background research on the design and evaluation of application programming interfaces (APIs), with a focus on the domain of machine learning for music technology software development. We present the design rationale for the RAPID-MIX API, an easy-to-use API for rapid prototyping with interactive machine learning, and a usability evaluation study with software developers of music technology. A cognitive dimensions questionnaire was designed and delivered to a group of 12 participants who used the RAPID-MIX API in their software projects, including people who developed systems for personal use and professionals developing software products for music and creative technology companies. The results from questionnaire indicate that participants found the RAPID-MIX API a machine learning API which is easy to learn and use, fun, and good for rapid prototyping with interactive machine learning. Based on these findings, we present an analysis and characterization of the RAPID-MIX API based on the cognitive dimensions framework, and discuss its design trade-offs and usability issues. We use these insights and our design experience to provide design recommendations for ML APIs for rapid prototyping of music technology. We conclude with a summary of the main insights, a discussion of the merits and challenges of the application of the CDs framework to the evaluation of machine learning APIs, and directions to future work which our research deems valuable.

Keywords: application programming interfaces, cognitive dimensions, music technology, interactive machine learning, user-centered design

1. INTRODUCTION

Research on the design of music systems with artificial intelligence techniques goes back more than 30 years (Dannenberg, 1985). Much of this work has been motivated by the exploration and discovery of new sounds, music, and new forms of musicianship and performance (Miranda and Wanderley., 2008). Within this domain, research focused on the design of mapping strategies with

interactive machine learning (IML)—i.e., using supervised machine learning (ML) for mapping between different kinds of inputs (e.g., sensor data, motion descriptors, audio features) and parameters of sound and music processes (e.g., Fiebrink et al., 2011; Françoise et al., 2013; Caramiaux et al., 2014)—has uncovered very meaningful advantages. They include, for instance, workflows with increased celerity and ease-of-use, intuitive exploration of complex mappings and high-dimensional parameter spaces, and increased utility of small training data sets. These findings are promising not only on their own, but also when considering the opportunities to broaden and accelerate innovation in music technology with ML (Bernardo et al., 2017). However, in order to facilitate the adoption of ML by music and creative software developers, the usability and the developer experience with new tools for designing, developing and using ML, must be considered and better understood.

IML approaches to building ML systems involve rapid cycles of human actions modifying an ML model and interactive examination of the outcomes of those modifications (Fails and Olsen, 2003). Unlike algorithm-driven approaches such as active learning, IML approaches entail human-driven cycles of model creation, change, and evaluation. As Amershi et al. (2014) write, IML enables “even users with little or no machine-learning expertise [to] steer machine-learning behaviors through low-cost trial and error or focused experimentation with inputs and outputs” (p. 106). IML approaches often provide ways for users to incorporate information about their goals or domain knowledge into the creation of an ML system, for instance by creating or curating training datasets that encode their understanding of the target behavior to be learned by an ML algorithm.

IML can be useful for ML problems in which the user’s goal is to encode a human-understandable behavior into the system, and the user is capable of iteratively creating or curating training data to steer the model toward achieving the desired behavior. This is the case for many problems in music technology involving the analysis of audio, visual, and sensor data, in which a human user is capable of providing a supervised learning algorithm with examples of data paired with the desired labels. A musician could, for instance, pair examples of music clips, or segments of gestural data, with classifier labels indicating mood or instrumentation. Musicians and creative technologists have frequently used IML to support the creation of new sensor-based systems for embodied interaction and musical expression, such as the design of new musical instruments and new creative physical computing systems (Hartmann et al., 2007; Fiebrink et al., 2011; Katan et al., 2015).

In this paper, we present the design rationale and a usability evaluation of the RAPID-MIX API, a toolkit and application programming interface (API) for rapid prototyping music technology with interactive machine learning (IML). Our main objective is to explore how the design decisions and trade-offs of an API for rapid prototyping with IML affect its usability and the developer experience. We also identify specific design features in other ML APIs and toolkits and provide recommendations which our research and design experience suggests can be applied to other work. This work contributes a deeper understanding of ML API usability and its impact on the experience of music

technologists and creative developers who are not ML experts and lack ML background. This work thus informs research and practice in the domains of API usability, human-centered ML, and music technology, where (to our knowledge) there is little research about human-centered design and evaluation of ML APIs.

The paper is structured as follows. This section introduces readers to concepts with background material on the design and evaluation of APIs, and contextualizes our work with information about other ML API and toolkit designs used in music technology. Section 2 describes the RAPID-MIX API as the main material and its underlying design assumptions. Section 3 describes the study with an adapted Cognitive Dimensions questionnaire. Section 4 presents a qualitative analysis of the results of the CDs questionnaire. In section 5, we discuss the main insights about the RAPID-MIX API design trade-offs and usability issues identified by the study. We also provide a set of recommendations for the design of ML APIs for prototyping music technology. We conclude in section 6 with a summary of the main insights and future work.

1.1. Design and Evaluation of APIs

Software developers integrating machine learning (ML) into their applications are likely to resort to third-party infrastructural software—that is, software that supports the development and operation of other software (e.g., middleware, software libraries and frameworks, online services, toolkits) (Edwards et al., 2003). The use of APIs—the developer-facing constituents of infrastructural software—is a standard and important practice in software engineering that prevails modularity and reusability (Fowler, 2004). Developers use API calls within their application code to extend their applications’ capabilities with infrastructural software functionality.

APIs can provide potential savings in time and effort for common development tasks. However, developers making an informed decision about adopting an API may have to consider their previous experience with a specific API and API domain, as well as with the API conceptual model and the available design cues and patterns (Blackwell, 2002). Accessing the efficiency gains that APIs provide in relation to the cost of programming a custom solution is not straightforward though. Furthermore, the structure of the API and its documentation may have a significant impact on the API learning experience, given the ingrained assumptions about prior conceptual knowledge, target application scenarios, code examples, and learning resources provided (Robillard and Deline, 2011). When designing an ML API, the lack of consideration for these aspects can lead to a challenging and overwhelming learning experience.

Designing an API is a challenging task, let alone an ML API. An API must meet users’ technical requirements—e.g., performance, robustness, correctness, stability, security (Henning, 2009). An API must be usable (Myers and Stylos, 2016) and provide effective learning (Robillard and Deline, 2011). An API must also be useful and provide an appropriate set of features for a space of potential client applications (Edwards et al., 2003). An ML API should provide the ability to train and evaluate existing ML algorithms on new data. A ML API can also

incorporate a certain degree of domain expertise; for instance, by making available complete ML pipelines with predefined choices of algorithms and parameters (Mellis et al., 2017), pre-trained models for transfer learning (Jialin and Yang, 2010), or other functionalities that are likely to be useful for supporting application development in particular domains.

There exist different approaches to API design and evaluation. A designer-centric approach to API design is mostly based on the designer's taste or aesthetics¹. This can be successful when the designer has an extensive experience in both API design and in the API application domain. Approaches based on API design heuristics use empirically-based knowledge that has been compiled into prescriptive guidelines or recommendations (e.g., Tulach, 2008; Cwalina and Abrams, 2009). There is, however, contradicting empirical evidence about the usability of certain API design heuristics (Myers and Stylos, 2016). User-centered design (UCD) approaches inform and drive API design with usability data (e.g., Clarke, 2010; Myers and Stylos, 2016). This can be useful to counteract misleading assumptions about API users who are not represented within the API designers' group. Nevertheless, usability approaches excessively focused on specific design features might fail to deliver in a more holistic way¹.

Myers and Stylos (2016) provide a comprehensive review of different methods to measure and improve the design of APIs. One traditional approach is API peer reviews (Wiegers, 2002), where technical peers examine and give feedback. An alternative is the API Concepts framework (Scheller and Kühn, 2015), which automatically evaluates both the API and samples of client code, considering user characteristics (e.g., learning style, programming experience) among the critical factors of evaluation. Other methods have been adapted from traditional HCI and usability engineering, including empirical and task-specific evaluation techniques (e.g., think-aloud protocol, cognitive walkthrough) as well as heuristics-based techniques (Myers and Stylos, 2016).

Other approaches to API evaluation are based on the Cognitive Dimensions (CDs) of Notations framework (Green, 1989; Green and Petre, 1996). CDs are a "broad-brush" set of evaluation tools that support discussion about the design trade-offs of notations and information structures. The CDs have been previously applied to the analysis and assessment of different types of music technology, including a music typesetting package (Blackwell and Green, 2000), music notation systems (Blackwell et al., 2000), sequencing interfaces (Nash, 2014), algorithmic composition software (Bellingham et al., 2014). API studies based on CDs typically either used the questionnaire originally developed by Blackwell and Green (2000), or a shorter or partially refactored version, specialized to a specific domain. For instance, the original CDs questionnaire Blackwell and Green (2000) was used by Austin (2005), who assessed the usability of a functional shading language for graphics programming. Diprose et al. (2017) used it to assess the abstraction level of an end-user robot programming API. Clarke and Becker (2003) derived a framework from the original CDs to characterize

specifically how API design trade-offs met the expectations of the API users, and applied it to evaluate Microsoft .NET class libraries. Watson (2014) applied Clarke's framework for improving API documentation planning. Wijayarathna et al. (2017) adapted Clarke's questionnaire for evaluating the usability of a cryptography API.

There is little research focusing on the human-centered design and evaluation of ML APIs. To our knowledge, there is no prior research which applies the Cognitive Dimensions framework in the evaluation of the usability of an ML API.

1.2. Machine Learning APIs and Toolkits for Music Technology

Developers are users of ML when they configure learning algorithms, and when they train, evaluate, and export models, or import the resulting pre-trained models into their music technology applications. When building IML or other "intelligent" systems for personal use in music performance and composition—i.e., end-user development (Lieberman et al., 2006)—or for others to use, in commercial applications of music technology, developers can employ custom-built learning algorithms. However, many developers will use general-purpose ML infrastructural software via API calls to build their applications, regardless of the specific end-user goal or end-user application usage.

Over the years, a number of general-purpose ML tools have been developed, including R packages such as Caret (Kuhn, 2008), graphical user interfaces (GUIs) such as Weka (Hall et al., 2009), and APIs such as scikit-learn (Buitinck et al., 2013). With the recent breakthroughs in deep learning, we observe an intensive push of ML development toolkits and APIs into the hands of developers—e.g., Google Tensorflow (Abadi et al., 2016), Apple CoreML² and TuriCreate³, Pytorch⁴. While most of these APIs target ML experts, some of them cater to an audience of ML non-expert users. However, many of these APIs still remain difficult to use.

Other initiatives push for the democratization of ML using a domain-specific approach, i.e., within certain domains of application and more creative endeavors, which include the generation and control of media, such as image, video, and music. MnM (Bevilacqua et al., 2005) is a toolkit which allows users to create custom gesture-to-sound mappings using statistical methods such as principal components analysis⁵, hidden Markov models⁶ and other algorithms. This toolkit is implemented as a suite of externals for Max⁷, which is used extensively in the context of experimental music technology. These externals (i.e.,

¹Venners, B. and Eckel, B., The C# Design Process: A Conversation with Anders Hejlsberg – Part I, <https://www.artima.com/intv/csdes2.html>, (accessed September 15, 2019).

²Core ML, <https://developer.apple.com/documentation/>, (accessed September 15, 2019).

³Turi Create, <https://github.com/apple/turicreate>, (accessed September 15, 2019).

⁴Pytorch, <https://pytorch.org/>, (accessed September 15, 2019).

⁵Principal component analysis (PCA) is a statistical method which converts observations of potentially correlated variables into a set of linearly uncorrelated variables (the principal components). PCA is often applied for dimensionality reduction high-dimensional data sets.

⁶A hidden Markov model is a Markov chain for which the state is only partially observable. A Markov chain is a method for modeling complex systems using random processes and probability, sequences of possible events and interdependent states.

⁷Max, <https://cycling74.com/products/max/>

processing components that are used within Max's graphical patching environment) enable users to program ML pipelines in the data-flow paradigm.

Fiebrink et al. (2011) used the Weka API in the development of Wekinator, a general-purpose standalone application for applying supervised machine learning. The Weka API is an object-oriented API, written in Java which provides standard implementations of learning algorithms. Wekinator provides a high-level interface to a workflow which that enables users to rapidly create and edit datasets, and to employ these algorithms (and others such as SVM, Dynamic Time Warping) to train and run ML models in real time. It also supports the configuration and mapping of sensor data to end-user musical software, using high-level application pipelines connected through the OSC communication protocol. This end-user programming (Lieberman et al., 2006) approach to IML has been employed in the exploration of user interactions with machine learning in the context of music composition and performance.

The Gesture Recognition Toolkit (GRT) (Gillian and Paradiso, 2014) is an OSS, cross-platform C++ library aimed to make real-time machine learning and gesture recognition more accessible for non-specialists. GRT adopted core design principles which include:

- Simplicity and accessibility, provided by a minimal code footprint and consistent coding conventions.
- Flexibility and customizability, supported by modular architecture structured around the metaphor of a real-time multimodal data pipeline.
- A supporting infrastructure offering a wide range of algorithms and functions for pre- and post-processing, feature extraction and data set management.

Although GRT provided benefits and advantages over more typical ML development environments (e.g., Matlab) it remained difficult to utilize by people who had not the C++ and software engineering skills for the lower-level parts of the code. Nevertheless, it paved the way for other approaches to ease user adoption. For instance, *ml.lib* (Bullock and Momeni, 2015) is an OSS machine learning toolkit designed for two domain-specific data flow programming environments, Max and Pure Data⁸. *ml.lib* was implemented as a set of modules that wrap up GRT library components (Gillian and Paradiso, 2014) and execute within these environments as external components. Besides GRT's core principles which *ml.lib* builds upon (Bullock and Momeni, 2015), other aspects of its design rationale include:

- enabling users without ML background to experiment with and integrate a wide range of ML techniques into their projects.
- taking advantage of the affordances of data-flow programming environments, including (a) rapid prototyping and (b) multimedia integration, (c) high-level abstraction which hides away threading and memory management, and (d) integrated documentation with interactive examples.
- maximizing learnability and discoverability through "a simple, logical and consistent, scalable interface."

- providing portability and maintainability through the use of a cross-platform and multi-target technology stack that supports different desktop operating systems and embedded hardware architectures and processors.

Another ML toolkit which builds upon GRT and takes another approach to bridge the gap for ML-non-expert developers is ESP (Mellis et al., 2017). The ESP approach intensifies the domain-specific and adoption orientation through the provision of augmented code examples of end-to-end ML pipelines (e.g., audio beat detection). These examples are written by experts using the GRT library and the OpenFrameworks creative coding framework. This approach makes a few assumptions such as the existence of a community of vested experts willing to contribute their ML design expertise to the creation of augmented code examples. Another assumption concerns the tight coupling of the augmented code examples with high-level GUIs, which is deemed fundamental to the learning of the machine learning workflows.

Other ML toolkits have been designed with usability as a primary concern. For instance, Keras is an open-source deep learning API which, according to the author F. Chollet⁹, was designed for usability and with usability principles in mind—consistent and simple APIs, end-to-end pipelines with minimal number of user actions required for common use cases, and clear and actionable feedback upon user error. The usability-focused innovation in ML API design of Keras led to its recent adoption as one of the main interfaces of Tensorflow ecosystem (Abadi et al., 2016). The Layers API from Tensorflow.js (Smilkov et al., 2019) is modeled after Keras, also building upon the advantages of Javascript (JS)—e.g., WebGL-accelerated end-to-end ML pipelines supporting both training and inference in the browser; predefined layers with reasonable defaults; ease of distribution and deployment; portability, server-side and client-side execution in the browser; the wide adoption and relatively low-entry barrier of the JS programming language for novice programmers.

As part of the Tensorflow ecosystem, Magenta.js (Roberts et al., 2018) is an API for pre-trained music generation models. This library was also positioned to bridge the ML-non-expert developers gap through the provision of an even higher abstraction level. One important design assumption is its abstraction-level; hiding away "unnecessary complexities from developers [...] would remove the need for machine learning expertise" (p. 1). Magenta.js employs a transfer learning approach (Jialin and Yang, 2010)—i.e., enables the integration of pre-trained models to trivialize end-user adoption—with model weights, parameters, and description made accessible from a URL (remote js-checkpoints). Magenta provides music-specific data structures such as NoteSequences—an abstract time representation of a series of notes, characterized by attributes pitch, instrument and strike velocity (akin to MIDI). It also provides API objects which wrap up deep learning models for musical application—e.g., music variational auto encoder MusicVAE, MusicRNN, Music Transformer, etc. These are at the core of a growing list of examples—both developed in-house

⁸Pure Data, <https://puredata.info/>

⁹<https://blog.keras.io/user-experience-design-for-apis.html>

and by the community—of the application of the library to demonstrate cutting-edge deep learning techniques for music generation through a carefully crafted set of interactive example applications with publicly available code.

ml5.js is another example of an ML API which also builds on *Tensorflow.js*. However, *ml5.js* provides an even higher-level of abstraction to empower artists and creative coders. According to Daniel Shiffman¹⁰, the two main barriers to the adoption of ML which *ml5.js* aims to overcome, are “having to install and configure a development environment and secondly, having to implement low-level mathematical operations and technical code.” *ml5.js* aims to support an accelerated learning experience by providing an integrated set of online resources—e.g., the API documentation linking collections of code examples with relevant applications in the P5.js online code editor, video tutorials. There are introduction to complementary technologies (e.g., javascript, the library P5.js, websockets) and to more challenging programming concepts such as asynchronous operations and callback functions. The code examples which enable training in the browser employ the *Tensorflow Visor*, an interactive visualization utility that provides feedback on the neural network training and loss function minimization.

There are common traits to these ML APIs and toolkits which have been created using a domain-specific approach to music technology. They share usability principles and accessibility concerns reflected in the ease of deployment, higher-level of abstraction, constraints to the ML pipeline building. They also share code examples grounded on musical or sound applications, which also show interactivity playing a fundamental role in improving the accessibility and understandability of the ML API for ML -non-expert developers. For instance, in the case of *ml.lib*, this happens through the affordances of the interactive data-flow environments, which are effective at conveying information structures with a pipeline metaphor (Green and Petre, 1996). In ESP, the GUIs are not just API client code; rather, they play an essential role in the illustration of the main functional code blocks and in enabling the training and inference workflows supported by an augmented code example ML pipeline. In *ml5.js*, ML model pipelines are built according pre-defined configurations based on the specific task (e.g., classification or regression). In *Magenta*, code examples feature pre-trained models such as recurrent neural networks for melody generation and musical accompaniment, automatic music generation from performance MIDI datasets, and for interpolation between melodic lines, drum sequences, and music styles.

ML APIs can influence the features and interaction style of the resulting applications. They can also impact the developers' working processes and experience with ML. However, the challenges and experience of developers working with ML APIs remain under-explored, particularly for the development of IML systems for creative and musical technology.

2. THE RAPID-MIX API

The RAPID-MIX API is a toolkit comprising ML libraries and learning resources (Figure 1) that were designed and implemented in the context of RAPID-MIX¹¹, an EU innovation project focused on the creative industries. The RAPID-MIX project stakeholders identified a variety of potential scenarios for an API that would support rapid prototyping with interactive machine learning for creative and music applications. The API was intended to support both product development by small and medium companies (SMEs)—including music technology companies, e.g., ROLI¹², AudioGaming¹³, and Reactable Systems¹⁴—as well as by individual developers working in creative and musical technology. Domains of use included education, games development, music technology, e-health, and sports. Use cases included potential creative products from the previous domains where sensor-based interaction, expressive multimodal control, mapping and rich audiovisual output could benefit from a flexible rapid-prototyping API. Target environments could be desktop, mobile or web apps, or embedded processors.

2.1. User-Centered Infrastructural Software

The design of the RAPID-MIX API followed a user-centric approach, where different stakeholders were engaged early and throughout the process, including academics, end-user developers—people developing systems for personal use (Lieberman et al., 2006)—and professional developers working in creative and music technology companies. The design of the API targeted students, “hackers,” and “makers” who might wish to develop other new technologies using ML. The RAPID-MIX API aimed to explicitly support IML approaches to systems development, in which developers can iteratively create, curate, and modify supervised ML training sets in order to influence model behavior.

Design iterations were informed by lightweight formative evaluation actions (Bernardo et al., 2018) using techniques such as direct observation, interviews and group discussions in workshops and hackathons, and remote Q&A sessions between API designers and users. This work contributed to a better understanding of the needs, goals and values of the target users of the RAPID-MIX API, which spanned a breadth of software development skills, experience, motivation, and technical approach expected from creative and music technology developers. Most notably, target RAPID-MIX API users had little to no prior ML expertise, which strongly informed the design considerations and trade-offs.

Early uses of the RAPID-MIX API by creative developers included the integration of the IML workflow in ultra-low-latency audio applications in embedded systems, driving visual parameters of video jockey apps with audio and multimodal

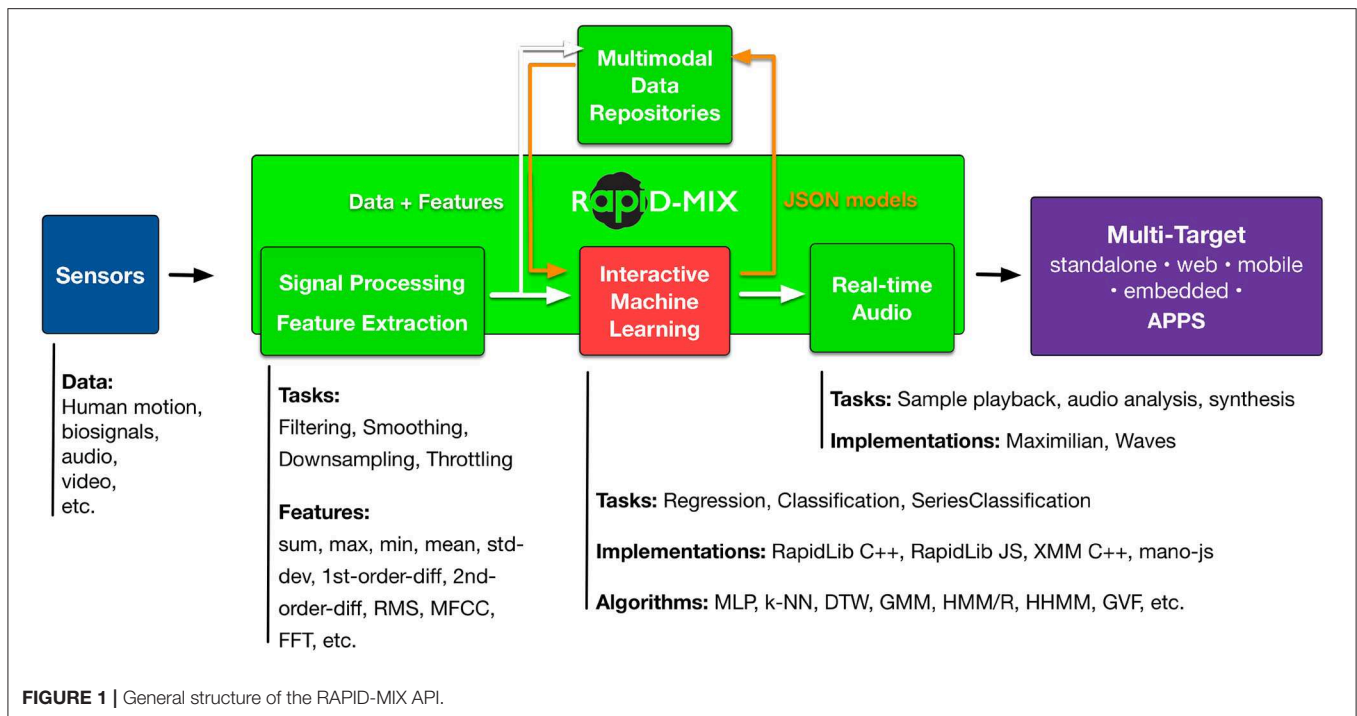
¹⁰<https://itp.nyu.edu/adjacent/issue-3/ml5-friendly-open-source-machine-learning-library-for-the-web/>

¹¹RAPID-MIX: Realtime Adaptive Prototyping for Industrial Design of Multimodal Interactive eXpressive technology, <http://rapidmix.goldsmithsdigital.com/>

¹²ROLI, <https://roli.com>

¹³AudioGaming, <http://www.audiogaming.net/>

¹⁴Reactable Systems, <https://reactable.com>



feature analysis, and browser-based audio synthesizers and sequencers using real-time sensor data (e.g., Leap Motion¹⁵ hand pose data, Myo¹⁶ electromyography and inertial measurement data, BITalino¹⁷ data), applications for custom control of 3D mesh animation, the rock-paper-scissors game, etc. Additional illustrative use cases have been developed by participants of this study (section 4.1) which include the creation of commercial musical software products.

2.2. API Design Decisions and Architecture

The RAPID-MIX API aims to facilitate rapid prototyping by developers in ways that are similar to how Wekinator (Fiebrink et al., 2011)—a popular GUI-based tool for creative IML—supports its users. For instance, it aims to minimize the number of actions a user needs to take to develop a working prototype (see **Listing 1**). Users need only to create an instance of an ML class, train a model, and run it on new data; no additional setup is required. Further, default values that support common use cases are provided for all configurable algorithm parameters so that developers do not need to make initial choices when building a working system. For example, by default the multilayer perceptron (MLP) has one hidden layer and the same number of hidden nodes as input nodes. If users find that this architecture is not suited to their needs, they can use additional functions to adjust either of these parameters.

The RAPID-MIX API aims to focus developers' attention on their intended system design, rather than on ML algorithms or architectures. It presumes an ML architecture common to many

applications involving real-time audio, visuals, or sensor-based interaction, in which inputs (i.e., vectors of values representing current sensor or media features) are sent to a trained model or set of models, which in turn produce a vector of outputs that are passed to some other real-time process. For instance, the sensor values generated by a specific hand position sensed with a Leap Motion (inputs) might be associated with a set of parameters for an audio synthesizer (outputs). The designer of a new system should primarily be focused on reasoning about what inputs are expected, what outputs are desired, and whether the current trained model is sufficient given these criteria—not about which specific ML algorithm should be used.

The API therefore makes a distinction between two types of design tasks (classification or regression tasks, corresponding to the assignment of discrete categories or continuous numerical values), and, separately, between two types of inputs (static or temporal data, which for instance would correspond to a hand position or a hand movement over time). The core API classes reflect this structure, for example “rapidmix::staticClassification.” When we must ask users to learn ML terminology, we take care to use standard terms, such as classification or regression.

The RAPID-MIX API wraps new and existing supervised ML algorithms in a modular fashion, allowing them to be configured for different use cases. Specifically, it includes FastDTW (Salvador and Chan, 2007), XMM (Françoise et al., 2013), Gesture Variation Follower (Caramiaux et al., 2014), k-nearest neighbor, and neural networks. These algorithms were chosen to have low training and run times, and the ability to create expressive models from small training data sets (Fiebrink et al., 2011). They have been integrated as module components

¹⁵LeapMotion, <https://www.leapmotion.com/>

¹⁶MYO, <https://support.getmyo.com/>

¹⁷BITalino, <https://bitalino.com/en/>

and made available in the different API subsets (i.e., RapidLib C++, RapidLib JS, XMM C++, mano-js).

Further classes are provided alongside the ML classes. For instance, there is a single class that provides an API for creating and managing sets of training data. This class is compatible with all of the ML classes, allowing users to switch easily between ML algorithms while keeping the same training set. The API provides signal processing and feature extraction functionality for audio and multimodal sensor data, ranging from basic processes (e.g., low-pass filters or RMS values) to audio segmentation and Mel frequency cepstral coefficients (Logan, 2000). It also provides methods for serializing and deserializing training data and trained models using JavaScript Object Notation (JSON).

The RAPID-MIX API is designed to allow users to test, train, and use algorithms on multiple devices and move easily from one device to another. In order to support native desktop, browser-based, mobile, and embedded applications, the API is available in both JavaScript (JS) and C++. The JS API provides client-side and server-side libraries, targeting desktop and mobile browsers. C++ is intended for low-level audio and media developers, native mobile apps, and embedded processors. It has been tested in openFrameworks and JUCE, as well as on Raspberry Pi and Bela embedded hardware (Bernardo et al., 2018).

The need to support such a wide range of users and use cases inevitably led to compromises. One substantial subset of the RAPID-MIX API functionality, RapidLib, includes the functionality for classification using k-nearest neighbor, regression using multi-layer perceptrons, temporal classification using dynamic time warping, and signal stream processing. The RapidLib subset is implemented in C++ and transpiled into asm.js using Emscripten (Zakai, 2011). This approach provides advantages such as reduced development time, a great deal of consistency across C++ and JS versions of API components, and more efficient JS code (Zbyszyński et al., 2017). The compromises that such approach entails is that RapidLib generated asm.js code base is opaque to JS users. Furthermore, some features that are idiomatic to one specific language such as multithreading and JSON support are difficult to implement across two languages.

In addition to the software libraries, the RAPID-MIX API comes with documentation and examples to help users learn about and experiment with the API. Working examples show users exactly how to implement common use cases, such as using a Leap Motion sensor to control multiple synthesis parameters (Figure 2), or applying classification to an incoming video stream. In addition to describing the API, the documentation also explains many relevant concepts behind the API, such as the application of filters to multimodal input, what a machine learning model is, or how to construct a training data get. Interactive online examples are provided so users can experimentally apply IML workflows to data created in real time in the browser.

In contrast to other ML APIs, the RAPID-MIX API does not provide built-in functionality for quantitative analysis of the performance of trained models. The RAPID-MIX IML workflow is intended to develop quick prototypes and allow

users to subjectively evaluate whether the resultant model is performing adequately, by applying the trained model to new data in real-time and observing the results. When training data are provided interactively, as in the main workflow encouraged by RAPIDMIX API, such direct observation of a model's behavior on new data is often the most effective way to assess a model's performance (and evaluation using more conventional metrics such as cross-validation can be misleading) (Fiebrink et al., 2011).

Listing 1 presents a “Hello World” example of the RAPID-MIX API in C++. Where practical, the same functions are part of the JavaScript API, although obvious differences (e.g., std::vectors) are not duplicated.

LISTING 1 | RAPID-MIX API “Hello World” example in C++.

```
#include <iostream>
#include "rapidmix.h"

int main(int argc, const char * argv[]) {

    //Create a machine learning object for regression
    rapidmix::staticRegression mtofRegression;

    //Create an object to hold training data
    rapidmix::trainingData myData;

    //Set up the first element of training data
    std::vector<double> input = { 48 };
    std::vector<double> output = { 130.81 };
    myData.addElement(input, output);

    //Add more elements
    input = { 54 };
    output = { 185.00 };
    myData.addElement(input, output);

    //Train the machine learning model with the data
    mtofRegression.train(myData);

    //Get some input
    int newNote = 0;
    std::cout << "Type a MIDI note number.\n";
    std::cin >> newNote;

    //Run the trained model on new input
    std::vector<double> inputVec = { double(newNote) };
    double freqHz = mtofRegression.run(inputVec)[0];

    std::cout << "MIDI note " << newNote;
    std::cout << " is " << freqHz << " Hertz" << std::endl;
}
```

3. METHOD

The overall objective of this work was to obtain a deeper understanding about how the design decisions and trade-offs of an API for rapid prototyping of creative technology with IML affect its usability and developer experience. We refined this objective into the following key research questions:

1. What usability issues can we find with the RAPID-MIX API?
2. How do users perceive the RAPID-MIX API design trade-offs and how do these relate to usability and developer experience?

To answer these questions we designed a study with participants who used the RAPID-MIX API in their work and who were asked to report on their experience using an adapted version of the CDs framework questionnaire by Clarke (2010). The questionnaire answers were analyzed using a qualitative approach that is detailed in sections 3.2 and 4.

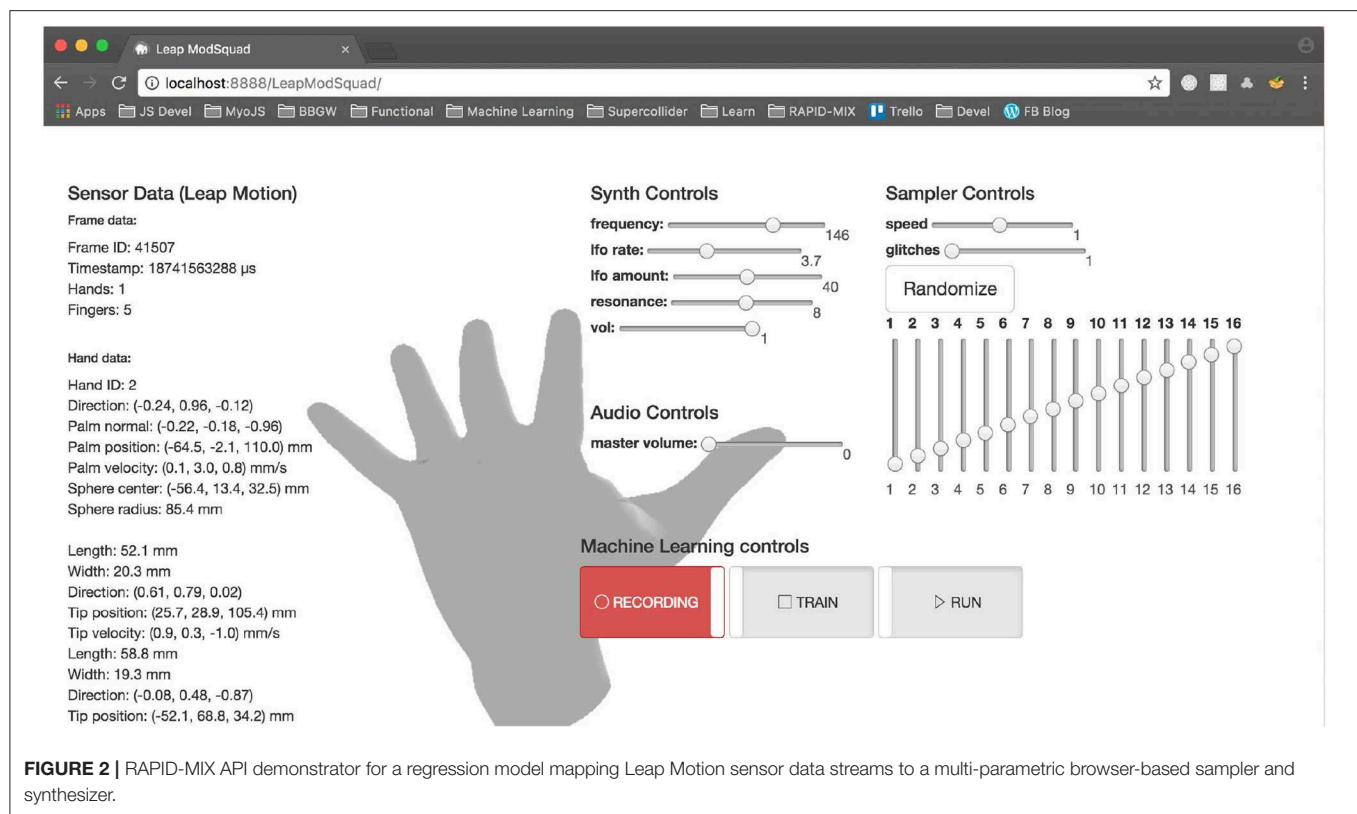


FIGURE 2 | RAPID-MIX API demonstrator for a regression model mapping Leap Motion sensor data streams to a multi-parametric browser-based sampler and synthesizer.

3.1. Participants

We selected participants who had used at least one subset of the RAPID-MIX API within a creative software project. Participants signed a consent form to participate on the study. Our sample set of 12 participants (1 female, 11 males) comprises 6 professional developers working in 3 small and medium-sized enterprises (SME) in creative technology, and 6 creative non-professional developers creating systems for their own personal use (students of different levels, spanning undergraduate, masters level and PhD students; see **Table 1**). Participants had varying software development experience, varying ML experience (from none at all to experience with frameworks such as tensorflow, scikit-learn, etc.). Participants had used different subsets of the API (i.e., RapidLib C++, RapidLib JS, XMM C++, or mano-js) for varying amounts of time (for less than 1 month to a little more than a year). Some participants used the API in personal projects or proofs-of-concept outside the commercial sphere; other projects were developed for commercial purposes in a professional context.

Commercial products created by participants include: a JS front-end component that integrated the IML workflow into a commercial biosignal analysis toolkit for rehabilitation engineers working with patients (P03, P04); an intelligent drum sequencer for iOS with custom gesture activation (P05, P08, P12); and a software-as-a-service that coordinates sonified movement workshops and soundwalks, using the multimodal and multimedia capacities of these collective events attendees' mobile devices (P09).

TABLE 1 | Listing of study participants.

| ID | Software dev. experience (years) | ML experience | API subset used | Time using API (months) | Use (personal, commercial) |
|-----|----------------------------------|---------------|-----------------|-------------------------|----------------------------|
| P01 | 4 | Some | RapidLib C++ | 8 | Personal |
| P02 | 1 | Some | RapidLib C++ | 11 | Personal |
| P03 | 6 | Some | RapidLib JS | 1 | Commercial |
| P04 | 6 | Some | RapidLib JS | 6 | Commercial |
| P05 | 14 | Some | XMM C++ | >12 | Commercial |
| P06 | 5 | Some | RapidLib JS | 5 | Personal |
| P07 | 3 | Some | RapidLib JS | <1 | Personal |
| P08 | 5 | None | XMM C++ | 1 | Commercial |
| P09 | 5 | None | mano-js | <1 | Commercial |
| P10 | 1 | Some | RapidLib JS | <1 | Personal |
| P11 | 1 | Some | RapidLib C++ | >12 | Personal |
| P12 | 7 | None | XMM C++ | 6 | Commercial |

3.2. The Cognitive Dimensions Questionnaire

We employed an adapted version of the CD's framework questionnaire (Clarke, 2010) as our research instrument, which appears in **Appendix A**. This questionnaire has been developed to support a comprehensive and systematic understanding of participants' experiences with the API, broken across several different dimensions. Clarke's questionnaire provides

TABLE 2 | The adapted Cognitive Dimensions framework used in our study.

| | |
|------------------------|---|
| Learning | |
| Abstraction Level | Magnitude of abstraction and style of abstractions of the API |
| Learning Style | Learning requirements and style encouraged by the API |
| Penetrability | Ease of access, retrieval, exploration, analysis, and understanding of the API components |
| Understanding | |
| Consistency | Similar semantics are expressed in similar syntactic form |
| Role-expressiveness | Purpose of an API component is readily inferred |
| Domain Correspondence | Clarity of domain mapping of API components |
| Usage | |
| Working Framework | Size of conceptual chunk or amount of context necessary to work effectively |
| Elaboration | Extent to which API must be adapted to meet developer needs |
| Viscosity | Resistance to change in refactoring |
| Premature Commitment | Constraints in the order of implementing API code |
| Error-proneness | Error incidence, recoverability and support |
| Application | |
| Work Step Unit | Amount of programming task completion achieved in a single step |
| Progressive Evaluation | Work-to-date can be checked at any time |
| Testability | Types of evaluation and assessment metrics that are adopted |

several benefits over the original CDs questionnaire, as it is tailored for API evaluation, and it also has an increased focus on learnability (Clarke, 2010)—i.e., introducing additional dimensions including Learning Style, Penetrability. We were inspired by prior studies that fine-tuned Clarke’s questionnaire to specific domains—e.g., Watson (2014) introduced high-level groupings of dimensions for a more effective distillation of results for improving API documentation planning; Wijayarathna et al. (2017) aimed to evaluate aspects of their API that were specific to cryptography by introducing additional dimensions (e.g., End-user protection, Testability). We have also adopted some of these dimensions with minor changes. **Table 2** summarizes the 14 dimensions used in our questionnaire, grouped into four high-level themes. Each dimension was addressed by several questions (**Appendix A**).

We first delivered a pilot version of our questionnaire to two participants. This version was longer and closer to the original version by Clarke (2010), and we received complaints about its length. We therefore shortened the questionnaire by removing some redundancies. The final questionnaire was delivered online, on paper, or through in-person or remote interviews, due to the geographical spread of the participants.

4. RESULTS

In this section, we report our findings about each of the dimensions included in our questionnaire. We employed content

analysis using NVivo to analyses responses. We adopted a deductive analytical approach in which we used codes based on the CDs framework and on the higher-level themes of **Table 2**, and on an auto-encoding analysis performed with NVivo.

We also tried to find correlations between the variables Software Development Experience, ML Experience, API subset, and time using the API, in the closed-end questions of each dimension (e.g., Q1—perceived level of abstraction, Q8—learning experience, Q11—experience with amount of context, etc.; **Appendix A**). Given the size of our sample, we ran Pearson’s chi-squared test with Yates correction, and Fisher’s exact test. We found no support for contingency between those variables in the dimensions’ quantitative results as none of the tests yielded statistical significance.

4.1. Abstraction Level (Q1–Q2)

Questions pertaining to this dimension aimed to investigate the appropriateness of the abstraction level of the RAPID-MIX API. We asked how appropriate the abstraction level was for participants’ development needs and why (Q1, **Appendix A**), and whether participants felt they needed to know about the API’s implementation details (Q2).

In responses to Q1, 7 of 12 participants found the overall API abstraction level “just right,” and 5 of 12 found it “too high level.” No one found it “too low level.” Five of the 7 participants who had used the API for longer than 2 months found the abstraction level “just right.” Participants who used different subsets of the API differed in their responses; all the participants using RapidLib C++ (4 participants) or mano-js (1 participant) considered these had the right abstraction level, and 3 of 4 participants using RapidLib JS considered it too high-level.

Participants who found the abstraction level just right described how the abstraction level enabled them to achieve their development goals (P01, P02, P06, P11). These included rapid prototyping (“...I was able to do rapid prototyping, rapidly!”), P06), simple implementations (“for a quick and simple implementation the abstraction level works well”, P03), and proofs-of-concept (P04). Participants also referred to the positive development experience the API provided, having found it “extremely easy to use in C++, which is usually a very confusing language” (P02), or non-obtrusive to the creative process—“I was able to implement most of the RapidLib functionality without losing my creative flow” (P06). P04 indicated that the API “facilitates the final programmer use” and saved her a lot of time by preventing her from having to handle implementation details.

Participants who found the RAPID-MIX API too high level (P03, P05, P07, P08, P10) experienced problems mainly because they needed further understanding of lower level details—“when I tried to learn a little more, knowing, for example, which model was being used, I saw the abstraction level as a hindrance” (P03)—or finer-grained control over certain API features—I found that while the algorithms work, I would have liked a bit more control over certain algorithms” (P10). Some participants complained about the lack of transparency of RapidLib JS’s high-level objects (Classification and Regression) which prevented them from knowing which algorithms were in use. Because RapidLib JS is

transpiled from C++ to asm.js, the source code and algorithm implementation is more opaque than the C++ version.

Participants who stated that they needed to know the underlying implementation (Q2) presented different reasons for this. Three participants (P05, P07, P10) found undocumented algorithm parameters—e.g., *k* in the *k*-nearest neighbor algorithm, the number of hidden units in the hidden layers of the multi-layer perceptron—and needed to understand how to use them, so had to look further into the implementation. Some of these participants worked on a product and related their reasons to needing a deeper understanding for customer-facing projects (P03, P08). For instance: “I needed to know what was behind the object. If I am going to make a product or give a chance to a customer to use one of our solutions based on the result of the api, and for some reason, something wrong happens it would be necessary to have a deeper knowledge of the whole object.” (P03).

One professional developer, P05, considered the understanding of the underlying implementation vital—“The API is very concise and there’s not much to learn, however choosing the correct parameters is a ‘dark art’” (P05). One participant found the library opinionated (“it is intended to work in a specific manner”) and had to look to the implementation to adapt it to their needs.

Participants who mentioned not needing to know the underlying implementation either mentioned that they already knew it, or, that they had the sufficient knowledge to be able to use the API successfully—“I felt that I needed general knowledge of how regression and classification algorithms worked. However, for my purposes this was enough. I could then just use the API without needing to know the exact implementation.” (P11).

4.2. Learning Style (Q3–Q9)

The questions about learning style aim to determine what knowledge was essential to use the API successfully, how much new knowledge participants had to acquire, and how participants went about using API documentation to attain this knowledge.

Participants perceived knowledge of the following ML concepts to be important in facilitating use of the API (Q3): the probabilistic nature of ML (P05); the greater importance of the choice of data in comparison to the choice of algorithm (P01, P05); basic ML concepts such as the difference between regression and classification (P02, P04, P11); the stages of the supervised learning workflow, such as collection and preprocessing of data, training and running the models (P01, P03, P04); and understanding the ML algorithms’ implementation and inner workings (P01, P03, P07). They also identified the following knowledge of non-ML topics as useful (Q4): threading and non-blocking async architectures (P06), client/server architectures (P09), deeper programming language knowledge (e.g., using generics) (P12), statistics for modeling data (P05), and practical knowledge about sensors, and human-computer interaction (P11).

Participants’ responses about their learning strategies (Q6) indicated that both novice and experienced developers tended to adopt an opportunistic approach (Clarke, 2007) to learning about the API: they frequently learned by copying sample code and employing hands-on exploration. The more experienced

developers appear to have complemented this with a more top-down approach to learning about the API components or architecture.

The majority of participants (9 of 12) indicated that they had to learn “just [the] right” amount to use the API (Q8). These participants defended this response with answers that mentioned the simplicity, ease of use, and beginner-friendliness of the API. For instance, participants wrote that the “code of the API is simple and concise” (P05), that it was “straightforward to use without having to read too much documentation” (P07), and that “I didn’t have to learn anything new to use the API and I didn’t want to learn a lot to train such simple models” (P02). The other 3 participants stated that the RAPID-MIX API documentation did not provide enough resources to support their learning, particularly regarding choosing appropriate algorithms and their parameterizations for a given problem. P12 wrote “one is left guessing numbers and trial and error exploration, if there’s no previous ML experience,” and P01 wanted “more complex examples so that people can try different ML structures.”

4.3. Working Framework (Q10, Q11)

Q10 aims to elicit an understanding of the amount and type of information (i.e., “context”) a user needs to maintain or keep track of while working with the API. Eleven participants responded, choosing multiple elements from the list of contextual information (one participant did not respond). The top 5 classes of context identified as necessary by respondents were: API methods (10 of 11 participants), API classes (8), data structures for training and model input/output (7), database (e.g., JSON, XML, CSV, database management service) (7), and types of ML algorithm (6). Less common responses included: local scope variables (4 of 11), system configuration settings (4), app configuration settings (3), global scope variables (2), registered events (1).

When we asked participants to describe this “amount of context” was “too simple,” “too complicated,” or “just right” (Q11), 8 of 12 participants reported it was “just right”. Participant explanations suggest this was driven by the simplicity of the API interface—“not very demanding, inasmuch as methods presented simple syntax. When developing, I didn’t usually keep track of them. When problems arose, it was always easy to navigate to tutorial examples and spot where scope or method syntax was not correct.” (P03). Contrastingly, the participant with the most ML expertise conveyed his reasons for what is important about context using more of a conventional rationale around context—“I needed to keep all the context in mind that is relevant to using the algorithm. I guess I don’t care that much about the particular data structure that the API expects, so it would be nice to not have to think about that. I don’t see how you could avoid that though” (P06).

Two respondents found that the amount of context they needed to keep in mind was too complicated. One of them, the less-experienced, found difficulties with developing architectural support for an increasing number of model outputs—“adjusting the output parameters of my application took a bit of time and thought to figure out what parameters needed to be global and what parameters needed to be local.” (P10). The other

respondent, a more seasoned developer, implemented a non-blocking asynchronous threading architecture for making a robust use of the API—e.g., “Training with large amounts of data can take a long time and should be non-blocking e.g., a future. However, it also needs to be cancellable.” (P04)—which entailed the use of a more comprehensive and complex “context.”

Interestingly, one participant referred specifically to training data and its specificities for the IML workflow as part of the ‘context’ to be kept in mind—“Often in the Application I would visualize the training data or rely on remembering it. So just being able to save the model without the training data was not useful and caused complexity issues. Especially when the training time of the models is very short and the datasets are small” (P02).

4.4. Work Step Unit (Q12)

Q12 asked participants whether the overall amount of code they had to write to integrate the API into their projects was too much, too little, or just right. Eight of 11 participants mentioned that their experience was just right.

The remaining participants answered that they had to write too much code. Their explanations identified several tasks that appeared to require too much code: (a) validation, (b) data management, (c) concurrency management, and d) management of model outputs. Three participants mentioned validation code as necessary to make their application safe and robust—e.g., “there’s not too much error handling, or check on the data format/range” (P12). Two participants referred to concurrency—e.g., “not ‘too much’ code directly related to the API, just too much boilerplate wrapper code in order to use the API successfully in the context of a large multithreaded mobile app with GUI and audio” (P05). Three participants mentioned having used an extensive amount of code for creating data structures and performing data management in the application. For instance, “I had to write lots of code for formatting training data, I feel like the API could have given an interface for recording, building and editing data sets rather than needing to be given the whole dataset at once or relying on user-written C++ vector functions to edit training data” (P02).

4.5. Progressive Evaluation (Q13)

Q13 asked participants about the amount of work needed to evaluate progress in using the API. Notably, though participants knew the questionnaire was focused on evaluating the API itself, we found that the majority of responses related to the task of evaluating progress of the IML workflow outcomes (i.e., the quality of training data, the training process, and the model results) rather than just progress in establishing a functional pipeline.

Participants identified the simple API interface as facilitating progress evaluation—e.g., “It was very easy to evaluate the integration of the API with my application. Because of the simple user interface of the API, I knew exactly what to expect from each method of the API.” (P02); “there is very little code required to use the API. Evaluating the performance of the tool, selecting the source data inputs, choosing a frame rate, ensuring orthogonality took up 10% of our time.” (P05).

Responses that expressed difficulty in evaluating progress shared some common themes. For instance, respondents complained about the lack of integrated visualization tools—“I evaluated my progress in using the API by implementing visualizations in D3 [...] I would probably like to minimize the amount of time spent on visualization code” (P07). Others complained about the lack of functionality to provide feedback about model accuracy improvements—“There’s no proper visualization of improvement, one is left with the trial and error to determine if the classification is improving or not, and no information on how good/bad it is.” (P12). One participant referred to the high abstraction level as a hindrance for progressive evaluation—“Because some of the functionality of the API is hidden for advanced or personalized use cases, I wasn’t completely sure about my own progress” (P06).

4.6. Premature Commitment (Q14–Q17)

Q14–Q17 examined how participants perceived the level of premature commitment required by the API—i.e., the need to make certain decisions too far in advance, and inflexibility in the order in which decisions had to be made.

Eight of 12 participants reported that they were forced to think ahead and make early decisions (Q14). Most participants found it necessary to make early decisions about data sources and preprocessing, data structures for inputs and outputs and their dimensionality, and the state machine architecture that supports switching between the modes of training and running the models (Q15). Some of the more advanced users, or users with more complex requirements for commercial product implementations, referred to planning the integration of the API components according to specific aspects of their use case—for instance, within a client-server or concurrent architecture.

4.7. Penetrability (Q18–Q23)

Questions about penetrability aim at understanding the degree of ease with which developers could access information about API components, and explore, analyse and understand their working details in order to achieve specific development goals.

Eight participants encountered some difficulties in finding necessary information about API details (Q18, Q19), indicating that the documentation of API subsets was insufficient. Most of these respondents had, at some point, finer-grained implementation requirements for which necessary details about the API became hard to find. Seven participants indicated having to learn about specific ML algorithms and parameter configurations (Q20). Some participants learned about these as they worked—e.g., “Online tutorial materials and examples were very helpful. However, should deeper potential of the API be explored, I can’t say that all questions would be easily answered.” (P02); “As my own knowledge [of IML] progressed I would have liked to be able to find out more detailed information about the neural network and how it performed the regression analysis” (P03).

Participants reported working to improve their understanding of the API (Q22) mainly through the process of trial-and-error exploration (5 participants) and by reading through the API source code (4 participants)—“Largely through trial and error I

began to get a sense of how the regression model worked” (P08); “By breaking it, using intellisense to find functions that were templated to exist, but did not have implementations in some models, so I started reading more of the API’s source” (P01). Some participants reported needing to use direct communication with the API developers (P01, P06, P09) and resorting to external documentation resources (P05).

Four participants believed the API and its documentation provided enough information for their needs (Q23), found easy access to that information (Q19), and that there was no lack of information about details (Q18). Most of these participants either had simple goals and remained at a high implementation level, or their exploration was technical-driven rather than design-driven—“My original interest [lay] in the C++ API, but resources and adaptation to the final product needs made me shift toward Javascript, which had magnificent learning materials” (P04). Some participants admitted not understanding the working details but were satisfied working with a “black box”—e.g., “I didn’t fully understand then. The results were adequate enough for our application” (P09); “I had no knowledge of the implementation details or how the API is generally structured apart from what was obvious from the code examples” (P05).

4.8. Elaboration (Q24)

Q24 asked about the ways that participants had adapted the API to fulfill their design goals (if any). Five of 12 respondents used the API “as-is.” Five others reported having wrapped the API in adapter classes to add the necessary functionality to overcome specific limitations of the API. Three of these respondents had added error handling and support for asynchronicity.

Two participants reported having forked the API and changing the library file structure. One respondent hacked the API to improve the learning capacity of the default regression object. His hack approximated the functionality provided by an undocumented MLP parameter—“The hack was to increase the dimensionality of the input vectors by duplicating their content. This would artificially increase the number of hidden units and allow the model to learn more complex patterns” (P06). No respondents reported trying to derive classes or override class methods.

4.9. Viscosity (Q25)

Q25 aims at understanding how easy it is to make changes to code that uses API calls. Seven of 12 respondents mentioned it was easy and two mentioned it was very easy to make changes to API integration code (Q25)—“Easy, there was barely any code to write to implement the API.” (P02); “Very easy. The interface is minimal and the actual parameters that one can change are few” (P12). Three respondents mentioned they did not need to refactor their code. The other two respondents described challenges around understanding the code in the context of refactoring it—“Easy as I wrote the code [...] When debugging issues though, I needed to check examples a lot to understand the described Test-Train-Run structure that I needed to implement. As in ‘to train only once and not to run the model when testing or training.’” (P01); “It was easy but needed a lot of understanding of the code.” (P08). One participant referred to the growing amount

of outputs as a difficulty for change—“As the amount of output parameters grew I found it sometimes difficult to keep track. Otherwise it was very easy” (P11).

4.10. Consistency (Q26)

Q26 asked participants if they noticed API elements that offered similar functionality, and whether the differences between them were clear (Q26). Five of 11 respondents mentioned having noticed consistent method names across classes. Three of the aforementioned 5 found lack of clarity between certain API classes—e.g., “Model set, Regression and Classification. The difference between these objects was not clear. The implementation[s] were all very similar and it was not clear which one to use” (P02). There were also issues around the use of the different kinds training data structures. The other two who noticed consistency of methods felt they understood the differences between them. For instance: “I like that there were a train, run functionalities in the code as this help me understand the models in similar way apart from the inner workings of course” (P01). The remaining respondents (6 of 11) did not noticed such similarities; one participant did not respond.

4.11. Role-Expressiveness (Q27–Q29)

We asked participants if it was easy to read and understand code that uses the API (Q27), and whether it was easy to know which classes and methods to use (Q29). We obtained unanimous responses to both questions—“Everything was very easy to interpret.” (P02); “Code is pretty self-explanatory and comments are concise enough” (P04) “Classes methods are efficiently named to understand what they are doing” (P08).

4.12. Domain Correspondence (Q30–Q32)

Questions about domain correspondence aim to determine whether API classes and methods map easily onto the conceptual objects in the users’ implementation.

We obtained unanimous positive responses about the ease of mapping the API code into developers’ conceptual objects (Q30). Two respondents provided reasons that related the simplicity of the API interface and the IML workflow to the ease of mapping to domain and conceptual objects of their implementation (Q31)—“the simple user interface made prototyping very quick making building a conceptual idea very easy and simple.” (P02); “I think because the training and the recognition phase is the same workflow, it’s easy to come up with concepts that match both.” (P07).

Participants seemed to have had a particular understanding of what was meant by the “mapping” of the API to an application domain (Q30); the majority of responses mention mapping API objects to classification or regression tasks, or to the IML workflow tasks. Most likely, participants have understood ML learning functions such as classification and regression, as enablers of functional mappings between domains (e.g., mapping gesture to sound, visuals, and discrete application events). This seems to be confirmed by the results of asking participants to provide examples of conceptual objects (Q31); only a few participants were able to refer to conceptual objects that did not overlap directly with ML domain concepts—“Once I had a clear

idea how I wanted to activate certain functionality, this made the process easier for me.” (P01). “Because the API enable separation between “recording” the data (i.e., training) and estimating the probabilities of category membership for an unknown example (recognition)” (P03).

4.13. Error-Proneness (Q33–Q36)

Questions about error-proneness aimed to elicit the participants’ experiences with encountering and recovering from errors in their use of the API.

Eight of 10 respondents reported that they had used the API incorrectly (Q33). Errors included: using an inconsistent number of data features between training data sets and test data sets (P05, P06, P09), using malformed data (P04), using labels inconsistently (P12) or malformed JSON (P05, P08), using a large-size training datasets which caused a crash (P11), attempting to predict from a model in an untrained state (P02), and using a higher-abstraction level object as a primitive (P02). Many of these incidents were caused by limitations in input validation of API methods.

Four of these respondents indicated that the API did not provide sufficient help to identify misuse (Q34)—e.g., no error messages, some “undefined behavior” output. Participants reported having experienced crashes of the API-client application without any notification with the subsets XMM C++ (P04, P10, P11) and XMM JS (P09). One participant resorted to logging (P09) and contacted the API developers directly to find and resolve the issue.

Most respondents indicated they were able to find a way to correct their use of the API (35). For instance, where participants encountered errors due to lack of input validation, they adapted the library to implement validation (P05, P12). Other participants simply became more aware of problems and more careful (e.g., in structuring the training data, choosing the correct dimensionality of inputs and outputs, validating model state, etc).

4.14. Testability (Q37–Q39)

Questions about testability aim to determine the types of evaluation and assessment metrics that were adopted by participants as they used the API and concluded their implementation of integration code.

Most participants indicated having used subjective evaluation to assess the results of the trained models (9 of 12), with criteria such as correctness (3), cost (3), decision boundary characteristics (1). Several participants referred to other criteria such as expressivity (1) or more creative ways of evaluation—e.g., “No testing was done on the models, just eyeing up the output and judging it creatively whether it works or not for the desired output” (P01). One participant mentioned having used seam tests for assessing training data. One participant did an objective accuracy evaluation of the models built with the API using unit tests with another ML library.

Seven of 11 participants found the API did not provide guidance on how to test the resulting application. The remaining respondents did not look for guidance for testing—e.g., “We tested very informally since there’s no effective way to test more objectively” (P12).

5. DISCUSSION

According to Clarke (2010), the CDs inspection can tell whether there are significant differences between what an API exposes and what a developer using the API expects. In this section, we use the results of applying the CDs questionnaire with RAPID-MIX API users to discuss design trade-offs with respect to developer experience and ML API usability. We use these insights together with our experience designing the RAPID-MIX API to provide recommendations for the design of ML APIs for prototyping music technology.

5.1. ML API Design Trade-Offs in Relation to Learnability and Understandability

Results indicate that the RAPID-MIX API qualifies as an ML API with a high or aggregate *abstraction level*. The high *abstraction level* is supported by its minimal surface area comprising a small number of classes, methods and parameters. These elements have been subsumed into a simple conceptual model of high-level design tasks and basic data structures. An ML API has direct *domain correspondence* if ML is considered its domain of correspondence. In the understanding of most users, RAPID-MIX API entities map directly onto ML learning tasks.

The high *abstraction level* appears to be consistent with the *learning style* of the RAPID-MIX API, which is more of incremental and step-wise. Both novice and experienced developers reported an opportunistic learning approach (e.g., having hands-on exploration and progressing through code examples, exploring, changing or copying sample code to their projects). Arguably, given that ML learning tasks and the algorithms require extensive description from API providers and learning from the users, this indicates that the learning and assimilation of ML concepts was successful. ML APIs with these characteristics can provide ML -non-expert users with adequate scaffolding for a more satisfactory and forgiving learning experience.

However, more experienced developers reported to have complemented their learning strategies with a more systematic, top-down structured learning approach to the components and architecture of the API. More advanced developers and more technically complex scenarios might require the flexibility and control that a lower-level ML API with more primitives, more ML algorithms and more exposed parameters for finer-grained control can provide. We found that a few respondents, the more experienced developers or the ones who had specific implementation requirements (e.g., finer-grained control, strict end-user concerns within customer-facing projects) needed to go “beyond the interface” to inspect the API source code and learn more about underlying ML algorithms. In that exploration, a few of them found useful parameters that had not been exposed. This finding informed a subsequent re-design to expose the parameters.

In scenarios of exploration and intuition building about ML, ML APIs with surface-level *penetrability* may appear to provide everything that is required to enable successful use and integration with client application code. Nevertheless, surface-level ML APIs may allow “black box” approaches in

its application and use. We found that the RAPID-MIX API was no exception to this. As developers build up knowledge and understand the IML workflow, which ML tasks to apply, or the number of inputs and outputs to use in a ML pipeline, they may seek to push forward their understanding of a ML model behavior. They may engage in a deeper exploration and experimentation to learn about the ML API intricate working details, such as the impact of choice of underlying ML algorithms and parameter change.

In the RAPID-MIX API, the overall *penetrability* is mostly sensitive to context and to implementation needs, with different RAPID-MIX API subsets providing distinct levels of *penetrability*. There were cases of deeper exploration fraught with fragmented documentation, and unclear dependencies between API primitives and abstractions. This gives the RAPID-MIX API a core *consistency* rather than full *consistency*. These issues affect the perceived *consistency* of an ML API, and consequently, its learnability. For instance, some participants resorted to external resources to understand ML concepts and algorithms, which may be considered resorting to a top-down approach to learning ML.

Different areas of an ML API may have distinct levels of *role expressiveness* which also affects its *consistency*. In most cases, the purpose of the RAPID-MIX integration code was correctly interpreted and matched user's expectations. Nevertheless, there were issues which prevented it to fully match users' expectations which gives it a lower *role expressiveness* as an ML API. One opaque subset (i.e., RapidLib transpiled from C++ to asm.js) prevented one user from determining the underlying implementation. As mentioned before, other users found undocumented lower-level methods or lacked configuration settings. The transparency at the level of the ML algorithm—or ML explainability—is another layer that may entangle with the overall ML API *role expressiveness*. However, ML explainability is a current and significant research problem that is out of the scope of this paper.

5.2. ML API Design Trade-Offs in Relation to Usability and Applicability

An ML API with a high-level, easy-to-acquire conceptual model can cater well to the opportunistic approach and needs of ML non-expert developers. In the case of RAPID-MIX API, a simple conceptual model based on ML tasks and simple data structures with inputs and outputs, makes it suitable for simple implementations and rapid and pragmatic prototyping with IML. It also helps us to uncover and better understand usage and application of ML APIs by ML non-expert users.

ML APIs with a high *API elaboration* should not impede any kind of user from achieving their design goals. They should enable great flexibility to the more proficient end of the user spectrum, such as the implementation of custom behaviors, custom ML pipelines and parameterization. Almost half of participants reported using the RAPID-MIX API “as-is” to meet their design goals. The other half required further API elaboration (e.g., more ML algorithms, more parameters, better error reporting). This tells that for users with simple goals the RAPID-MIX API was sufficient. Alternatively, it can tell that, for more critical users, or, users with more sophisticated implementation goals, the API was not sufficient.

Arguably, the RAPID-MIX API exhibits a medium level of *API elaboration* as advanced users may use its extensible architecture to extend the API default capabilities with custom implementations. The few participants who extended the API default objects did so using adapter classes to extend the default objects and methods with validation, concurrency, and error reporting. However, these users improved upon base limitations of the ML API. For a user, extending an ML API might defeat the whole purpose of using it in first place. Users who do not expect, or do not have the know-how to extend the existing functionality, might find problematic in doing so. They may opt for using a different ML API or framework altogether, or resort to integrate independent ML algorithms.

Developers integrating an ML API in their client application code need to keep track of the information which enables them to work effectively (i.e., the *working framework*). Interestingly, half of the respondents did not mention ML algorithms as part of their *working framework*. This might reflect a trade-off with the *abstraction level* of the API; or alternatively, the adoption of specific ML API design assumptions (i.e., in the case of RAPID-MIX API, data and use cases on the foreground of users' attention and ML algorithms on the background). The lack of preponderance of the ML algorithm may be unsurprising if it reflects minimal ML requirements or a local *working framework* (i.e., ML API objects and methods, local variables) that suffices for simple implementations. However, the *working framework* may not be entirely or directly represented by the ML API or the scope of the ML API integration code—e.g., extrinsic elements such as the ML training data, or in a global or system-level working framework, client application and system configuration settings, external device data specifications, performance requirements, etc.

In a minimal test (e.g., hello world example, unit tests with a ML API) the *work-step unit* might be local and incremental. Despite the minimal surface area of a ML API, developers may have design requirements that scale the quantity of ML API integration code extensively. In these cases, an ML API can have a parallel *work-step unit*, where the steps to implement and achieve the full design goals are distributed throughout different scopes in the integration code. Given the interactive nature of the IML workflow, the ML API integration code will most likely scale up to comprise multiple and independent code blocks. This was the case with a few of the implementations with the RAPID-MIX API, e.g., asynchronous event handlers for collecting data and building an ML data set on the fly, for triggering evaluation of new data, or persistence to data repository. ML API integration code may also require the instantiation of multiple auxiliary objects that interact together (e.g., GUI, data management, validation, concurrency), which make using and understanding more challenging.

Similarly, an ML API may support a *progressive evaluation* of the integration code at local level, functional chunk (that is, after the implementation of certain groups of tasks, such as setting data structures and training data set, or after the train and run methods), or parallel components (i.e., multiple and independent code blocks). The majority of respondents reported needing a fully functional pipeline and to experiment with IML workflow in order to check progress on the overall

implementation task with the RAPID-MIX API. An ML API may support *progressive evaluation* at parallel components, as it requires a fully functional implementation and interaction between different ML API objects.

An ML API that presents the user with a small number of choices about how to accomplish design goals with minimal implementation differences between alternatives, can expose a minor and reversible level of *premature commitment*. The RAPID-MIX API also has a low level of *viscosity* that allows users to easily make changes and refactor integration code. This is consistent with the notion that raising the *abstraction level* reduces *viscosity* (Green and Petre, 1996); low *viscosity* is also supported by the API small-surface area. Such ML API qualities invite a trial-and-error exploration and an opportunistic approach, and are supportive for ML-non-expert users.

The RAPID-MIX API situates at a medium level of *error-proneness* given the reports about recurrent issues of misuse, error support and recoverability. These findings indicate opportunities and the direction for technical improvements, such as providing more robust validation of inputs, and better communication of error status through error messages.

Concerning testability, the RAPID-MIX API promotes more of a direct and informal evaluation using subjective criteria. This confirms its alignment with the IML approaches that the API is inspired on. In any case, most developers seem to be unaware of different methods or evaluation alternatives, and seem find the concept difficult to articulate. Also noted was the lacking of guidance about evaluation alternatives, which seems to require specific ways to be successfully transmitted, such as with richer media.

5.3. Recommendations for the Design of ML APIs for Prototyping Music Technology

1. *Adopt a user-centered infrastructural software design approach*—find domain-specific core test applications early on which might be compelling to users and help them to understand, and that can inform the ML API core design features (Edwards et al., 2003). In the music technology domain, these could consist of, for instance, a range of new sensor-based interaction applications with complex mappings to music processes (e.g., the main focus of RAPID-MIX API, GRT, and related toolkits), automatic music composition applications (e.g., Magenta), or other types of automation for music production environments. These applications can help to determine many of the ML API design features such as the surface area, its elaboration level (or extensibility) or abstraction level.
2. *Reduce the skills and usage barriers with a minimal code footprint and reasonable defaults*—design the ML API abstraction level to lower the entry barrier for ML-non-expert-users by abstracting details away. Design for improved readability, efficiency and reduce cognitive load with terse ML API code and minimal boilerplate. Users starting with ML API functions should experience a default or typical behavior with default parameters. For instance, RAPID-MIX API offers a high abstraction level, in which ML tasks are high-level objects and data structures are simple arrays. This contrasts with the abstraction level of Tensorflow and GRT, with tensors as data structures or low-level math operations. Reducing the complexity and the number of possibilities of building ML pipelines can accelerate the immediate engagement with the ML algorithm and data, both programmatically and via an IML workflow. This can foster a bottom-up learning style, and provide an increased focus on their design goals.
3. *Facilitate the first contact through an immediate hands-on-code experience*—minimize the cognitive load associated with installation issues to accelerate the first contact at structuring an ML pipeline and fully experiencing an IML workflow for a musical application. Users find it difficult to adopt a tool if they are not able to see it working quickly and providing compelling results, and the installation steps can drastically undermine the developer experience. ML APIs such as tensorflow.js, ml5.js, and the RAPID-MIX API, which offer “zero-install” access to model building and inference in a browser environment can be very compelling for novice users to start with. Similarly, users can benefit from plugins which wrap up ML API components and ease the integration with environments such as Max, Pd, or OpenFrameworks.
4. *Provide adequate conceptual scaffolding for the ML API code*—help the user build an adequate mental model for the ML integration code using different abstractions, if possible from the domain of application, such as end-to-end pipelines, modular building blocks, and training and inference workflows. This can help users to better understand, not only the alternatives which the API makes available (i.e., ML algorithms, objects, and data structures) but how they fit within the working framework required to accomplish their design goals when building musical applications. ML API users building intelligent music systems will develop a working framework of how to set integration hooks between the inputs and outputs of an ML pipeline and areas of the client code (e.g., the controller data streams, the UI event handlers, the audio engine).
5. *Provide many code examples of real-time interactivity between user, data and ML algorithms that can be applied to musical processes*—provide support for the development of intuition and basic understanding with an experiential approach and contrasting ML API code examples that gradually disclose complexity. This will provide users with a smooth learning curve and experience to building ML pipelines and workflows for musical applications. A real-time IML workflow where the end-user creates, curates, and modifies training data iteratively to build ML models mapped to musical parameters, and steer their behavior based on direct observation, trial-and-error and hands-on-exploration, can yield a smaller gulf of execution and evaluation (Norman, 2013) than other workflows. Code examples can support opportunistic approaches—i.e., hacking, appropriation, e.g., ESP (Mellis et al., 2017)—to ML systems development, which might be more enticing to novices or aligned with the goals of rapid prototyping musical applications. Novice users tend to use code examples as the basis and starting point of their music technology creation, so they might be written as building blocks.
6. *Design your API documentation with relevant ML terminology and curated external resources*—design the documentation

with an adequate level of penetrability to support an effective learning experience—e.g., navigation, content structure, links to API code (Meng et al., 2019). Limit the use of specific ML terminology conforming with standard terms, while aiming “at the most common case”—e.g., the *Glossary of Terms* (Kohavi and Provost, 1998)—applied to the music domain. For example, understanding the meaning of classification both as an ML task and musical design task (e.g., musical gesture recognition) may lead to cognitive benefits—standardization for improved memorability (Norman, 2013), a general principle which claims usability can be increased by having to learn about a term only once, which potentially lowers the barrier to participation. Documentation includes code commenting practices and the curation of links to third-party content which can provide competent and alternative means of explanation—broader, deeper, or more interactive and engaging (e.g., Youtube, StackOverflow, online playground tutorials).

7. *Error reporting, progressive evaluation and feedback mechanisms*—reduce the evaluation gap of the ML API integration code and help users to recover from errors by providing them with error messages for the relevant parts of the ML API. Most errors identified during usage of RAPID-MIX API were related to ill-structured data sets, and inconsistency in labeling, types and number of input and outputs. Build input validation on the methods of your API to help users recover from run-time errors. Error messages should be clear, objective, and indicative of the problem and the context where it appeared, and propose a solution. The lack of progress and termination feedback in the ML model training stage was considered problematic. Methods or functions which complete asynchronously such as ML model training, benefit from progression and completion feedback. Provide API methods and complementary tooling (e.g., visualization libraries) for accessing the configuration and state of the ML pipelines and workflows. Use them in the examples and documentation to help users build a correct mental model from a solid pattern and prevent errors.
8. *Support the diversity of user engineering skills for ML -non-experts users*—novice developers require a strong proposition with regards to the conceptual scaffolding. This might entail creating more visual and holistic resources, which might convey more effectively the “big picture,” and creating minimal and simple examples, with a standard code styling and no optimization for readability. Experienced developers require another level of elaboration and penetrability to reach their design goals. They will value lower-level primitives for control and configuration of ML algorithms and pipelines, a wider selection of ML algorithms and more sophisticated data structures, which may yield more expressiveness in the final result. To strike this challenging balance between both ends of the spectrum of user developer-skills, it is fundamental to build an extensible and modular ML API architecture. It is also important to differentiate documentation and guides according to user development skill levels and to tailor and provide support for a more adequate learning journey.
9. *Build a thriving community and ecosystem comprising documentation, resources and infrastructure*—an active

community can support new users with the on-boarding process and with troubleshooting issues. It can also give more experienced users opportunities to contribute with solutions, mentor, network, and peer-review open-source contributions and extensions to an ML API. Online fora and Q&A platforms such as StackOverflow provide the media for the community to engage and interact and keep a history of answers to issues previously raised by other users. Meetups, workshops, and hackathons can grow the community offline and strengthen its bonds.

6. CONCLUSION

This study employed a qualitative and user-centric approach to explore and better understand how ML API design may facilitate the learning, use and rapid adoption by creative software developers and music technologists. The design of an ML API can impact its adoption, the user-developers’ working processes, and the client application features and interaction style. Current ML API designs and designers show awareness about the importance of adopting design principles which guide usability, learnability and accessibility. However, research focused on the human-centered design, evaluation and developer experience with ML APIs is fundamentally under-explored, in particular, of ML APIs specialized in the development of systems for creative and musical technology. This kind of user study is therefore important for how it builds upon a more nuanced connection between designers and end users of an ML API. We used an adapted version of the CDs questionnaire to explore how the design decisions and trade-offs of an API for rapid prototyping with IML relate to its usability and the developer experience.

The application of the CDs to the usability assessment of ML APIs helped uncover problems and directions of improvement, mostly related to documentation fragmentation, support for understanding intricate working details, error support and recoverability, and lack of evaluation guidance. Results also indicate that the RAPID-MIX API caters well to beginners and ML-non-expert users in general. It appears to support incremental learning approaches and to provide a low entry barrier and smooth learning curve to ML. The direct correspondence of the API to a high-level conceptual model which focuses on supervised ML learning tasks, end-to-end ML pipelines and simple data structures for datasets, appears to support effective learning, understanding and use. The structure and set of entities of this ML API support usage with minimal amount of code and context, trial-and-error exploration, easy refactoring and easy adaptation to custom user needs. This facilitates opportunistic development approaches, which are driven by design and rapid experimentation goals, and prone to happen in contexts of learning and creative and music technology development.

The CDs framework opens up interesting perspectives of analysis that support a rich and deep discussion about ML API design. However, we faced some challenges in the general application of the CDs, mostly related to communication and interpretation issues with the CDs vocabulary, and validity and reliability issues, which typically occur in questionnaire and survey methods with small samples (Adams and Cox,

2008). Other challenges relate to the difficulty to establish a scale and rate a ML API for each cognitive dimension of analysis. We also found limitations to the CDs concerning the interactions of an ML API with other artifacts, such as the text-editor or programming environment where ML API integration code is programmed, or its documentation media. Although a CD assessment cannot lead to full usability validation (Dagit et al., 2006) of an ML API, it can lead to new insights which may trigger new design iterations and thus become a useful and pragmatic resource to ML API designers.

Future work includes avenues of research which build on the CDs and quantitative methods as pragmatic methodological tools for ML API and notation designers. One avenue is to investigate a more focused and formalizable set of dimensions, which may help to analyse the use of IML and ML APIs more adequately. Another avenue of research is to explore ways to augment the cognitive dimensions framework to account more holistically for a set of interdependent artifacts—including language notations, programming interfaces, development environments, documentation and other high-level aspects which Petre (2006) has identified. Finally, we are exploring new research tools for conducting online quantitative usability studies with ML APIs which may scale to more participants and provide more generalizable outcomes.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

ETHICS STATEMENT

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements. The patients/participants provided their written informed consent to participate in this study.

REFERENCES

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). “Tensorflow: a system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (Savannah, GA), 265–283.
- Adams, A., and Cox, A. L. (2008). “Questionnaires, in-depth interviews and focus groups,” in *Research Methods for Human Computer Interaction*, eds P. Cairns and A. L. Cox (Cambridge, UK: Cambridge University Press), 17–34.
- Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T. (2014). Power to the people: the role of humans in interactive machine learning. *AI Magaz.* 35, 105–120. doi: 10.1609/aimag.v35i4.2513
- Austin, C. A. (2005). *Renaissance: a functional shading language* (MSc thesis). Iowa State University, Ames, IA, United States.
- Bellingham, M., Holland, S., and Mulholland, P. (2014). “A cognitive dimensions analysis of interaction design for algorithmic composition software,” in *Proceedings of Psychology of Programming Interest Group Annual Conference* (Brighton, UK), 135–140.
- Bernardo, F., Grierson, M., and Fiebrink, R. (2018). User-centred design actions for lightweight evaluation of an interactive machine learning toolkit. *J. Sci. Technol. Arts* 10:2. doi: 10.7559/citarj.v10i2.509

AUTHOR CONTRIBUTIONS

FB the primary author, designed and led the study, and contributed to the design and development of the RAPID-MIX API. MZ was the lead designer and developer of the RAPID-MIX API. MG provided high-level research guidance and contributed to all aspects of study. RF contributed to the design of the questionnaire and provided critical guidance and supervision throughout all the stages of the study. All authors contributed to the production and review of this work and to the design of the RAPID-MIX API.

FUNDING

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 644862 and the UKRI/AHRC research grant Ref: AH/R002657/1.

ACKNOWLEDGMENTS

We would like to thank the participants of cognitive dimensions study for their time and effort. We would like to thank Steven Clarke from Microsoft VS Code and Visual Studio usability group for providing details about his API usability studies with the Cognitive Dimensions. Finally, we would like to thank the other RAPID-MIX consortium members with whom we had the pleasure to collaborate. They are Fred Bevilacqua, Xavier Boisserie, Andres Bucci, Fabian Gilles-Renn, Sergi Jordà, Joseph Larralde, Sebastian Mealla, Panos Papiotis, Adam Parkinson, Hugo Silva, Atau Tanaka, and Jean-Baptiste Thiebaut.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/frai.2020.00013/full#supplementary-material>

- Bernardo, F., Zbyszyński, M., Fiebrink, R., and Grierson, M. (2017). “Interactive machine learning for end-user innovation,” in *Proceedings of the Association for Advancement of Artificial Intelligence Symposium Series: Designing the User Experience of Machine Learning Systems* (Palo Alto, CA), 369–375.
- Bevilacqua, F., Müller, R., and Schnell, N. (2005). “MnM: a Max/MSP mapping toolbox,” in *Proceedings of the 2005 Conference on New Interfaces for Musical Expression* (Vancouver, BC: National University of Singapore), 85–88.
- Blackwell, A. F. (2002). “First steps in programming: a rationale for attention investment models,” in *Proceedings - IEEE 2002 Symposia on Human Centric Computing Languages and Environments, HCC 2002* (Washington, DC), 2–10.
- Blackwell, A. F., and Green, T. R. G. (2000). “A cognitive dimensions questionnaire optimised for users,” in *Proceedings of 12th Workshop of the Psychology of Programming Interest Group (PPiG)* (Cozenza), 137–154.
- Blackwell, A. F., Green, T. R. G., and Nunn, D. J. E. (2000). “Cognitive dimensions and musical notation systems,” in *ICMC* (Berlin).
- Buitinck, L., Louppe, G., and Blondel, M. (2013). API design for machine learning software: experiences from the scikit-learn project. *arXiv [Preprint]*. arXiv:1309.0238.
- Bullock, J., and Momeni, A. (2015). “ml.lib: robust, cross-platform, open-source machine learning for max and pure data,” in *NIME 2015 Proceedings* (Baton Rouge, LA), 3–8.

- Caramiaux, B., Montecchio, N., Tanaka, A., and Bevilacqua, F. (2014). Adaptive gesture recognition with variation estimation for interactive systems. *ACM Trans. Interact. Intell. Syst.* 4, 1–34. doi: 10.1145/2643204
- Clarke, S. (2007). “What is an end user software engineer?” in *Dagstuhl Seminar 07081: End-User Software Engineering*, eds M. Burnett, G. Engels, B. Myers, and G. Rothmel (Dagstuhl: Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik), 1–2.
- Clarke, S. (2010). “How Usable Are Your APIs?”, in *Making Software: What Really Works, and Why We Believe It*, eds A. Oram and G. Wilson (Sebastopol, CA: O’Reilly Media, Inc.), 545–565.
- Clarke, S., and Becker, C. (2003). “Using the cognitive dimensions framework to evaluate the usability of a class library,” in *Proceedings of the First Joint Conference of EASE and PPIG* (Keele), 359–366.
- Cwalina, K., and Abrams, B. (2009). *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable.NET Libraries*, 2nd Edn. Crawfordsville, IN: Addison-Wesley.
- Dagit, J., Lawrance, J., Neumann, C., Burnett, M., Metoyer, R., and Adams, S. (2006). Using cognitive dimensions: advice from the trenches. *J. Vis. Lang. Comput.* 17, 302–327. doi: 10.1016/j.jvlc.2006.04.006
- Dannenberg, R. B. (1985). “An on-line algorithm for real-time accompaniment,” in *Proceedings of the 1984 International Computer Music Conference* (Paris), 193–198.
- Diprose, J., MacDonald, B., Hosking, J., and Plimmer, B. (2017). Designing an API at an appropriate abstraction level for programming social robot applications. *J. Vis. Lang. Comput.* 39, 22–40. doi: 10.1016/j.jvlc.2016.07.005
- Edwards, W. K., Bellotti, V., Dey, A. K., and Newman, M. W. (2003). “Stuck in the middle: the challenges of user-centered design and evaluation for infrastructure,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI ’03* (FT Lauderdale, FL), 297–304.
- Fails, J. A., and Olsen, D. R. (2003). “Interactive machine learning,” in *Proceedings of the 8th International Conference on Intelligent User Interfaces IUI 03* (Miami, FL), 39–45.
- Fiebrink, R., Cook, P. R., and Trueman, D. (2011). “Human model evaluation in interactive supervised learning,” in *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems - CHI ’11* (Vancouver, BC), 147.
- Fowler, M. (2004). “Module assembly, [programming]” in *IEEE Software*, 21, 65–67. doi: 10.1109/MS.2004.1270764
- Francoise, J., Schnell, N., and Bevilacqua, F. (2013). “A multimodal probabilistic model for gesture-based control of sound synthesis,” in *Proceedings of the 21st ACM International Conference on Multimedia - MM ’13* (Barcelona), 705–708.
- Gillan, N., and Paradiso, J. A. (2014). The Gesture recognition toolkit. *J. Mach. Learn. Res.* 15, 3483–3487. doi: 10.13140/2.1.4216.2886
- Green, T. R. G. (1989). “Cognitive dimensions of notations,” in *People and Computers V*, eds A. Sutcliffe and L. Macaulay (Cambridge, UK: Cambridge University Press), 443–460.
- Green, T., and Petre, M. (1996). Usability analysis of visual programming environments: a cognitive dimensions’ framework. *J. Vis. Lang. Comput.* 7, 131–174. doi: 10.1006/jvlc.1996.0009
- Hall, M. A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explor.* 11, 10–18. doi: 10.1145/1656274.1656278
- Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S. R. (2007). “Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI’07* (San Jose, CA), 145.
- Henning, M. (2009). API design matters. *Commun. ACM* 52:46. doi: 10.1145/1506409.1506424
- Jialin, S., and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 1, 1–15. doi: 10.1109/TKDE.2009.191
- Katan, S., Grierson, M., and Fiebrink, R. (2015). “Using interactive machine learning to support interface development through workshops with 19 disabled people,” in *CHI ’15: Extended Abstracts on Human Factors in Computing Systems 2015* (Seoul).
- Kohavi, R., and Provost, F. (1998). Glossary of terms. Special issue on applications of machine learning and the knowledge discovery process. *Mach. Learn.* 30, 271–274.
- Kuhn, M. (2008). Building predictive models in R using the caret package. *J. Statistical Software* 28, 1–26. doi: 10.18637/jss.v028.i05
- Lieberman, H., Paternó, F., Klann, M., and Wulf, V. (2006). End-user development: an emerging paradigm. *End User Dev.* 9, 1–8. doi: 10.1007/1-4020-5386-X
- Logan, B. (2000). “Mel frequency cepstral coefficients for music modeling,” in *ISMIR*, Vol. 112 (Plymouth, MA), 211–212.
- Mellis, D. A., Zhang, B., and Leung, A. (2017). Machine learning for makers: interactive sensor data classification based on augmented code examples. 2, 1213–1225. doi: 10.1145/3064663.3064735
- Meng, M., Steinhardt, S., and Schubert, A. (2019). How developers use API documentation: an observation study how developers use API documentation: an observation study. *Commun. Design Q* 7, 40–49. doi: 10.1145/3358931.3358937
- Miranda, E. R., and Wanderley, M. M. (2008). *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*, Vol. 32. Middleton, WI: A-R Editions, Inc.
- Myers, B. A., and Stylos, J. (2016). Improving API usability. *Commun. ACM* 59, 62–69. doi: 10.1145/2896587
- Nash, C. (2014). “Manhattan: end-user programming for music,” in *Proceedings of the International Conference on New Interfaces for Musical Expression* (London, UK), 221–226.
- Norman, D. A. (2013). *The Design of Everyday Things*. New York, NY: MIT Press.
- Petre, M. (2006). Cognitive dimensions ‘beyond the notation’. *J. Vis. Lang. Comput.* 17, 292–301. doi: 10.1016/j.jvlc.2006.04.003
- Roberts, A., Hawthorne, C., and Simon, I. (2018). “Magenta.js: a JavaScript API for augmenting creativity with deep learning,” in *Proceedings of the 35th International Conference on Machine Learning* (Stockholm), 2–4.
- Robillard, M. P., and Deline, R. (2011). A field study of API learning obstacles. *Empir. Softw. Eng.* 16, 703–732. doi: 10.1007/s10664-010-9150-8
- Salvador, S., and Chan, P. (2007). FastDTW: toward accurate dynamic time warping in linear time and space. *Intellig. Data Anal.* 11, 561–580. doi: 10.5555/1367985.1367993
- Scheller, T., and Kühn, E. (2015). Automated measurement of API usability: The API concepts framework. *Inform. Softw. Technol.* 61, 145–162. doi: 10.1016/j.infsof.2015.01.009
- Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., et al. (2019). “TensorFlow.js: machine learning for the web and beyond,” in *Proceedings of the 2nd SysML Conference* (Palo Alto, CA).
- Tulach, J. (2008). *Practical API Design: Confessions of a Java™ Framework Architect*. New York, NY: Apress.
- Watson, R. (2014). “Applying the cognitive dimensions of API usability to improve API documentation planning,” in *Proceedings of the 32nd ACM International Conference on The Design of Communication CD-ROM* (New York, NY), 2–3.
- Wiegars, K. E. (2002). *Humanizing Peer Reviews*. Technical Report.
- Wijayarathna, C., Arachchilage, N., and Slay, J. (2017). A “Generic cognitive dimensions questionnaire to evaluate the usability of security APIs,” in *Human Aspects of Information Security, Privacy and Trust. HAS 2017. Lecture Notes in Computer Science*, Vol. 10292, ed T. Tryfonas (Cham: Springer), 160–173.
- Zakai, A. (2011). “Emscripten: An LLVM-to-JavaScript Compiler,” in *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion* (Portland, OR: ACM).
- Zbyszyński, M., Grierson, M., Yee-king, M., and Fedden, L. (2017). “Write once run anywhere revisited: machine learning and audio tools in the browser with C++ and emscripten,” in *Web Audio Conference 2017* (London, UK: Centre for Digital Music; Queen Mary University of London), 1–5.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Bernardo, Zbyszyński, Grierson and Fiebrink. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



A Dynamic Representation Solution for Machine Learning-Aided Performance Technology

Jason Palamara^{1*} and W. Scott Deal²

¹ Department of Music and Arts Technology, Indiana University-Purdue University Indianapolis, Indianapolis, IN, United States, ² Donald Tavel Arts and Technology Research Center, Department of Music and Arts Technology, Indiana University-Purdue University Indianapolis, Indianapolis, IN, United States

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Bowei Chen,
University of Glasgow,
United Kingdom
Fabio Aurelio D'Asaro,
University of Naples Federico II, Italy

*Correspondence:

Jason Palamara
japalama@iu.edu

Specialty section:

This article was submitted to
Machine Learning and Artificial
Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 30 October 2019

Accepted: 03 April 2020

Published: 08 May 2020

Citation:

Palamara J and Deal WS (2020) A
Dynamic Representation Solution for
Machine Learning-Aided Performance
Technology. *Front. Artif. Intell.* 3:29.
doi: 10.3389/frai.2020.00029

This paper illuminates some root causes of confusion about dynamic representation in music technology and introduces a system that addresses this problem to provide context-dependent dynamics for machine learning-aided performance. While terms used for dynamic representations like forte and mezzo-forte have been extant for centuries, the canon gives us no straight answer on how these terms must be applied to literal decibel ranges. The common conception that dynamic terms should be understood as context-dependent is ubiquitous and reasonably simple for most human musicians to grasp. This logic breaks down when applied to digital music technologies. At a fundamental level, these technologies define all musical parameters using discrete numbers, rather than with continuous data, making it impossible for these technologies to make context-dependent decisions. The authors give examples in which this lack of contextual inputs in music technology often leads musicians, composers, and producers to ignore dynamics altogether as a concern in their given practice. The authors then present a system that uses an adaptive process to maximize its ability to hear relevant audio events, and which establishes its own definition for context-dependent dynamics for situations involving music technologies. The authors also describe a generative program that uses these context-dependent dynamic systems in conjunction with a Markov model culled from a living performer-composer as a choice engine for new music improvisations.

Keywords: music and machine learning, music and AI, dynamic representation, machine learning aided performance, improvisation, Ableton Live, Max for Live, music technology

INTRODUCTION

As of this writing, music technologies (software and hardware) cannot perform relative dynamics, only absolute dynamics. If a given system is set to play a tone at 0 dBFS, it will do so regardless of context. Music technologists of many stripes, such as professional audio engineers, often adapt methods for handling this, for instance, how professional audio engineers use the faders on a mixer to adapt the incoming audio signals for a particular situation, given the particulars of the room, the number of people present, and many other factors. However, music technologies do not adapt themselves to different contexts natively, which often causes amateur or nascent users to make mistakes leading to many amplitude-related errors, such as feedback or the tendency to mix without dynamic contrast (“brickwalling”) (Devine, 2013).

Confusingly, there exists a dizzying preponderance of methods music technologists use to represent dynamic levels (dB Full Scale, dB Sound Pressure Level, MIDI velocities, to name a few) (Dannenberg, 1993). These various systems, helpful as they are in many respects, thus inveigle one of the four fundamental properties of musical sound (volume, pitch, duration, and timbre) in a haze of pseudo-scientific mystery. The recent development of the LUFs and LKFS (the EBU R128 Standards, released in 2014) scales may do much to alleviate the preponderance of complaints against the rising levels of loudness (where the amplitude level of one piece of music is compared to the next), but this scale will not fix a lack dynamic contrast within a piece of music as it is being composed, mixed, or performed via improvisational or generative technologies.

One might think that the system of dynamic representation that has been with us for centuries would have been definitively codified long ago, but as illuminated by Blake Patterson, many musicians don't follow a composer's intent when they play to any appreciable degree (Patterson, 1974). Moreover, much anecdotal evidence suggests that this unfamiliarity with the various systems for representing dynamic levels may result in a general dismissal of the importance of one of the four fundamental parameters of sound. As noted by Kyle Devine in his article *Imperfect Sound Forever: loudness wars, listening formations and the history of sound reproduction*, quite often, the lack of dynamics in modern music has more to do with market-driven forces and personal taste than with user's technological naiveté (Devine, 2013).

Matthias Thiemel goes to admirable length to explain that, concerning acoustic music makers, dynamics have always been a fundamental parameter with which musicians "create meaning and structure" (Thiemel, 2001). However, he also goes to great length to explain that the history of the concept in music is one that eschews a literal understanding of exact loudness levels, in favor of an ever-adapting definition, which must change according to the whims and predilections of the time in which a composition is conceived. Thus, as he explains, the dynamic fortissimo might mean one thing in an early piece of Beethoven but means something completely different in a later piece by the same composer. That dynamics must be understood in the context of the composition and composer who wrote them is largely understood by professional performing musicians and musicologists, and this definition changes not only from instrument to instrument, from piece to piece, and even depends on the range of the instrument in question or the venue in which the dynamic in question is to be played.

A professional trumpet player, upon encountering the dynamic forte in the midst of an opera score of Puccini, may be capable of calculating the required dynamic using some internal concatenation of variables including the composer's intention, the conductor's most recent indication, the range of the given note, the size of the hall, and the probability of accidentally overpowering the ensemble even though no "solo" was called for in the score. The number of variables occurring to the player in question will vary greatly due to a great many factors, i.e., the maturity and experience level of the player, the cultural setting, however, perhaps this short list will hopefully illustrate the sheer number of factors involved in such a decision, which in this case

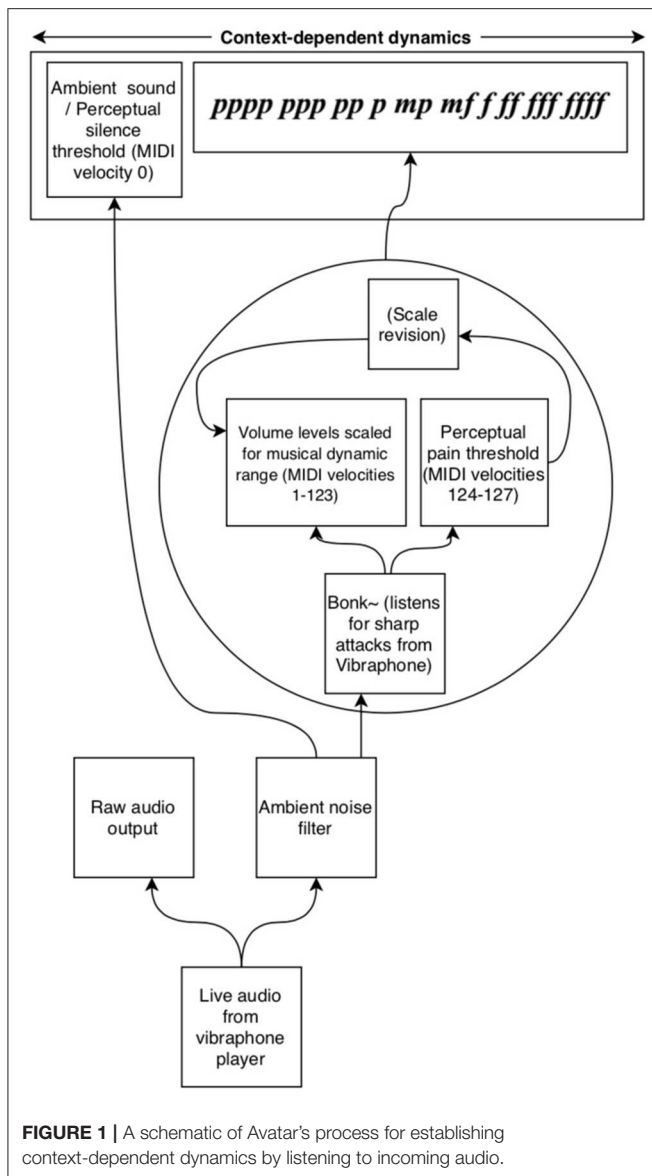
might result in the player in question playing the aforementioned note with a measurable dynamic level of 82 dB. In another setting, the same player might play the same excerpt at 65 or 90 dB. Dynamics would seem to be "all relative."

For music technology, however, this malleable understanding of musical dynamics presents a sizable problem. For instance, if one was tasked with transcribing the composer's handwritten score into one of the many notation programs currently available (Finale, in this case), the system would automatically assume that the forte marking in question corresponds to a certain MIDI velocity, which, when played via a digital instrument, will have the same results every time the user hits the space bar. In the case of Finale, a dynamic marking of forte corresponds to a MIDI velocity of 88 (out of 128 possible values, from 0 to 127). When this velocity level reaches the digital instrument, the dynamic will be converted into a loudness level, which is easily definable as 88/128ths of the instrument's total volume. Every time this instrument plays this dynamic level, the same volume level will be called upon to playback, no matter the context. If a composer wants to prepare a "fixed media" score or part for the aforementioned trumpet player, either there will need to be another performer who will manage dynamics for the fixed media part to provide context-appropriate dynamic choices or the composer will tend to avoid large dynamic contrasts altogether.

METHOD

To address the issues above, the authors here present a network of interconnected programs, collectively called *Avatar*, which may begin to fill the gap between musical technologies and context-dependent musical dynamics (**Figure 1**). The system has been designed trained to listen for a specific timbre (the vibraphone), filtering out noise, and non-intentional sound. This system then uses incoming amplitude levels to establish an adaptive perceptual framework for two key musical perception concepts, silence and the pain threshold. Finally, the system provides context-dependent MIDI velocities and musical dynamic representations of the audio it is hearing. These dynamics can then be used by generative music systems to play along and inform musical choices with context-dependent volume levels. In performance, this system follows the dynamics of the human vibraphone player, as a human collaborator would. The current context-dependent dynamic system is composed of two programs, *sig2~* (**Figure 2**), and *dyna* (**Figure 3**), and has been written in the Max-for-Live language, to facilitate use by music technologists in live performance using Ableton Live.

A third program, the *AvatarPlayer* (**Figure 4**), which will be discussed toward the end of this section, makes use of a pitch transition Markov model, culled from performances by a living composer-improviser. This program takes in messages from *sig2~* and *dyna* to generate context-dependent musical choices as it plays along with live vibraphone input. A fourth program, the *AvatarMachineLoader* (**Figure 5**), has been developed and used by the authors to create a database of Markov transitions that are used by the *AvatarPlayer* to generate new music.



Aside from the standard objects found natively in the Max language, these programs also make use of a number of external (added) Max objects. Most notably among these are a number of external Max patches developed by the authors with the prefix *HIMI* (Human Inclusive Musical Intelligence, a tangentially related project), the *bonk~* object (from Puckette, Apel, and Zicarelli, as revised by Böhm), the *ml.markov* object (from Benjamin Smith's *ml.** machine learning for Max external package; Smith and Deal, 2014), and a number of utility objects from Karlheinz Essl's *RTC-lib* package of externals (Essl, 1988). With the exception of the objects from the *HIMI* library, which will be included in the commercial release, these external objects are open source or commonly available via Cycling "74's Max Package manager. To produce MIDI files from audio recordings, the authors also use the *Onsets and Frames* audio-to-MIDI converter from Google Magenta (Hawthorne et al.,

2017; Dinculescu et al., 2019), which may be implemented via JavaScript, Python, or used online, and Ableton Live's three built-in audio-to-MIDI converters.

THE IMPORTANCE OF PERCEPTUAL FRAMEWORKS, SILENCE, AND PAIN

The system we present begins by establishing some fundamental perceptual frameworks extant within a human's musical understanding (Buettner, 1981), but which have been largely absent from the world of musical technologies. Though John Cage is correct in asserting that there is "no such thing as silence" (Cage, 1992), as argued by Elizabeth Hellmuth Margulis, the perception of silence is fundamental to our appreciation of music in general (Margulis, 2007). Cage's arguments aside, human beings appreciate music not by taking in a stream of audio and giving attention to the loudest elements, but by framing music as what happens between a conceptual understanding of silence (here defined as ambient sounds, incidental noises, and unintentional sound, which does not pertain to the music presented) and sounds that occur beyond a loosely defined perceptual pain threshold. While the technical human pain threshold corresponds to volume levels over 120 dB SPL, many listeners establish a more personal definition that most likely includes any intensity over 90 dB SPL in most contexts (Smith, 1970). As Margulis states, silences "facilitate processing by chunking the [musical] stream into units whose elements pertain to one another and should be understood, evaluated, and remembered together, by allowing time for the listener to synthesize and reflect on the chunk that has just passed" (Margulis, p. 5). Conversely, the perceived pain threshold provokes the listener to avoidance, covering, or protecting their ears, and in extreme cases, leaving a venue while the music is still happening. These concepts are the fundamental boundaries of human music-making, beyond which a musical performance is apprehended as "too quiet," "too loud," or "painful" (Fisher, 1929). For music technologies, without an understanding of these two fundamental concepts, there can be no context-dependent musical dynamics (Cope, 2004; Collins, 2012).

THE *SIG2~* PROGRAM

The *sig2~* program, the first link in the context-dependent dynamic system, begins by taking in raw audio, measuring the maximum and minimum levels of audio levels it encounters. In human musical performance, these definitions change over time, and so *sig2~* changes, accordingly, adapting its minimum (silence, noise floor), and maximum (pain threshold) throughout the performance as a human listener might.

The *HIMI.elimin8~* object (Figure 6), developed by the authors for a tangentially related project, is an adaptive filter algorithm that controls a noise gate, which is here useful for filtering ambient sound out of the incoming audio. It takes incoming audio for a short period of time (500 ms), measures its average strength, and gates the incoming signal accordingly, passing only signals stronger than the established noise bed

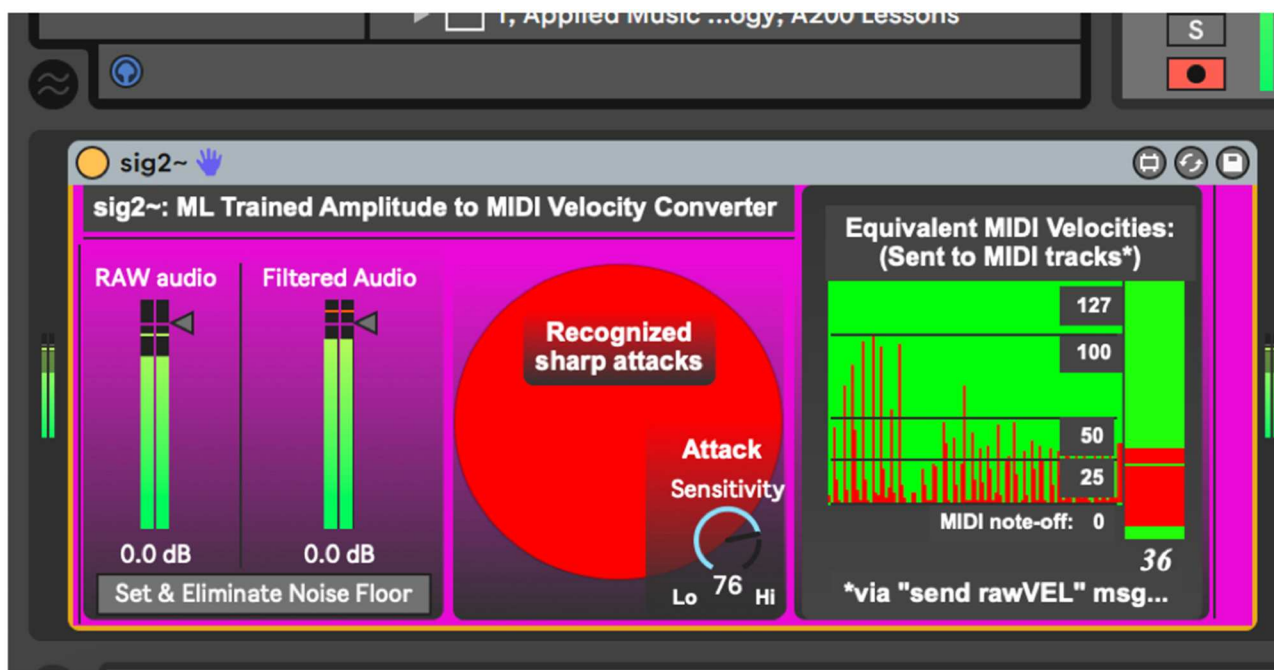


FIGURE 2 | Sig2~.amxd, a program that uses machine learning to establish a conceptual framework for silence and the pain threshold.



FIGURE 3 | Dyna.amxd, a program that translates MIDI velocities into various dynamic representation systems.

through (Figure 7). This process begins automatically once the device is loaded but can be manually reset if need be. As a human listener ignores the ambient hum of an air conditioning unit while trying to listen to live music, this system works best if the performer remains as quiet as possible during this setup

process so a noise bed definition can be made. After establishing this noise bed, the system is optimized to listen only for strong signals and establish its dynamics with anything below this noise bed defined as a non-musical event. As with any gating process, there is a danger here that if the gate is set too high, it will cut

out audio intended for the performance. If this occurs, the user has two options, increase the signal (feed the system a higher level of gain), or simply hit the “Set & Eliminate Noise Floor” button again.

When using reactive electronics in a live audio situation it is greatly advantageous to filter unwanted sounds, sustaining resonances, and incidental sounds so the system may listen specifically for the performer using the system, rather than trying to distinguish the performer’s sound from an undifferentiated stream of audio. Thus, *sig2~* uses *bonk~*, a common external Max object that may be trained to listen for specific timbres, to build a spectral template of the player’s audio (*bonk~* was developed by Miller Puckette, Theodore Apel, and David

Zicarelli, 64-bit version by Volker Böhm) (Puckette et al., 1998). This template can be saved or rewritten if the player is going to use the system again later. By clicking the “Train Timbre Recognition System” button (Figure 8), the user enables the *bonk~* object to listen for the timbre of the instrument you are playing. As the current project uses a vibraphone player as the listening target, the authors have created a timbre template for vibraphone, which is automatically read by *bonk~* upon loading the device. This system will operate with no training data but will simply listen for any incoming sharp attack rather than distinguishing a specific instrument from incidental noise.

Since *bonk~* parses an incoming audio stream into 11 frequency bands, any of which may recognize a frequency extant in the timbre it is hearing, it was useful to include a dial control that sets attack sensitivity (0–100). After repeated trials, it was discovered that *bonk~* best recognizes the timbre of the vibraphone above a sensitivity setting of 76. Similar trials would likely be necessary to train and perfect the process for another instrument’s timbre. Another purpose behind using the *bonk~* object is to identify sharp attacks and pass them on to the *dyna* device. Upon detecting a sharp attack, *sig2~* sends a message *wirelessly* to any related objects in the session via the “sharp” message.

The attack recognition “speed limit” default of 50 ms slows there cognition process to within human performance limits. As of this writing, few human musicians can play notes at a rate faster than 10 Hz (Martin and Martin, 1973). Jason Barnes, the world’s first true cyborg drummer, when wearing his robotic drummer prosthesis, can play at speeds up to 25 ms, hence the default speed limit caps the system’s use to provide for less glitchy playback (Weinberg and Driscoll, 2006). At this point, human perception comes into play as well, as, beyond 20 ms, human

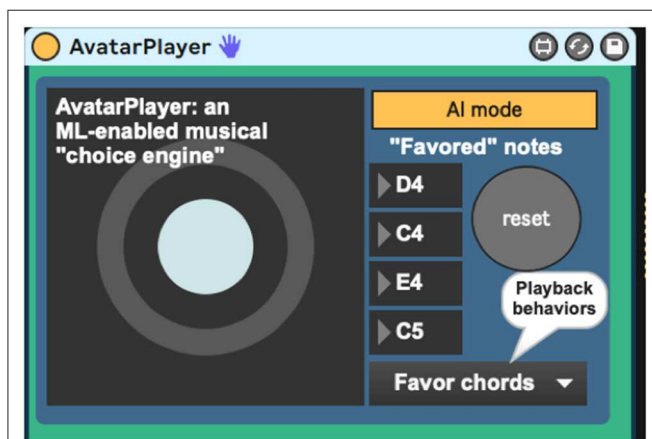


FIGURE 4 | AvatarPlayer.amxd, a program that uses machine learning to generate new musical choices in a given player’s style.

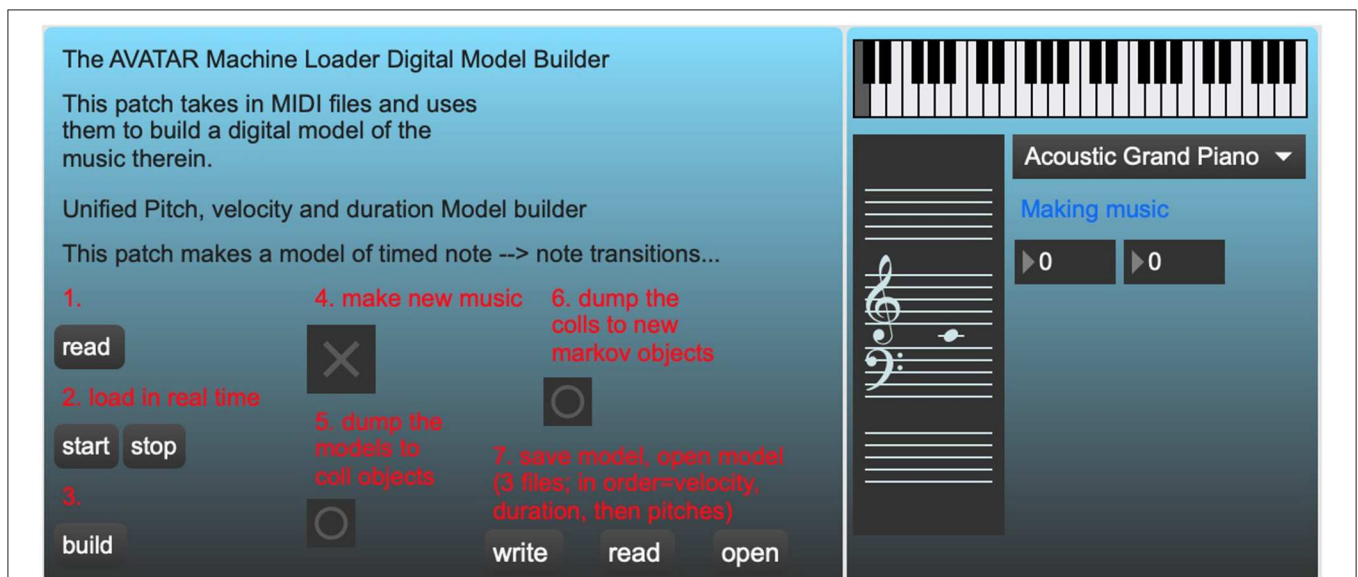


FIGURE 5 | AvatarMachineLoader.maxpat, a standalone Max program that builds a Markov model of pitch transitions from MIDI files.

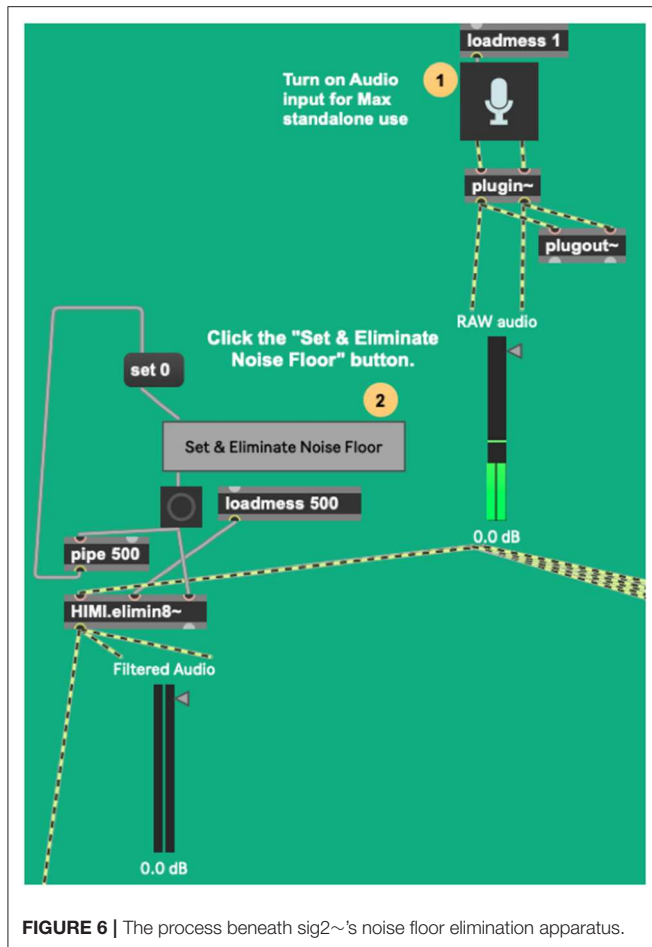


FIGURE 6 | The process beneath sig2~'s noise floor elimination apparatus.

beings find it hard to distinguish beats as distinct events, rather, perceiving them as connected events or waves.

Sig2~ uses another adaptive algorithm to establish a perceived pain threshold. Upon encountering strong audio levels, sig2~ again lowers its sensitivity using the *HIML.peakamp~* object. This process is rather simple but continually calculates an overall maximum peak amplitude from the audio it has encountered. Sig2~ then uses this maximum to scale incoming audio peaks to a MIDI-friendly 128 integer range (0–127). In this manner, if the system suddenly encounters a greater amplitude level than previously encountered, it simply adapts its scale to accommodate the new peak. The newly defined 128 integer MIDI velocity is sent out of the device via the output at the bottom and also “wirelessly” via the “rawVEL” send object. By establishing these perceptual frameworks and maintaining in real time, the sig2~ program can be relied on to convert incoming audio levels into MIDI velocities, giving any connected program a reliable dynamic context in an easy-to-use MIDI format.

THE DYNA PROGRAM

The *dyna* program is the second device that makes up the context-dependent dynamic system. *dyna* takes in the equivalent

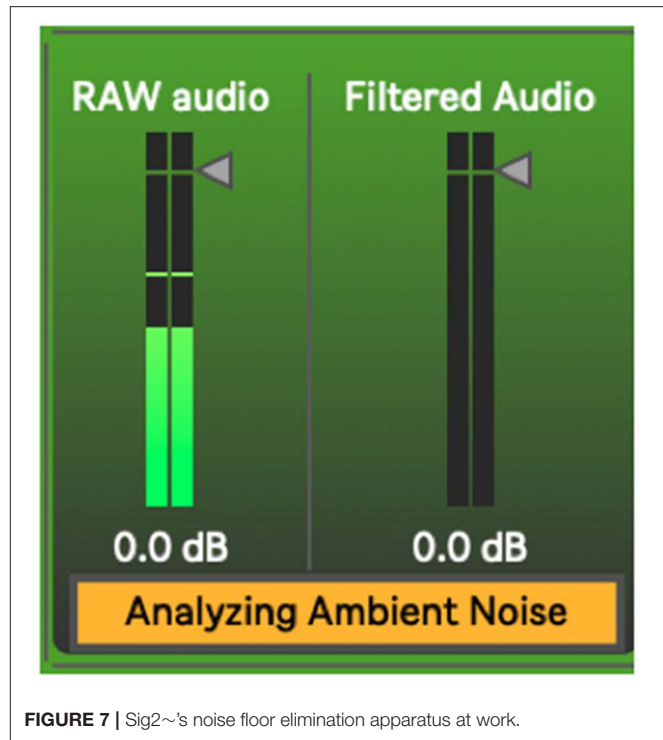


FIGURE 7 | Sig2~'s noise floor elimination apparatus at work.

MIDI velocities sent from the sig2~ device (via the “rawVEL” message) and defines dynamic ranges either derived from various popular music notation programs or as defined by the user. The sharp message, also sent by sig2~ acts as a trigger for the dynamic representations, giving the performer a context-dependent dynamic of their volume level as they play.

Inside the “sharpvelocitycalculation” subpatcher (Figure 9), *dyna* combines the incoming “sharp” message (indicating that a sharp attack has occurred) and the raw MIDI velocity of the most recent attack into a single message. One of the stickiest problem *dyna* addresses is the visual feedback to the performer, which needs to be extremely responsive (new higher dynamics may happen very quickly) but slow enough that the visual feedback system does not constantly “twitch” from dynamic to dynamic quicker than can be read by a human. To counteract this problem, this subpatcher makes use of another preexisting object called *HIML.waiter*, a Schmitt-Trigger delay that waits until all inputs have ceased before beginning a short delay and then sending a second bang. In the interim, if *HIML.waiter* receives further input, the delay is canceled, and the process is reset. *HIML.waiter* slows this process and allows the system to give preference to louder dynamics and ignore quick reflections at lower dynamics. This is modeled after human perception, which prefers louder dynamics, instead of perceiving quick reflections as reverberation (Doyle, 2004).

The actual dynamic definitions further illuminate the problems inherent in the dynamic representation used by many of today's most popular music software. As shown in Table 1, there appears little agreement regarding dynamic and their corresponding MIDI velocities among programs such as Finale,

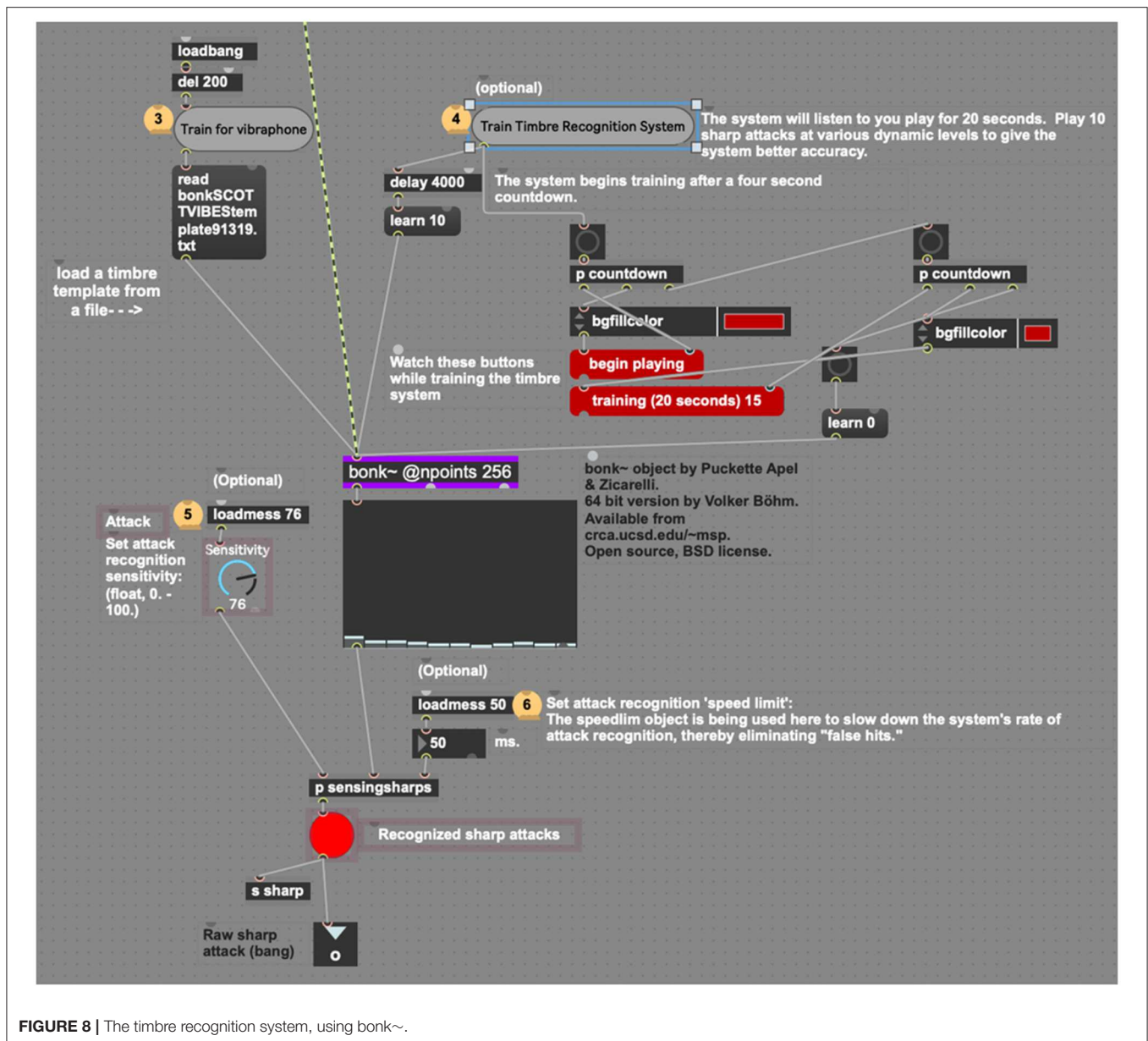


FIGURE 8 | The timbre recognition system, using bonk~.

Sibelius, MuseScore, and Dorico. Thus, with *dyna*, the authors have picked what seems to be the best solution to MIDI velocity dynamics currently on the market and also provided options for users to define dynamics in several other ways using the defaults from the major notation programs as presets.

Using the minimum and maximum established amplitudes from the processes described above, *dyna*'s representational default applies a logarithmic curve to the MIDI values between 0 and 127 taken from the Dorico notation program. Dorico's default dynamic scheme assigns these velocity values to the 10 most common dynamic values pppp, ppp, pp, p, mp, mf, f, ff, fff, and ffff. However, unlike the other leading notation programs, the developers of Dorico have wisely left room at the top of their dynamic range (velocities 124–127) to accommodate for what audio engineers might call headroom, but which the authors

here define as the volume above the perceived pain threshold. By default, incoming amplitude levels are thus evaluated by *dyna* based on these definitions. If situations or preferences call for changes to these definitions, the presets for other dynamic schemas are available via the menu at the bottom right of *dyna*'s user interface.

The user may also create unique dynamic presets (which are savable as Ableton Live.adg files for users of Max-for-Live). However, using the Dorico default has one interesting benefit in the case of the *dyna* program. Once the system has defined its context-dependent dynamics, the upper limit will remain stable unless the system suddenly encounters a greater dynamic. Thus, *dyna* can also be used as a pedagogical tool, as, a few minutes into a performance, greatly expanded dynamic peaks can be jarring to human listeners. In this case, a user employing *dyna* with the

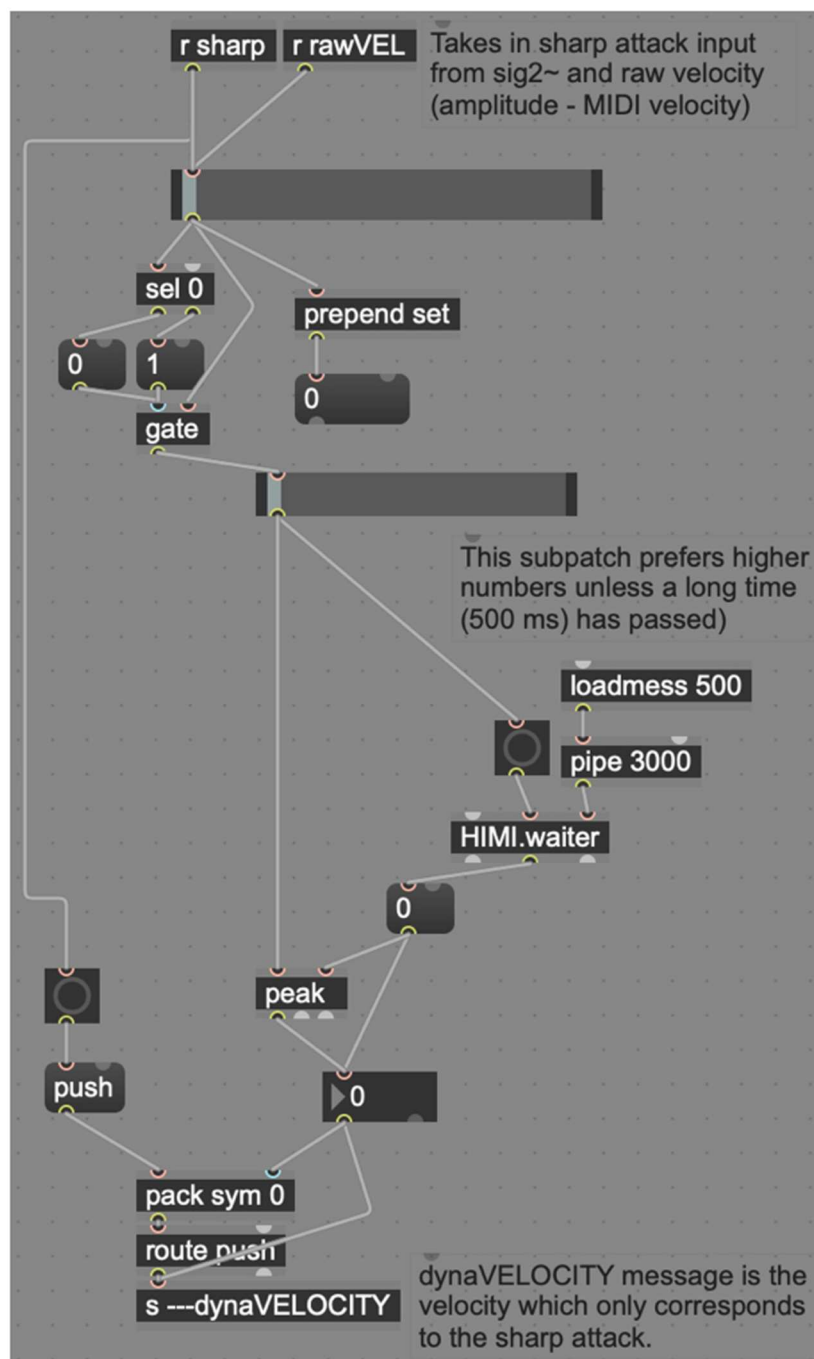


FIGURE 9 | Dyna's sharpvelocitycalculation patch.

default settings would see the system register the “pain” dynamic before resetting the maximum, thus alerting the performer to a dynamic unevenness they might not have discovered otherwise. Overall, once the context-dependent dynamic system is trained, it will listen to the player intelligently and report on the player's dynamic performance, allowing the player to better assess their dynamic proficiency vs. what is on the written page.

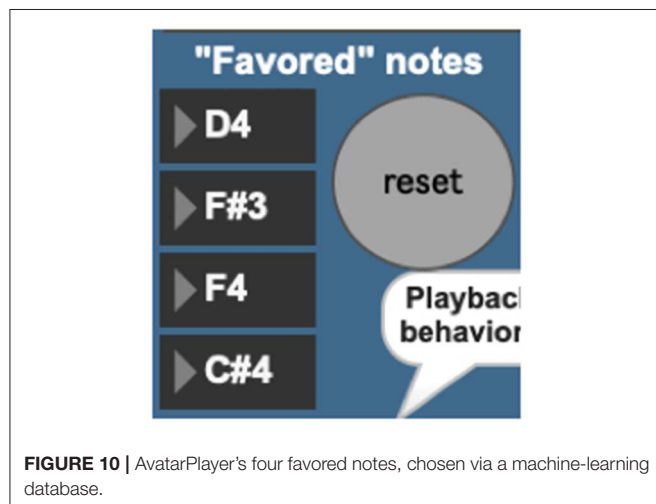
THE AVATARPLAYER PROGRAM

The *AvatarPlayer* program uses Markov note-to-note state transitions derived from a living composer-performer's improvised performances as a “choice engine,” providing a dynamically sensitive duet while listening to a live performance on the vibraphone. Using this system, the percussionist

TABLE 1 | A comparison of dynamic markings and corresponding MIDI velocities used by various notation programs.

| Musical dynamic | pppp | ppp | pp | p | mp | mf | f | ff | fff | ffff |
|--------------------------------------|------|-----|----|----|----|----|----|-----|-----|------|
| Finale MIDI velocities | 10 | 23 | 36 | 49 | 62 | 75 | 88 | 101 | 114 | 127 |
| MuseScore MIDI velocities | | 16 | 33 | 49 | 64 | 80 | 96 | 112 | 127 | |
| Dorico default MIDI velocities* | 5 | 14 | 25 | 46 | 61 | 77 | 89 | 101 | 119 | 123 |
| Dorico linear curve MIDI velocities* | 5 | 14 | 25 | 46 | 61 | 77 | 89 | 101 | 119 | 123 |
| Sibelius MIDI velocities | | 16 | 39 | 61 | 71 | 84 | 98 | 113 | 127 | |

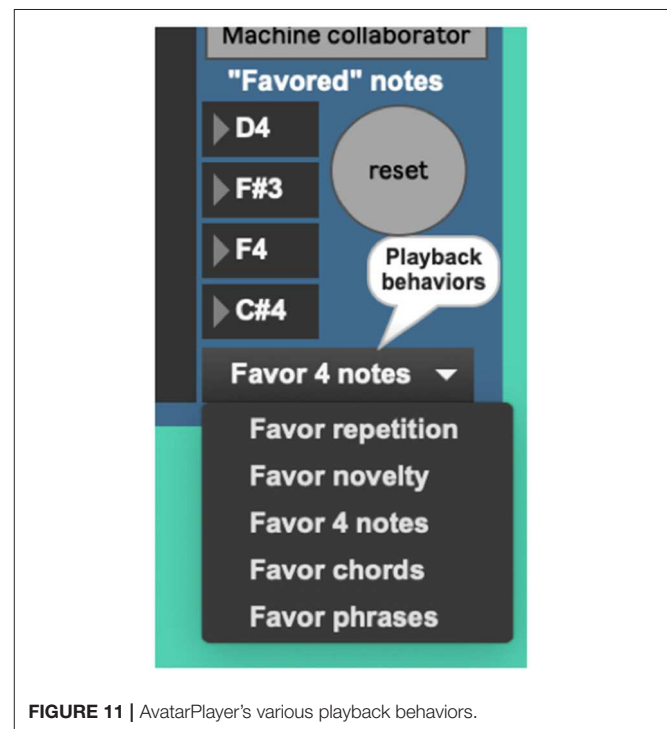
*These results were achieved using the default curve set to 2.5 and a linear setting of 1. Blacked-out areas represent dynamics not used by these programs.



improvises on the vibraphone while the system listens, playing when he is playing and stopping when he stops. When the *AvatarPlayer* plays, it generates novel pitch content based on the Markov model of the player's style, filtered through several algorithmic AI processes. While the Markov model provides statistical probabilities to drive note choice, these algorithmic AI behavior processes change the way these data are used and are patterned after real-life vibraphone improvisation techniques used by the percussionist model.

The *AvatarPlayer*'s use of the Markov model is currently governed by five playback behaviors. The first such behavior ("favor four notes") queries the Markov model for four notes to favor in its performance (Figure 10). Favoring these four chosen notes gives the note output a noticeable tonal centricity, a quality noticeably common in the live performer model's playing style. The behavior "Favor novelty" queries the Markov model for note-to-note transitions one by one, which creates a somewhat randomized, atonal quality. The behavior "Favor repetition" picks one note and favors repeating it two-out-of-three times. Cycling through these behavior modes (Figure 11) provides the system with a performance that sounds more human and less randomized, a common complaint against many generative music applications.

AvatarPlayer is also equipped with an autonomous AI mode, which makes musical accompaniment without listening to the



live input. This mode is often useful in sound checking and system tests and can even be MIDI-mapped in Ableton Live so as to be turned on and off during performance. Its MIDI output can be easily recorded, so as to generate new compositional material, but *Avatar*'s true purpose is as a collaborator using the sharp message from *sig2~* and the dynamic velocities from the *dyna* program to create a sensitive context-dependent musical collaborator that actively listens as a musician would in a duet.

To further invoke a blended human-machine cyborg aesthetic, *Avatar*'s developers have also created a number of high-quality Ableton Live instruments using the percussionist-model's own vibraphone as the sound source. In practice, the *Avatar* system listens to a human playing a vibraphone and plays along sensitively using an audible simulacrum of the vibraphone, behaving to a certain degree in the manner of the original human it is modeled after.

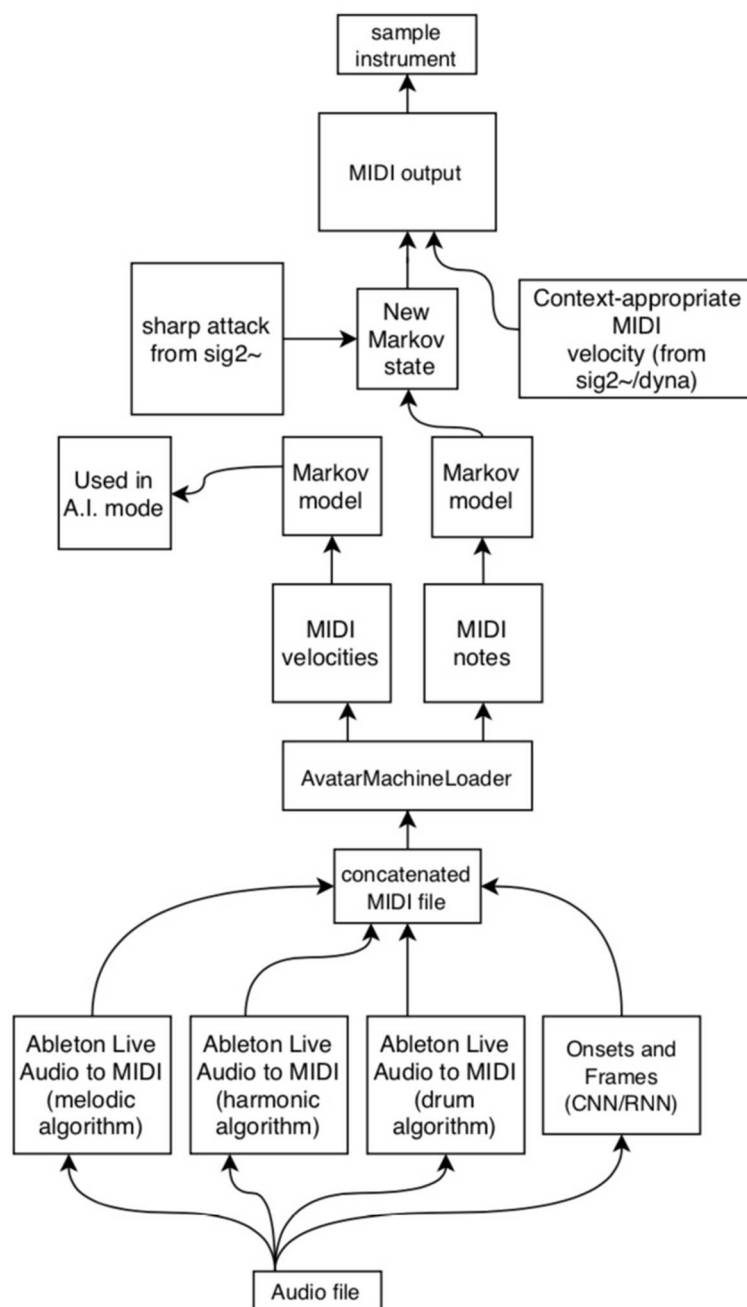


FIGURE 12 | A schematic of how various machine learning technologies are used to create a Markov state transition table for a given performance.

BUILDING THE MACHINE LEARNING MODEL: THE AVATARMACHINELOADER PROGRAM

As described above, the *AvatarPlayer* program makes use of a Markov model of pitch state transitions gathered from an analysis of real audio (**Figure 12**). The first hurdle in accomplishing this task is the transcription of a large number of audio files

into MIDI files that the *AvatarMachineLoader* program may analyze. Though there are a number of systems designed to do this currently available to the user, the best and most easily obtainable is the *Onsets and Frames* model released by researchers at Google Magenta. This model may be used online quite easily or implemented via open-source JavaScript or Python code (Hawthorne et al., 2017). *Onsets and Frames* achieves this level of accuracy by using two separate neural networks (a convolutional

neural network and a recurrent neural network) to detect pitch onsets even in polyphonically complex audio files.

As the *Onsets and Frames* model is designed at present to listen for piano timbres, rather than vibraphone, results are somewhat mixed. To improve accuracy of the eventual model, the authors also produced MIDI files using Ableton Live's built-in audio-to-MIDI transcription features, which use three different algorithms to assess pitch (focusing variously on melodic pitch movements, harmonic clusters, or rhythms). By painstakingly converting each audio recording using all four of these models, the authors hoped that the dataset (combined into a single multi-hour MIDI file) is significantly large enough to decrease the prominence of false positives and misapprehended pitches (Li et al., 2018). As these systems represent a sort of black box, it is difficult to say definitively whether the present model is an accurate representation of the original player. Accordingly, this procedure was repeatedly tested using MIDI files of music by gold standard composers like J.S. Bach and Vince Guaraldi. As new improvisations are supplied by the model composer-performer, the audio files are converted in these four ways and added to an ever-growing concatenated MIDI file.

Once the MIDI file dataset is compiled, the file must be passed through a Markov chain generator object (*ml.markov*) in order to produce a database of state transitions. The authors have achieved the best results setting this object to "order 4." This setting produces state transitions that take into account the previous three notes. Since the ultimate goal is to create a model for vibraphone performance, the developers thought it best to generate a Markov model that took into account standard vibraphone performance technique, which utilizes four mallets in two hands, meaning the performer is often commonly improvising using groups of four notes.

While the present *AvatarPlayer* program utilizes a database of only pitch-to-pitch state transitions, the *AvatarMachineLoader* program also uses additional *ml.markov* objects to generate state transitions for MIDI velocities, durations, and harmonies. These late additions will eventually make it possible to generate novel material that is patterned after a more complete model of the original living composer-performer's performance. After these models are built, they may be saved as a simple text file that can be loaded into the *AvatarPlayer*'s own *ml.markov* object. Once the *AvatarPlayer*'s model is loaded and built, a simple *bang* message will generate new MIDI pitches conforming to the model's state transitions.

RESULTS

The *Avatar* system, consisting of *sig2~*, *dyna*, and the *AvatarPlayer*, was successfully debuted at the Fata Morgana music and art festival in Indianapolis on October 3, 2019, by percussionist and Professor Scott Deal. The system performed admirably and has since been featured at the MusicaAcoustica festival in Beijing, China, on October 22, 2019. These performances were well-received, and it has

been reported that the system is rather easy to use and implement, even in the absence of the developers. A number of national and international performances for 2020 have already been scheduled.

It is often difficult to quantify the success or failure of musical experiments where the end result is a creative phenomenon. Such is the case with the present system in that the end result is not quantifiable data, but public performances using the system. As outlined above, the public performance record of this system is still in its infancy. While the recent results have all been promising, the pool of users will be extremely limited until such a time as the *Avatar* system is released commercially. As such, a commercial release date of February 28 has been announced, after which the system may be tested by the public and hard data may be collected and assessed.

A data-driven comparison of similarities between the living composer-performer upon which the system is modeled and the model itself would be a useful metric by which the system could be judged. The authors are presently beginning work on a future paper involving assessments of this type, which could be used as a model for assessments of future musical machine learning projects.

DISCUSSION, SCALABILITY, AND LIMITATIONS

Though various other technologies exist that purport to translate volume into MIDI velocity, results from this context-dependent system have been encouraging. The uses of more accurate instantaneous audio to MIDI transcription are many, as MIDI is the definitive control protocol underlying any music technology. It is hoped that in the near future, systems like *Onsets and Frames* may evolve into easily implementable real-time audio-to-MIDI. However, even with reliable real-time audio-to-MIDI, technologies will do little to cure the lack of context-based dynamics outlined above.

That said, the context-dependent dynamic system outlined above has many other potential applications. By filtering out excess noise and amplitude overages, this system could be adapted to control lighting and video effects, or to transcribe audio to dynamic notation in real time (perhaps as a plug-in for one of the notation programs mentioned above). The minimum and maximum definitions defined by *sig2~* could be used to automatically (and cheaply) mix audio channels in situations where a professional audio engineer is unavailable. Systems like Landr, which use machine learning to mix or master audio files, are already affecting the market. Perhaps a context-dependent system for dynamics could do similar things for the ensemble classroom. But still, the system's most exciting possibilities revolve around creating new music by artificial intelligence or in enhancing the performance capabilities of human musicians with technology (Rowe, 1992; Miller, 2003; Weinberg and Driscoll, 2006).

While the present system works reliably, much more work is on the horizon. At the outset, the authors sketched out a goal of

building a system that would build a machine learning model of what it hears in real time, save it, and update the model as more data became available. The present system does these things in achingly slow fashion, and not in real time. Another drawback of the lack of real-time adjustment has only become apparent after repeated use of the system. Users of the system have recently reported that since the present system does not save its definitions for the perceptual frameworks of silence and the pain threshold from session to session, *Avatar* seems to begin each session with a heightened sensitivity to loud sounds, and takes a significant amount of interaction with a player before the system is trained to react appropriately. It is hoped that along with the focus on making real-time machine learning models, the system will also eventually be able save these adaptations and remember them in future sessions. With the advent of very reliable audio-to-MIDI transfer via models like *Onsets and Frames*, the authors hope that it may be possible to make real-time machine learning a feature of the *Avatar* system in the near future.

The current focus on dynamics, while fundamental, presents only one important parameter of musical performance. To truly listen to music like a human, the system must also listen for, understand, and differentiate pitch, timbre, and duration. Beyond these low-level features, there are high-level features, like mood, emotion, tonal implications, and many others that provide much of the richness inherent in the best music of all genres. Adapting *Avatar* for durational perception, closely related to the concepts of dynamics and silence, has already begun in earnest. The machine learning tools currently extant within the Max environment (notably, the previously mentioned *ML** and the *ml.lib* externals package from Nick Gillian), while brilliantly developed, leave much to be desired in the way of easy connections to common musical practice. Creating Markov models for monophonic pitch-to-pitch transitions are useful and simple to build at present but doing the same for a given MIDI file's harmonic content or

articulative character requires a complete redesign of the system's inputs and outputs.

Another limitation of the project as it stands is the dependence on the *bonk~* object's timbre recognition capabilities, which could be enhanced greatly. At the very least, adding multiple timbral models to *Avatar* will allow the user to use instruments other than the vibraphone, greatly widening the user base. Once these systems are improved, it should not be difficult to provide the system with timbral recognition capabilities. The ultimate goal of the authors is to provide a truly intuitive program that listens, rather than having to be managed by a knowledgeable user. These and the many other goals of this team will take much time and considerable hard work, but the rewards of such an enterprise are well-worth the effort.

OTHER INFORMATION

Project Link: <http://tavellab.net/>

Operating system: Mac OSX 10+ / Windows 10+.

Programming language: Max standalone, also works as Max-for-Live device within Ableton Live.

Restrictions for non-academic use: none.

DATA AVAILABILITY STATEMENT

The data analyzed in this study is subject to the following licenses/restrictions: Creative Commons v 4.0, CC BY-SA (Attribution-ShareAlike). Requests to access these datasets should be directed to Jason Palamara/IUPUI (japalama@iu.edu).

AUTHOR CONTRIBUTIONS

JP wrote the manuscript and all code. WD provided feedback for the software, beta testing the software, and providing the original musical performances upon which the ML database was based.

REFERENCES

- Buettner, S. (1981). Cage. *Int. Rev. Aesthet. Sociol. Music* 12, 141–151. doi: 10.2307/836557
- Cage, J. (1992). *Silence*. Frankfurt: Suhrkamp.
- Collins, N. (2012). Automatic composition of electroacoustic art music utilizing machine listening. *Comput. Music J.* 36, 8–23. doi: 10.1162/COMJ_a_00135
- Cope, D. (2004). A musical learning algorithm. *Comput. Music J.* 28, 12–27. doi: 10.1162/0148926041790685
- Dannenberg, R. (1993). Music Representation Issues, Techniques, and Systems. *Computer Music J.* 17, 20–30. doi: 10.2307/3680940
- Devine, K. (2013). Imperfect sound forever: Loudness wars, listening formations and the history of sound reproduction. *Popular Music* 32, 159–176. doi: 10.1017/S0261143013000032
- Dinculescu, M., Engel, J., and Roberts, A. (2019). “MidiMe: Personalizing a MusicVAE Model with User Data,” in *Workshop on Machine Learning for Creativity and Design* (Vancouver, BC: NeurIPS).
- Doyle, P. (2004). From ‘My Blue Heaven’ to ‘Race with the Devil’: echo, reverb and (dis)ordered space in early popular music recording. *Popular Music* 23, 31–49. doi: 10.1017/S0261143004000042
- Essl, K. (1988). Real time composition : komponieren mit computerunterstützung. *Positionen Texte Zur Aktuellen Musik* 1, 15–19.
- Fisher, W. (1929). What is music? *Musik. Q.* 15, 360–370. doi: 10.1093/mq/XV.3.360
- Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., et al. (2017). Onsets and frames: dual-objective piano transcription. *arXiv [Preprint]*. arXiv:1710.11153. doi: 10.5281/zenodo.1492341
- Li, T., Choi, M., Fu, K., and Lin, L. (2018). Music sequence prediction with mixture hidden markov models. *arXiv [Preprint]*. arXiv:1809.00842. doi: 10.1109/BigData47090.2019.9005695
- Margulis, E. (2007). Moved by nothing: listening to musical silence. *J. Music Theory* 51, 245–276. doi: 10.1215/00222909-2009-003
- Martin, J., and Martin, D. (1973). Auditory perception. *Br. Med. J.* 2, 459–461. doi: 10.1136/bmj.2.5864.459
- Miller, S. (2003). *Computer Music Journal*, 27, 89–92. doi: 10.1162/comj.2003.27.1.89
- Patterson, B. (1974). Musical dynamics. *Sci. Am.* 231, 78–95. doi: 10.1038/scientificamerican1174-78
- Puckette, M., Apel, T., and Zicarelli, D. (1998). “Real-time audio analysis tools for Pd and MSP,” in *Proceedings of International Computer Music Conference*. (San Francisco, CA: International Computer Music Association), 109–112.

- Rowe, R. (1992). Machine listening and composing with cypher. *Comput. Music J.* 16, 43–63. doi: 10.2307/3680494
- Smith, B. D., and Deal, W. S. (2014). “ML.* Machine learning library as a musical partner in the computer-acoustic composition flight,” in *Proceedings of the 2014 International Computer Music Conference* (Michigan: ICMA), 1285–1289.
- Smith, L. (1970). Noise as a Pollutant. *Can. J. Public Health* 61, 475–480.
- Thiernel, M. (2001). *Dynamics*. Grove Music Online. Available online at: <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000008458>. (accessed October 30, 2019).
- Weinberg, G., and Driscoll, S. (2006). Toward robotic musicianship. *Comput. Music J.* 30, 28–45. doi: 10.1162/comj.2006.30.4.28
- Conflict of Interest:** The authors declare the following financial interest: this software has been commercially released for a nominal fee through <http://tavellab.net/>.

Copyright © 2020 Palamara and Deal. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Automated Page Turner for Musicians

André Tabone, Alexandra Bonnici* and Stefania Cristina

Department of Systems and Control Engineering, University of Malta, Msida, Malta

OPEN ACCESS

Edited by:

Sriram Natarajan,
The University of Texas at Dallas,
United States

Reviewed by:

Rinkaj Goyal,
Guru Gobind Singh Indraprastha
University, India
Steven J. Simske,
Colorado State University,
United States

*Correspondence:

Alexandra Bonnici
alexandra.bonnici@um.edu.mt

Specialty section:

This article was submitted to
Machine Learning and Artificial
Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 01 November 2019

Accepted: 29 June 2020

Published: 11 August 2020

Citation:

Tabone A, Bonnici A and Cristina S
(2020) Automated Page Turner for
Musicians. *Front. Artif. Intell.* 3:57.
doi: 10.3389/frai.2020.00057

An increasing number of musicians are opting to use tablet devices instead of traditional print media for their music sheets since the digital medium offers the benefit of storing a lot of music in a compact space. The limited screen size of the tablet devices makes the music difficult to read and musicians often opt to display part of the music page at a time. With fewer music lines on display, the musician will then have to resort to scrolling through the music to read the entire score. This scrolling is annoying since the musicians will need to remove their hands from the instrument to interact with the tablet, causing a break in the music if this is not done quickly enough, or if the tablet is not sufficiently responsive. In this paper, we describe an alternative page turning system which automates the page turning event of the musician. By actively monitoring the musician's on-screen point of regard, the system retains the musician in the loop and thus, the page turns are attuned to the musician's position on the score. By analysing the way the musician's gaze changes between attention to the score and the instrument as well as the way musicians fixate on different parts of the score, we note that musicians often look away from the score and toward their hands, or elsewhere, when playing the instrument. As a result, the eye regions fall outside the field-of-view of the eye-gaze tracker, giving rise to erratic page-turns. To counteract this problem, we create a gaze prediction model that uses Kalman filtering to predict where the musician would be looking on the score. We evaluate our hands-free page turning system using 15 different piano songs containing different levels of difficulty, various repeats, and which also required playing in different registers on the piano, thus, evaluating the applicability of the page-turner under different conditions. Performance of the page-turner was quantified through the number of correct page turns, the number of delayed page turns, and the number of mistaken page turns. Of the 289 page turns involved in the experiment, 98.3% were successfully executed, 1.7% were delayed, while no mistaken page turns were observed.

Keywords: page-turning, eye-gaze tracking, Kalman filter, eye-hand span, half-page turns

1. INTRODUCTION

In this rapidly evolving world, digital media is taking precedence over the physical, printed form for information storage and presentation. Rather than printing books, these are instead being laid out on screens, and whole libraries can now be accessed from one's home or stored within a handheld device. These convenient changes have made it to the world of music. Musical scores are readily available as free, digital documents through digital libraries such as the IMSLP¹ or as purchasable

¹<https://imslp.org/>

PDF files from online stores. Digital sheet music offers musicians the advantages of availability and portability, compacting large volumes of works into a single, portable device (Laundry, 2011). Digital sheet music, however, introduces the problem of readability. Traditional, printed music uses the standard A4-size paper, where music players can expect sheet music to have stave heights of 7.5–8.5 mm (Nieweg and Vaught, 2011). The screen-size of regular digital tablets, however, does not permit the display of the entire page while retaining the same stave dimensions. Thus, musicians will either downscale the sheet such that it fits within the space available, or keep the desired size while panning/scrolling to see the entire score (Bell et al., 2005). The latter will require the musician to either pan the score or incur more frequent page turns in comparison to when using a printed score.

Page turns are annoying, requiring the player to momentarily release one hand from the instrument to make the turn. In high-quality music books, editors typeset the music such that the page turn coincides with a natural pause in the music, be it in the form of rests or notes of a longer duration (Laundry, 2011). However, this is not always possible, and musicians develop their particular method to overcome the annoyance of page-turning. Although there are various options to interact with the page on a tablet device, for example, through scrolling or tapping, these actions are not easily controllable when executed at speed. Thus, page-turning on a tablet device is no more comfortable than on print material.

Commercial software and hardware that address this problem exist. These solutions may fall under two categories, namely, manual or fully automated page-turners. Manual page-turning solutions require voluntary user input to trigger a page-turning event. For example, AirTurn² provides a foot pedal system which allows the music player to activate page turns through the use of an external foot-pedal device. While such an approach may be suitable in some cases, some instrumentalists require the use of their feet for their instrument foot pedals (Laundry, 2011). Thus, automated page-turning would be more desirable. Tablet applications such as MobileSheets³, SheetMusic⁴, PhonicScore⁵, and ClassicScore⁶ provide such a facility by employing a scrolling score, where the rate of the scroll is determined from the tempo of a pre-recording playback of the music in ClassicScore, which could be adjusted according to some preferred speed in MobileSheets and SheetMusic. Both these options are not ideal since the performer is required to adhere strictly to some specific tempo for the duration of the piece, which, often, results in a performance which is not stylistic. Applications such as PhonicScore allow the scrolling to adjust according to the musician's playing by taking into account real-time audio data. However, these methods are susceptible to background noise, the timbre of the instrument as well as note errors by the performer and are, therefore, not very reliable.

An ideal page-turning system would, therefore, be one which can operate without the use of additional gestures, that is, a system that functions on the already existing interactions between the musician and the score. In this manner, the musician can remain in control over when the page turn occurs while shifting the burden of the actual page turn onto the system controlling the music. Moreover, the page-turning system needs to be robust to errors that may potentially be introduced by the musician.

In our earlier work (Bonnici et al., 2017), we show how eye-gaze tracking can be used to monitor the musician's interaction with the score and thereby create a hands-free page turning system. This system was, however, limited to rigid head and eye movements due to the inherent noise that exists within eye-gaze tracking. In this paper, we extend this work by using a Kalman filter approach to model the musician's gaze interaction and hence provide a robust prediction of the musician's gaze location. We use this information together with a half-page turning system to ensure that the musician will have the current and the subsequent stave present on screen at all times.

2. RELATED WORK

Page-turning systems can be broadly categorized into two groups, namely applications for physical, printed books and applications for digital media, as shown in **Figure 1**. Systems that operate on physical books need to first engage with the top-most printed page. The device needs to lift this page from the remaining pages using mechanisms such as suction tubes, friction wheels, adhesive, or magnetic clips (Wolberg and Schipper, 2012). The page-turner then elevates the single page and transports it, face down, on the other stack of pages. Once turned, the device secures the sheet in place through some restraining mechanism to ensure that the loose sheet does not infringe on any further page-turning actions. Thus, mechanical systems need to balance the speed of turning the page with the relative fragility of the paper so as not to tear the paper (Wolberg and Schipper, 2012). Such mechanisms, therefore, tend to be relatively slow and are most often used in the context of page-turners for people with physical disabilities where the need outweighs speed and efficiency.

Page-turners based on digital media are more common in music applications. The reason for this stems from the increasing availability of tablet devices as well as digital sheet music. As shown in **Figure 1**, page-turners for digital media can be subdivided into two further categories, namely, those that are fully autonomous and those that depend on some form of user input. Fully autonomous systems rely on preset timing, scrolling through the music sheets at a fixed tempo (Bell et al., 2005). While these systems may allow for manual adjustments of the performing speed at the start of the performance, real-time adaptations to changes in speed are not possible. Thus, these systems are not adequate for musicians. Systems which depend on some user input can, once again, be divided into two categories, those that rely on active user input and those which utilize a passive user-input. Systems which

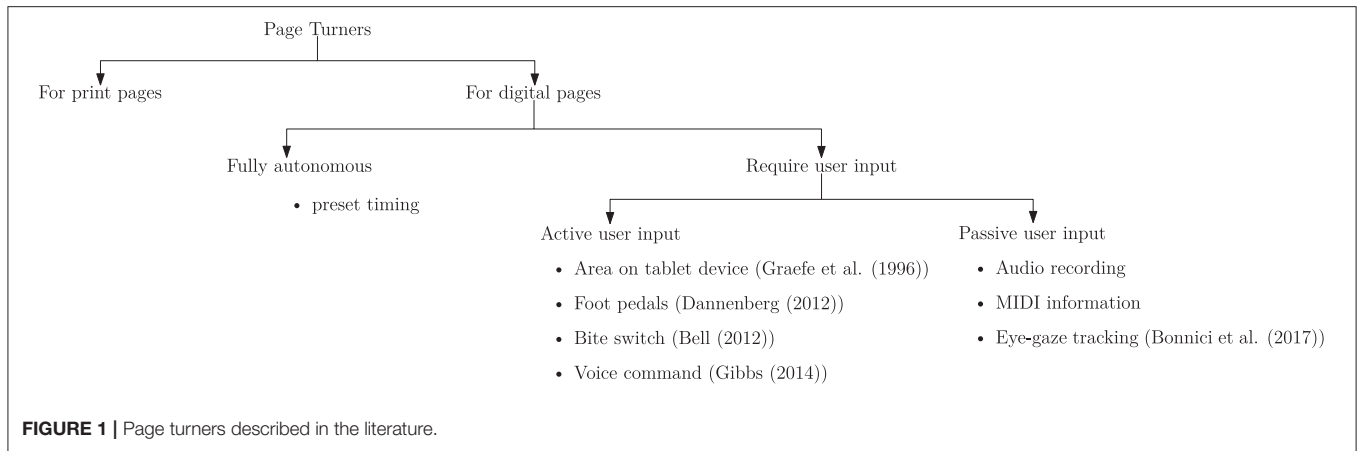
²<http://www.airturn.com/>

³<http://www.zubersoft.com/mobilesheets>

⁴<http://www.musicnotes.com/apps/>

⁵<http://phonicscore.com/>

⁶<http://blog.naver.com/earthcores>



require active user input require some action from the user to activate a page turn. This action can be in the form of tapping a foot pedal (Dannenberg, 2012) or an area on the tablet device (Graefe et al., 1996), bite-switches (Bell, 2012), or voice command triggers (Gibbs, 2014). While some of these techniques may be more effective than others, they still depend on the quick response of the device to the user response. The alternative approach of using passive user-input is, therefore, more attractive for the musician. These approaches involve tracking the musician's progress on the score and use this implicit user-input to determine when to activate a page turn. The tracking can be carried out through audio recordings, through MIDI information obtained from the instrument or by monitoring the users eye-gaze.

2.1. Score-Following Systems

When pianists perform a musical piece from a written score, they read the music notation and translate this information into the motor action needed to press the piano keys. The keying-action, in turn, activates the mechanisms that produce an audio signal. Page-turning may, therefore, utilize score-following based on eye-gaze tracking, the keying-action or the audio signal. Eye-gaze trackers typically provide the on-screen (x, y) coordinates corresponding to the pianists point-of-regard, and hence, the position on the score from which the pianist is currently playing. The keying action and the audio signal, on the other hand, provide information on the notes played. The data stream obtained from both keying-action and audio signal has a different format to the musical score and, therefore, requires alignment of the data to the score.

In Dorfer et al. (2016), this is carried out by training an end-to-end multimodal convolutional neural network (CNN). The score image S_i , consisting of one stave of sheet music, is quantized into overlapping *buckets* B_j . Likewise, the spectrogram of the corresponding audio signal is also divided into snippets $E_{i,j}$ of a fixed length of 12 s. The CNN is trained to match the rightmost onset in the spectrogram $E_{i,j}$ to the bucket B_j containing the corresponding note j . The resulting CNN model is then used to predict the expected location \hat{x}_j of an audio snippet with a target note j in the corresponding sheet music image.

This approach matches the spectrogram within ± 1 image bucket in 84% of the test cases. However, the method does not take into account that the music may have repeated patterns which would result in multiple matches between the audio extract and the score. Moreover, the approach also does not take into account the possibility that the performance may deviate from the written score. Such deviations can be intentional, for example, when the musician adds ornaments or chord embellishments not notated in the score. The musician can also introduce temporal changes within the music as a means of expression. These tempo changes would typically affect the estimation of the note onsets (Chen and Jang, 2019). Unintended changes to the performance are also possible, depending on the skill level of the performer. These errors may include incorrect keying of notes, repetitions to correct note errors or note omissions, resulting in jumps in the performed note sequences (Noto et al., 2019). As a result, the audio extracts may not necessarily have a direct match with the score image.

To correct for the possibility of repeated patterns, Dorfer et al. (2017) introduce temporal information through the use of Dynamic Time Warping (DTW). DTW computes the optimal non-linear alignment between two sequences, using a local cost measure that relates points from the two sequences to each other. In Dorfer et al. (2017), the two sequences comprise of the sheet music and the audio excerpts. A neural network is used to compute a local cost measure between the score sequence and the audio excerpts. The resulting cost matrix is then used by the DTW to align the sheet music and audio excerpts. However, the score-audio alignment is carried out offline. Thus, this approach is not suitable for page-turning applications, which requires real-time alignment of the two.

Chen and Jang (2019) propose an audio-score alignment process based on a similar approach. Note onsets are detected from the audio signal, extracting a feature vector to describe the signal around each onset. Finally, the feature vector is compared to the score using a dynamic programming approach, using a modified constant-Q transform as a measure of similarity. This measure allows for invariance to instrument timbre and overtone interference. To allow for the online score-following, Chen and Jang (2019) then modify the algorithm to reduce the

computational time required to align the audio to the score. To achieve this, they assume performance stability and performance continuity. That is, the musician adheres to the tempo marking on the score, and does not introduce sudden tempo changes. The musician is also expected to play the score sequentially and avoid skips or jumps to other sections of the score. These assumptions allow the onset matching algorithm to predict the location of onsets based on the tempo. They also limit the computation of concurrencies to onsets around the previously matched concurrencies. Chen and Jang (2019) achieve a mean latency of 19.2 ms obtained from 10 pre-recorded, human-played, four-part chorales composed by Bach.

While the results obtained in Chen and Jang (2019) do allow for real-time score following, they are based on assumptions of continuity and stability in the performance. These assumptions are valid for performances played when the musical piece has been mastered but do not necessarily hold during practice time when jumps and repetitions can be expected. While jumps and repetitions are difficult to predict through monitoring the audio signal alone, the eye-gaze information can provide invaluable insights on the point-of-regard of the pianist. It is, therefore, possible to deduce the position on the score from which the music is being played. Reading music has commonalities with the reading of linguistic texts, and thus, techniques for gaze tracking in linguistic texts may also apply to musical scores. Unlike linguistic texts, however, music does not have groupings based on fixed words. Instead, groupings are based on pitch structure, temporal structure, articulation, phrasing and orthographic conventions. The visual complexity of the musical score is, therefore, based on the decisions taken on all these levels (Huovinen et al., 2018). When reading, the grouping structures have an essential role in determining the eye movement, defining the duration of the fixations and the landing position of the next fixation.

Fixation points do not necessarily correspond to specific note symbols as long as they lie close enough to the symbol for this to be within the area of vision. This tolerance allows grouped structures, for example, quaver pairs or harmonic chords, to be treated with one single fixation (Puurttinen, 2018). In music reading, fixating on symbols ahead of the current playing position allow the musician to allocate sufficient time to process the symbols while keeping the general rhythmic characteristics of the music. In music reading, this is referred to as reading ahead and results in an eye-hand span. That is, the difference between the notes being played and the fixation point (Rosemann et al., 2016). Any salient difficulties spotted in the score will affect the timing of the saccades launched ahead. Upcoming symbols which appear to be less regular or non-typical will attract first fixations earlier in the musical performance. As a result, the eye-hand span may have local increases due to the musico-visually complex features of the notated score (Huovinen et al., 2018). Moreover, unexpected rhythmic or harmonic changes can locally decrease the eye-hand span (Penttinen et al., 2015; Rosemann et al., 2016).

It is also important to note that although pianists need to look at the score to read the music, they do not do so at all times. In a solo setting, glances to the keyboard are commonplace and help the pianist verify the correct hand position on the keys. Such glances to the keyboard are more common with

lower skill level, or when the music leaps through the keyboard registers (Cara, 2018). In ensemble playing, glances at partners are an essential way of communication between the ensemble members (Vandemoortele et al., 2018).

Noto et al. (2019) use Bayesian inference to estimate the pianist's position on the score using both eye-gaze and keying information, integrating the two sources into a single Bayesian inference by using a Gaussian mixture model. The keying and gaze data are modeled by Normal distributions whose parameters are adapted to each subject. The subjects are instructed to play a set extract without stopping or correcting any misplayed notes such that the keying and eye-gaze information can be easily aligned with the ground-truth. An exhaustive dynamic programming search is performed to find the best matching keying pattern from which the average and variance in the most likely matching position is obtained. Likewise, the eye-hand span is assumed to follow a normal distribution with mean (μ_{g_x}, μ_{g_y}) and variance ($\sigma_{g_x}, \sigma_{g_y}$) which are obtained by aligning the gaze data with the expected score position. By learning the eye-hand span distribution, the current gaze point (g_x, g_y) can be estimated. This estimate is then used in the Bayesian inference model to determine the most likely position for a match between the score and the keying data.

Similarly, Terasaki et al. (2017) also adopt a combined keying and eye-gaze tracking approach. However, Terasaki et al. (2017) use a Hidden Markov Model (HMM) to create a gaze model. The output probability of the HMM follows a normal distribution with the center coordinates of each note as the mean value. The model determines the initial transition probability and the state transition probability by learning the gaze position coordinates (g_x, g_y) of the gaze when performers are practicing while looking at the musical score. The output probability of the gaze model expresses the gaze likelihood, that is, the probability that the subject is looking at a particular place on the score. This probability score is reflected in the score following by multiplying the cost of the dynamic programming match with the gaze likelihood.

Both these approaches have been evaluated with single-line stave systems, and while Chen and Jang (2019) do take into account the possibility of loss in eye-gaze data, their approach simply waits for the eye-gaze data to become available once more. The two methods also make the general assumption that the eye-gaze will always move ahead. However, in the presence of two stave lines, as typical in piano music, the eye-gaze may also oscillate in the vertical direction. The eye-gaze may also shift backwards when the subject glances at the clef, key-signature and time-signature, particularly if these change within the piece, while at the same time, keying information remains moving forward. Moreover, the eye-hand span may require different local distributions, depending on the characteristics of the piece. Thus, more robust treatment of the eye-gaze information is required.

2.2. Displaying the Score

An automated page turning system must also take into account the way the music is displayed on screen and how the page turn is executed. Such a system must take into account the player's experience, allowing the pianist to, not only read the music with ease (Bell et al., 2005; Nieweg and Vaught, 2011) but also to

remain well aware of the context of the music they are playing. These considerations will restrict the amount of information that can be presented on the screen while exclude instantaneous jumps between sections of the music (Laundry, 2011). Several options for digital score visualizations have been proposed in the literature. The simplest method offers the presentation of sheet music as a continuous stream, either horizontally with the score scrolling across the width of the screen, or vertically with the score scrolling across the length of the screen. Such digital layouts, however, are not popular with music players since it is easier to lose track of the current position on the score (Bell et al., 2005). Alternative representations, where the score is kept static until a page turn activates overwriting old material with new have been proposed. Here, several visualizations are possible; for example, a two page system may be used with the page turn shifting the whole page to the left, such that the left hand page always displays the current score page to be played while the right hand page displays the next one (Graefe et al., 1996; Blinov, 2007). The screen size of a typical, portable digital tablet, however, does not allow for the display of two pages simultaneously without reducing the page size beyond what can be comfortably read by the music player. Alternative digital music systems which involve displaying a single page make use of the fact that the digital screen may be divided into two parts, allowing for split-page turning whereby, after some time delay, the top part of the page can display new content while the bottom part of the page retains the current content, before this too is updated. In order to indicate the change in content, visualizations such as page peeling, or highlight lines have been used (Bell et al., 2005; Blinov, 2007; Laundry, 2011).

Digital page turning systems must also take into account the display of music with repeated sections, particularly when these sections are long. Since digital displays divide the printed scores into sub-pages for a comfortable fit on the device display space, any such repeat instructions may require going back several pages, aggravating what is already an annoying problem. To resolve the problem, automated page turning can be combined with a system of bookmark annotations to allow the player to go back and forth in the document with greater ease (Jin, 2013). However, instantaneous jumps from page to page in the music are considered distracting to music players (Laundry, 2011). This supports the concept of a flattened score in which all repeats of the musical score are expanded (Jin, 2013). Such a flattened score may be obtained by representing the sheet music using a formal language representation through optical music recognition algorithms, allowing the flattened score to be checked for errors in the interpretation of the repeat instructions (Jin, 2013; Dannenberg et al., 2014; Ringwalt et al., 2015).

3. A KALMAN FILTER MODEL FOR EYE-GAZE PAGE TURNING

In this paper, we adopt a Kalman filter approach to create a robust eye-gaze tracking model that can smoothen the noisy eye-gaze data recorded from the eye-gaze tracker while compensating for loss of input due to glances away from the score as well as local variations in the eye-hand span. To model the eye-gaze

pattern across the screen, we assume that the score image has been pre-processed using the score processing steps described in Bonnici et al. (2017), that is, the page is sub-divided into sub-pages comprising of two systems, repeats have been flattened and a half-page turn is adopted. We also assume that each system is comprised of two staves as typical of piano music.

3.1. Reading Model

Music, like text, is read from left to right (Huovinen et al., 2018), such that the current position on the score may be expressed by the linear equation:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_k \quad (1)$$

where $\mathbf{x}_k = (x_k, y_k)'$ denotes the current location on the score from which the subject is reading at the instance k while $\Delta \mathbf{x}_k = (\delta x_k, \delta y_k)'$ denotes the displacement in the reading position. The horizontal component δx_k of the displacement vector depends on the reading velocity, that is, the local velocity with which the piece is being read which depends on the tempo of the piece as well as the local complexity of the score. Toward the end of the system, however, the horizontal reading position will revert to the start of the next system and is, therefore, a function of the width of the system. Thus, the horizontal displacement may be expressed as:

$$\delta x_k = \begin{cases} f(v) & \text{within the same system} \\ f(w) & \text{at the end of the system} \end{cases} \quad (2)$$

where v is the reading velocity and w the width of the system, as illustrated in **Figure 2**.

Since each system consists of two staves, let us, without loss of generality, assign the vertical component of the reading position to be at the middle of the system as illustrated in **Figure 2**. While the reading position remains within the same system, this vertical component is expected to remain unchanged. In the transition from one system to the next, this vertical component is expected to shift vertically as a function of the separation between the two systems. Thus, the vertical displacement component δy_k can be expressed as:

$$\delta y_k = \begin{cases} 0 & \text{within the same system} \\ f(s) & \text{at the end of the system} \end{cases} \quad (3)$$

where s is the separation between two systems as illustrated in **Figure 2**.

An eye-gaze tracker will provide information on the point of regard $\mathbf{g} = (g_x, g_y)$ of the subject on the screen. This point of regard corresponds to the subject's current reading position such that the point of regard \mathbf{g} may be used to adjust and update the reading position predicted through Equation (1). In particular, the point of regard may be used to estimate the local changes in the reading velocity, allowing for updates to both the reading position and the horizontal displacement δx_k . However, from literature on eye-gaze movement during music reading, we know that the eye movement may have variations in the vertical directions corresponding to the subject scanning both staves in the system. The eye-gaze movement will also have

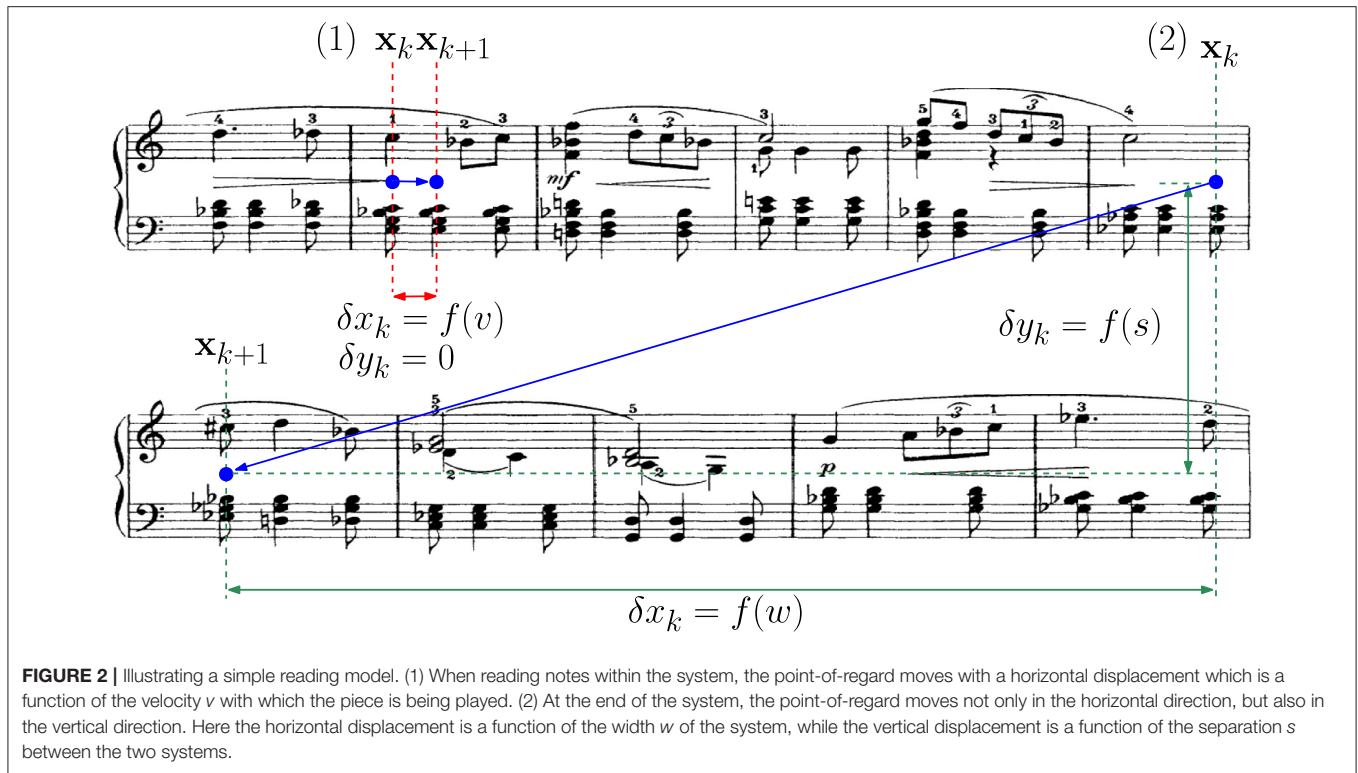


FIGURE 2 | Illustrating a simple reading model. (1) When reading notes within the system, the point-of-regard moves with a horizontal displacement which is a function of the velocity v with which the piece is being played. (2) At the end of the system, the point-of-regard moves not only in the horizontal direction, but also in the vertical direction. Here the horizontal displacement is a function of the width w of the system, while the vertical displacement is a function of the separation s between the two systems.

horizontal variations around the note being read as the reader shifts their gaze to read upcoming notes. Moreover, glances at keyboards, or partners results in eye-gazes that do not always correspond to the reading location on the score. Thus, the eye-gaze must be considered as a noisy measurement and a method that compensates for noisy data must be adopted.

3.2. The Kalman Filter

The discrete time Kalman filter provides such a tool. The Kalman filter assumes that a system is governed by a process modeled by the linear stochastic model given by Equation (4) for which a measurement \mathbf{z} may be related to the state vector \mathbf{x} using Equation (5) (Maybeck, 1979)

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u} + \mathbf{w}_{k-1} \quad (4)$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \quad (5)$$

where \mathbf{w} and \mathbf{v} are random variables representing the process and measurement noise, respectively. These are assumed to be white Gaussian noise processes with zero mean and covariance matrices Q and R , respectively. Matrix A relates the state at the previous time instant $k-1$ to the state at the current time instant k , matrix B relates the optional control input \mathbf{u} to the state and matrix H relates the state to the measurement \mathbf{z} .

The Kalman filter estimates the state inputs in two steps referred to as the prediction step and a correction step, such that feedback from the measurement \mathbf{z} is used to obtain better estimates of the state vector \mathbf{x} . The prediction step is used by the filter to make *a priori* predictions of the state and error covariance using the knowledge gained about the process up to the current

time instant. These are denoted as $\mathbf{x}_{k|k-1}$ and $P_{k|k-1}$, respectively, and are given by:

$$\mathbf{x}_{k|k-1} = A\mathbf{x}_{k-1|k-1} + B\mathbf{u}_{k-1|k-1} \quad (6)$$

$$P_{k|k-1} = AP_{k-1|k-1}A' + Q \quad (7)$$

The *a priori* state and error covariance estimates are then updated in the correction step which takes into account the most recent measurements obtained at the current time instant. The updated *a posteriori* estimates, denoted by $\mathbf{x}_{k|k}$ and $P_{k|k}$ are obtained through the correction update step:

$$K_k = P_{k|k-1}H'(HP_{k|k-1}H' + R)^{-1} \quad (8)$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + K_k(\mathbf{z}_k - H\mathbf{x}_{k|k-1}) \quad (9)$$

$$P_{k|k} = (I - K_kH)P_{k|k-1} \quad (10)$$

where K is a gain matrix which is estimated by the Kalman filter to minimize the *a posteriori* error covariance and weighs the difference between the predicted and actual measurements to update the *a priori* state estimate $\mathbf{x}_{k|k-1}$ to obtain the *a posteriori* state estimate $\mathbf{x}_{k|k}$ (Maybeck, 1979).

3.3. Application of the Kalman Filter Model for Page Turning Applications

Let us consider the pianist reading music from a single system. If we assume a reading model in which the reading velocity remains constant, then, the process model may be expressed as:

$$x_{k+1} = x_k + \delta x_k \quad (11)$$

$$y_{k+1} = y_k \quad (12)$$

$$\delta x_{k+1} = \delta x_k \quad (13)$$

Comparing this model to the process model defined by Equation (4), the reading position within the system may be modeled by a process model with a state vector $\mathbf{x} = (x, y, \delta x)'$ such that the matrix A is given by:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

with a zero control input. By considering the eye-gaze position $\mathbf{g} = (g_x, g_y)'$ as the noisy measurement \mathbf{z} we can define the matrix H as:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

We hypothesize that within the single system the Kalman filter correction step will provide the necessary correction to the state vector \mathbf{x} to allow for local adjustments in the reading velocity.

Let us now consider the instance when the pianist is transitioning from one system to the next. In section 3.1, we note that this transition requires an additional displacement in the reading position to initialize the reading position to the start of the subsequent system. There are various possibilities to take into account the transition between two systems. For example, if we assume that the transition between the systems is instantaneous, then the additional displacement required can be introduced through the control input \mathbf{u} . Alternatively, a switching Kalman filter (Murphy, 1998) may be employed to create two reading models, one to model reading within the system and another to model the transition between systems, switching between reading models. However, we hypothesize that for page turning applications, the Kalman filter model will be sufficiently quick in correcting for the reading position when the subject transitions between systems such that no additional inputs or reading models are required. In this manner, we balance accuracy of the reading model with speed and efficiency in the tracking.

3.4. Determining the Kalman Filter Parameters

To apply the Kalman filter model, we need to determine the covariance Q of the process noise, the covariance R of the measurement noise as well as the initial error covariance P . To determine estimates for these values, four volunteers were invited to read and play eight set pieces while recording both eye-gaze and keying information. The subjects were asked to play the extracts first as a sight-reading task and then, after allowing a 2-min practice session. Moreover, the extracts were selected such that they contained examples of irregular time and key signatures, varying rhythmic and pitch complexities, and tempo changes. The keying information obtained directly from the MIDI output of the digital piano was synchronized with the score through dynamic time warping. The note onset from the MIDI data was then used to align the eye-gaze information with the keying information and the score. In this manner, we could observe the eye-gaze data under different conditions, allowing for monitoring of variations in

the eye-hand span, glances at keyboard and variations on the reading advancements.

From the registration of the MIDI data with the score, we observe that, in general, the pianists position on the score follows the process model described by Equation (13). Variations from this model in the vertical direction exist when the pianist's position on the score shifts from the top to the bottom line of the system. While variations in the horizontal directions are observed mainly due to deviations from the constant tempo model. Using these observations, we empirically determine the initial values for $P_{k|k-1} = 0.1I$ where I is the identity matrix, and set the process noise covariance to

$$Q_k = \begin{bmatrix} 0.2 & 0 & 0.6 \\ 0 & 0.85 & 0 \\ 0.6 & 0 & 0.2 \end{bmatrix}$$

choosing these values as they best describe the observed variances in the MIDI note onsets. Moreover, from the registration of the point-of-regard and the MIDI data, we observe larger deviations between the point-of-regard and the position on the score. These deviations are due to forward and backward glances as well as vertical oscillations as the pianist reads from both staves of the system. Since these deviations represent the variance that we can expect in the measurement, we empirically set the measurement noise covariance to

$$R = \begin{bmatrix} 5 \times 10^{10} & 0 \\ 0 & 5 \times 10^5 \end{bmatrix}$$

as this best describes the observed variances between the measured point-of-regard and the position on the score.

3.5. Loss in the Eye-Gaze Measurement

The discussion thus far assumes that the eye-gaze tracker in use can locate the pianist's eyes at all times. However, from our preliminary study, we note that there are instances when pianists shift their position at the piano, for example, by leaning toward the higher or lower registers of the piano. In doing so, the eyes shift out of the field of view of the eye-gaze tracker, resulting in a loss of eye-gaze measurements. This loss results in measurement data of $\mathbf{z} = 0$. While the Kalman filter tolerates noisy data, long instances of erroneous measurements will cause the Kalman filter to diverge, particularly since such losses in the eye-gaze measurements tend to occur over long, consecutive time intervals. Such divergence may lead to accidental page turns which is undesirable.

To compensate for loss in measurement data, we monitor the eye-gaze measurements and in the case of consecutive losses, we interpolate the missing eye-gaze measurements. The interpolation uses the assumed process model such that:

$$\mathbf{z}_{1,k} = \mathbf{x}_{1,k-1|k-1} + \mathbf{x}_{3,k-1|k-1} \quad (14)$$

$$\mathbf{z}_{2,k} = \mathbf{x}_{2,k-1|k-1} \quad (15)$$

When the pianist's eye are once again within the field of view of the tracker, the measurement data will revert to those obtained through the eye-gaze tracker, allowing the Kalman filter to update the state vectors with the new, actual measurement input. Although this approach may introduce some drift, the error due to this drift will not be as large as the divergence caused due to loss in the measurement data.

This approach allows us to use a hybrid model to determine the pianist's location on the score. At instances when measurement data is available, the pianist's position is determined through the Kalman filter eye-tracking model. In the absence of any measurement data, we follow the constant velocity model until sensible measurements are once more obtained from the eye-gaze tracking device. Relying only the interpolation models of Equations (14) and (15) would make the estimation of the pianist's reading position susceptible to the inherent noise of the eye-gaze tracking device as well as variations in the eye-gaze movements as discussed above.

3.6. Using the Reading Position to Effect a Page Turn

Page turning is effective if, when the pianist approaches the end of the system on the page, the new system of the subsequent page is already within the pianist's field of view. In this paper, we adopt the half-page turning described in Bonnici et al. (2017), with the score having already been pre-processed to identify the systems, bar-lines and with all repeats flattened. Since the viewing device is intended to be a regular-sized tablet, for readability, each page consists of only two systems displayed at any one time. With two systems per page, half-page turns involve updating one system at a time. Thus, a system S_n will be updated with system S_{n+2}

when the pianist reads from the system S_{n+1} . However, we note that due to looking-ahead habits, toward the end of a system, the pianist may have both systems in focus. Updating a system the instance the gaze is averted to the next system may, therefore, be too distracting for the pianist. For this reason, it is desirable to allow the gaze to settle in the new system before effecting the half-page turn.

To achieve this, we create a rectangular area of interest on each new system displayed. This rectangular area of interest spans from the second detected bar-line to the last bar-line of the system as shown in **Figure 3**. We use this region of interest to accumulate the number of times the pianists gaze falls within the region of interest. By requiring a minimum number of gaze instances within the region of interest, we may ensure that the pianists gaze would have settled on the new system such that effecting the page turn would not be distracting. Empirically, we determine that for pieces played at an average tempo of 120 bmp, we may set the minimum threshold to a fifth of the width of the region of interest. We normalize this threshold with the user-defined average speed of execution of the piece to take into account that faster (slower) average tempo will reduce (increase) the time spent within the region of interest.

4. EVALUATION METHODOLOGY

To evaluate the performance of the proposed Kalman filter model and subsequently, the eye-gaze based page-turning, we adopt a two-step evaluation process, using the model first with a set of simulated data, followed by an evaluation with real eye-gaze data. The simulated tests allow us to observe the Kalman filter model with respect to ground truth data and hence, determine the



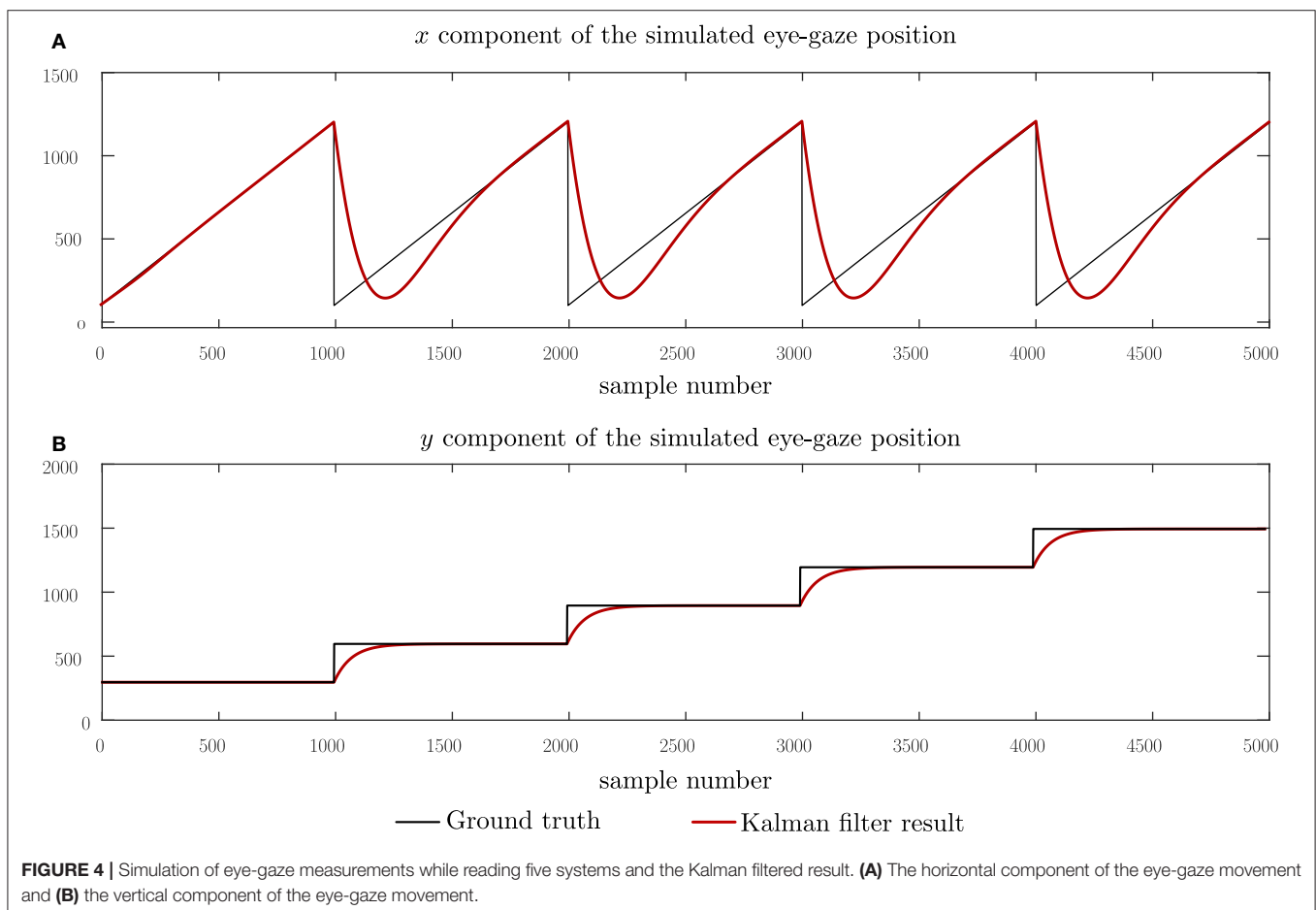
residual error of the Kalman filter model. The simulated ground-truth data was created using the process model described by Equation (1) using a steady tempo of 120 bpm to simulate the reading velocity, a system width of 1,200 pixels, a system height of 200 pixels and a separation of 300 pixels between systems, resulting in an effective page size of $1,200 \times 700$ pixels. For simulation purposes, we create a page consisting of five systems as shown in **Figure 4**. We then introduce perturbations to the ground-truth data to simulate expected characteristics in the real data. We first simulate short-time losses of the eye-gaze measurement data which can be brought about by glances at the keyboard. These are modeled as impulses in both the horizontal and vertical components of the eye-gaze measurements. We then model the noise in the measurement of the point of regard due to micro-saccades in the eye movements as well as noise introduced by the eye-gaze sensor itself. This noise is modeled as additive Gaussian noise, changing the signal-to-noise ratio by varying the variance of the noise distribution. The final simulation attempts to emulate longer losses in the eye-gaze measurements by introducing longer zero-pulses to the ground truth measurement data. This evaluation allows us to compare the effect of the measurement data interpolation on the resulting Kalman filter outcome.

In all these tests, the Kalman filter outputs were expected to follow the ideal input in an over-damped

manner due to a tendency of the Kalman filter model to withstand changes in each direction as set in the noise co-variance matrices.

To evaluate the Kalman filter model with real eye-gaze data, we use the SMI RED500 eye-gaze tracker⁷. This eye-gaze tracker uses infrared illumination alongside computer-based image processing to detect the gaze location of the user on a designated area of interest. For optimal conditions of operation, the subject is to sit 60 – 80 cm away from a 22-inch monitor, where an allowable head box of roughly 40×20 cm is formed. Under these conditions, the system offers a binocular tracking with a maximum sampling rate of 500 Hz, contact-free measurement, small automatic head-movement compensation for head movement velocities of up to 50 cm/s by using the corneal reflexes and a typical gaze position accuracy of around 0.4° . The eye-gaze tracker is connected to a workstation running the iView XTM software. This software facilitates the capturing of eye movements by controlling all the camera equipment and processing all eye and scene video signals captured. This workstation is connected, via Ethernet, to a personal device which hosts our page-turning application. Our application is Matlab-based and communicates to the workstation by using an application

⁷<https://imotions.com/hardware/smi-red500/>



programming interface (API) provided by the iView XTM software development kit (SDK). The API allows our page-turning application to control the SMI Red500 and retrieve eye-tracking data.

Using this setup, two tests were carried out. In the first instance, the subject was asked to read and perform 15 piano scores normally, allowing changes in speed within the performance. For this test, 15 different musical pieces were selected such that the pieces exhibited different levels of difficulty and changes in tempo. For these pieces, the performance of the Kalman filter model for page turning applications was evaluated by counting the number of successful page turns, delayed page turns, and advanced page turns. For the purpose of this work, we define successful page turns as those page turns that do not interrupt the flow of music. Delayed page turns are defined as those instances when the pianist has completed the system but the next system is not displayed, introducing a delay in the flow of the music. Likewise, advanced page turns are page turns triggered before the pianist has finished reading the system. These page turns are more disruptive than delayed page turns since they introduce jumps in the music. In the second part of the reading test, we deliberately introduced re-starts and skips in the flow of music to determine whether the Kalman filter model was equally able to retain the successful page turns under large disturbances from the assumed reading model.

5. RESULTS

Figure 4 shows the performance of the proposed Kalman filter model under clean, idealized eye-gaze measurements. These measurements will be used as ground-truth when evaluating the performance of the Kalman filter model. From **Figure 4**, we can observe that, as expected, the Kalman filter acts as an over-damped filter, allowing the system states to reach the desired output. The results shown here demonstrate that the proposed model can follow through changes in reading direction corresponding to shifts in the eye-gaze between different systems. A root-mean-square (RMS) error of 106.0 pixels was observed with this input and this corresponds to a lag between the ideal and predicted states. This lag can be broken down into a lag of 103 pixels in the horizontal direction, equivalent to 8.5% of the page width, and 25 pixels in the vertical direction, equivalent to 3.6% of the page height.

In **Figure 5**, we simulate brief losses in the measurement data with impulses inserted at equally spaced intervals along the measurement. We note that the Kalman model filters out these impulses such that the predicted gaze positions lie close to the expected ground truth. An RMS error of 109.9 pixels was observed, which indicates that the difference between the Kalman filter results and the ground truth is mostly due to the lag observed in **Figure 4** and that the impulses introduced have little effect on the Kalman filter performance.

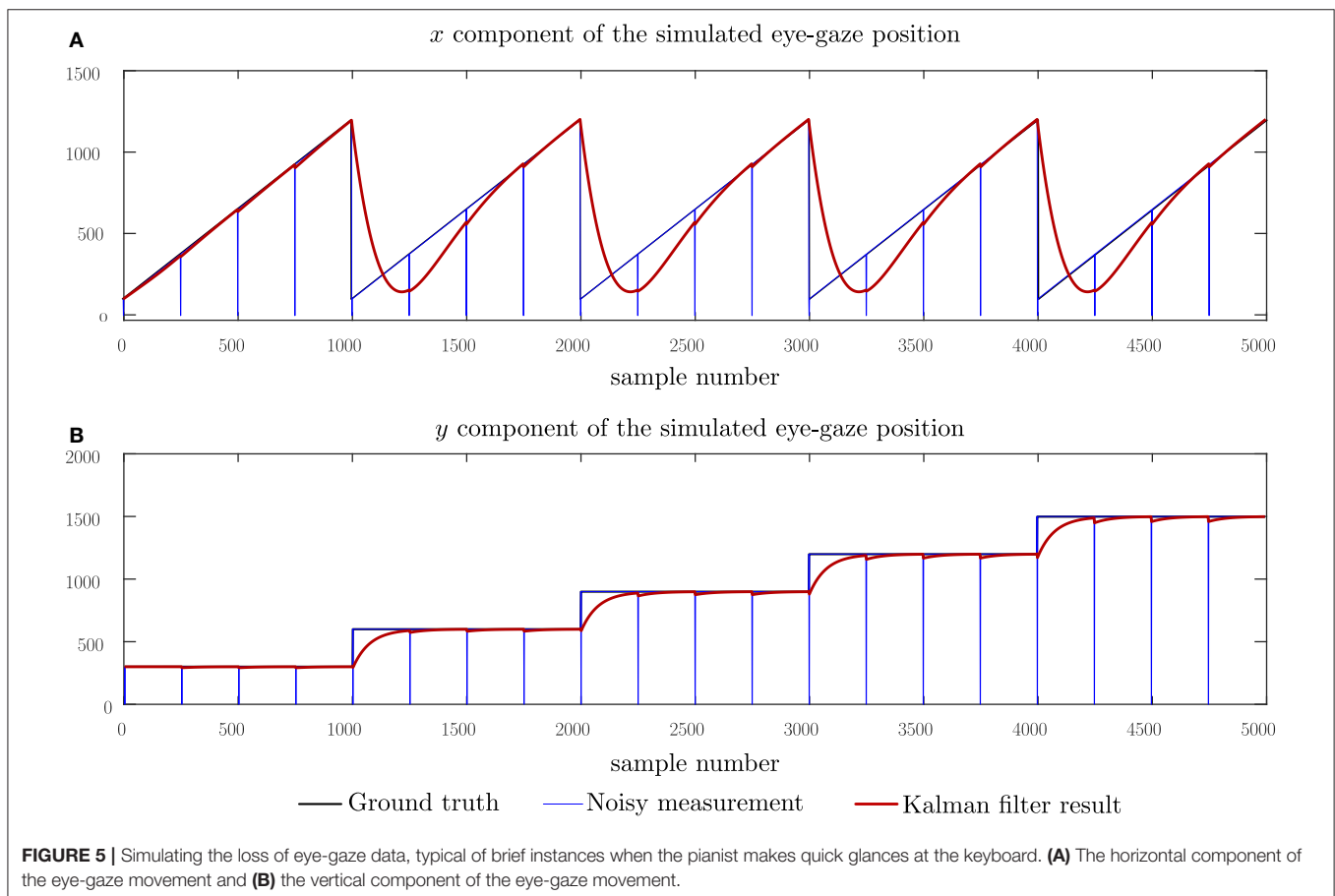
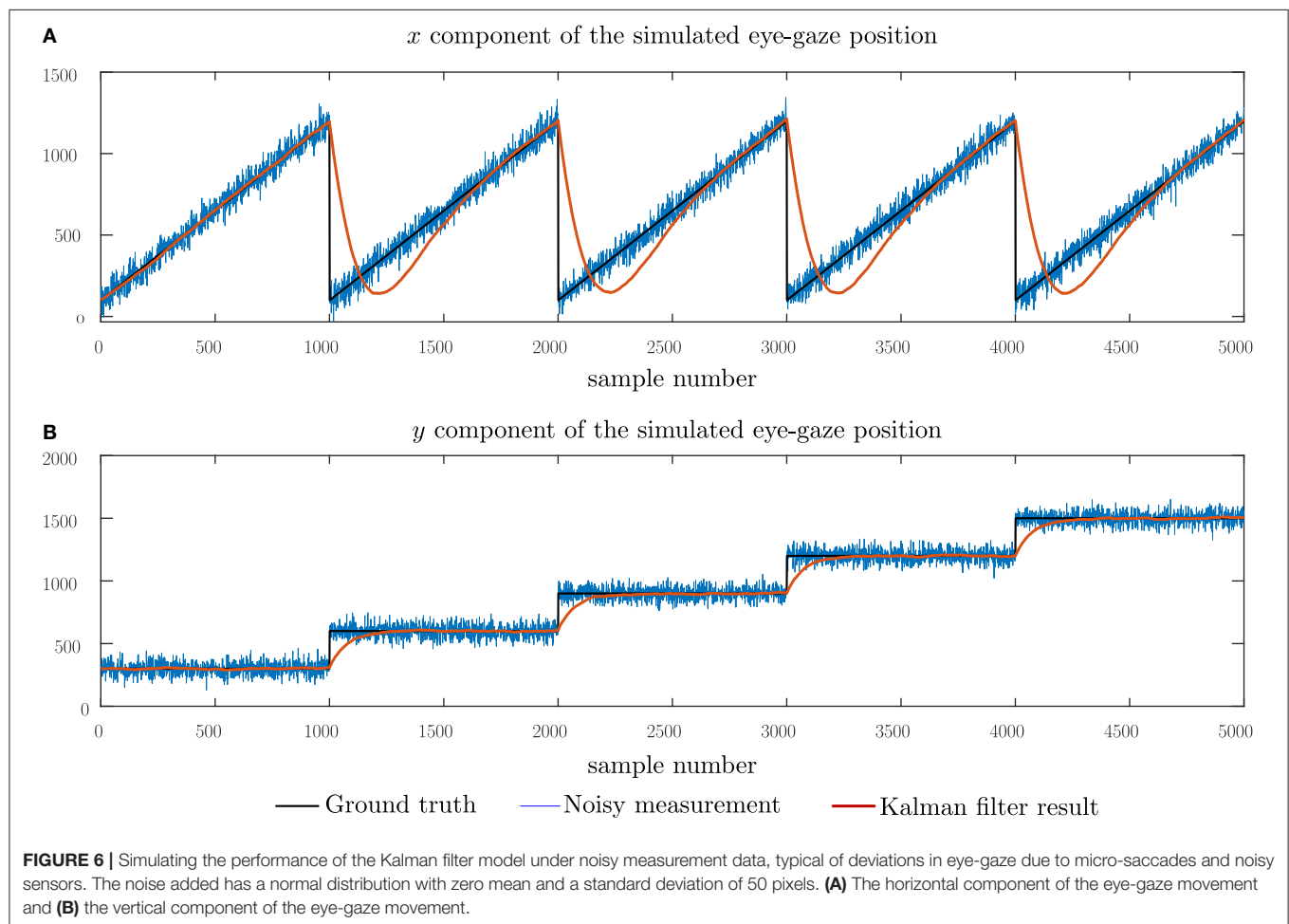


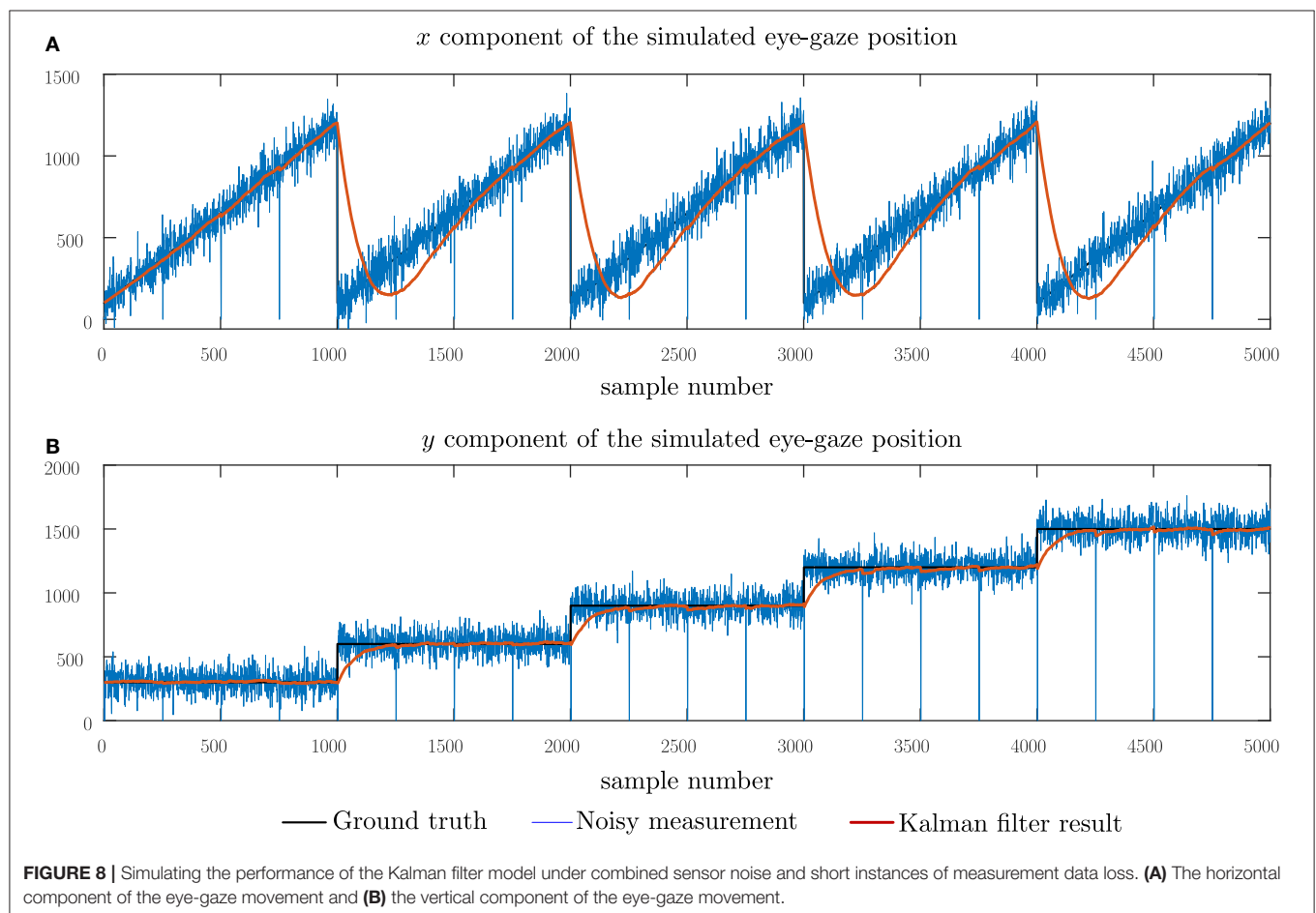
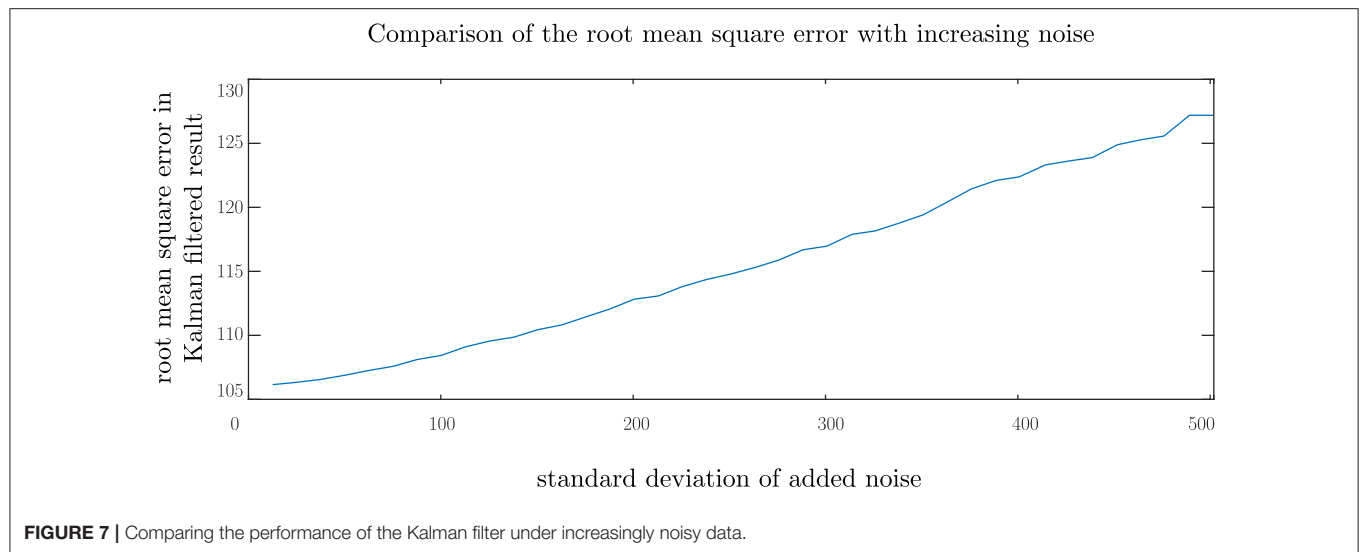
Figure 6 shows the performance of the Kalman filter model under the presence of normally distributed noise having zero mean and a standard deviation of 50 pixels. This graph demonstrates how the Kalman filter model adopted compensates for noisy signals and is, therefore, robust to noise in eye-gaze movements due to micro-saccades as well as noise in the sensor itself. For a noise with a standard deviation of 50 pixels, an RMS error of 107.2 pixels was observed. This error is comparable to the error due to the lag introduced by the model. In **Figure 7**, we show the change in the RMS error with increasing noise up to a standard deviation of 500 pixels. From this graph, we may note that, although the RMS error of the Kalman filter increases, the remaining noise in the filtered data is greatly reduced in comparison with the noise in the measurements. The performance of the Kalman filter model was further observed under combined impulse and normally distributed noise, mimicking instances of noisy sensor and short data losses. The results of this simulation are shown in **Figure 8** and, in this case, an RMS error of 111.5 pixels showing that the Kalman filter has the same level of performance under the combined noise models.

In **Figure 9**, we observe the effect of longer periods of measurement data loss, comparable to instances when the

subject's eyes fall outside the field-of-view of the eye-gaze tracker. Here, we compare the performance of the Kalman filter (red) with the same filter model but after performing measurement data interpolation (green). From this result, we may note that loss in the measurement causes the Kalman filter to drift toward the zero level, recovering toward the ground-truth once the measurement data is regained. By applying the measurement data interpolation, the Kalman filter output is being effectively clamped to the constant reading model which not only reduces the drift from the ground truth, but also allows the Kalman filter model to recover from the loss of measurement data more quickly.

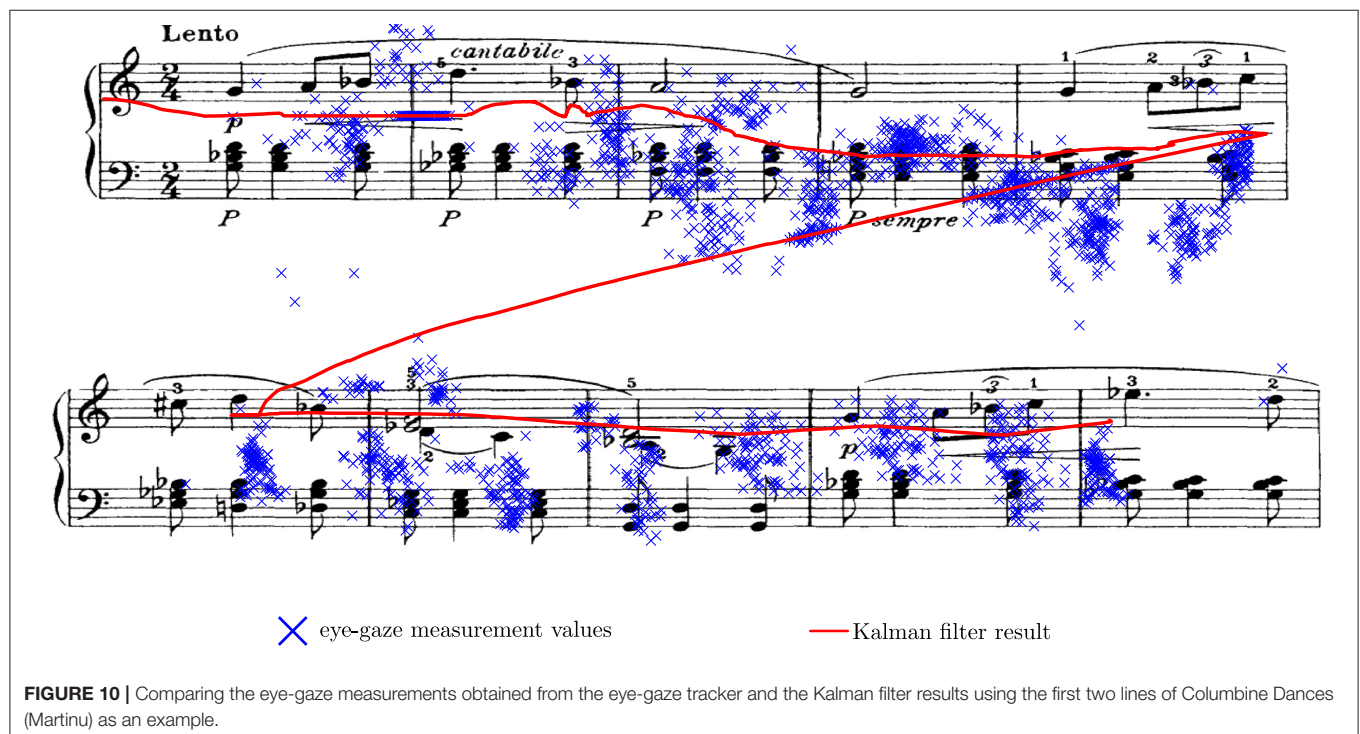
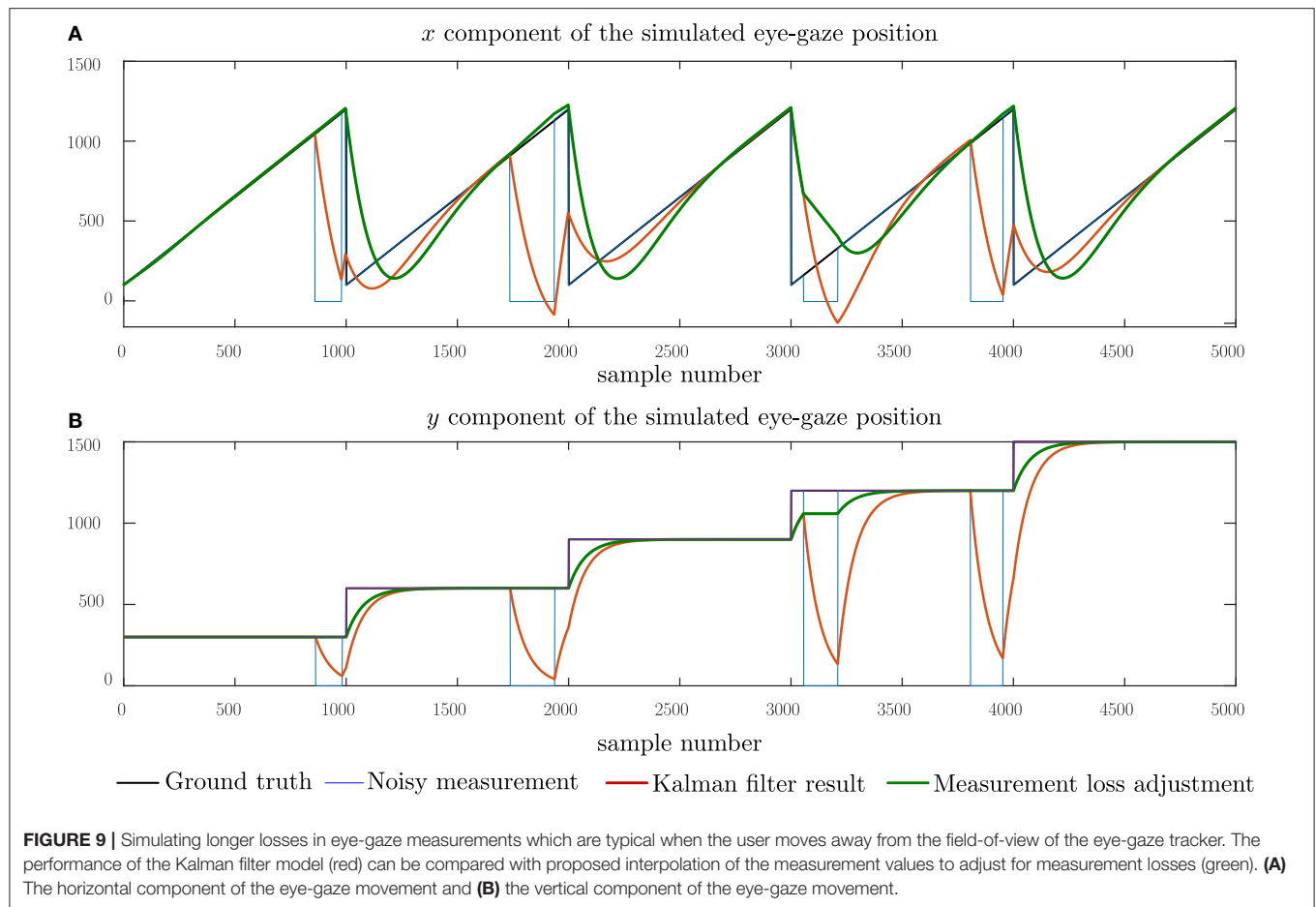
Figure 10 shows the eye-gaze measurements sensed by the eye-gaze tracker while the subject performed the extract. The state-vector from the Kalman filter model is superimposed on this sensed data. Similar to the simulated tests, we can observe that the Kalman filter model reduces the noise in the eye-gaze position estimation, resulting in smoother eye-gaze movements on the score. In **Figure 11**, we show the Kalman filtered gaze locations and the instances when page turns occur for the entire piece. We superimpose on the graph the region of interest within which, we expect the page turn to occur. Page turns within these regions will ensure that the pianist has the next system in place before reaching the end of the current system. Page turns

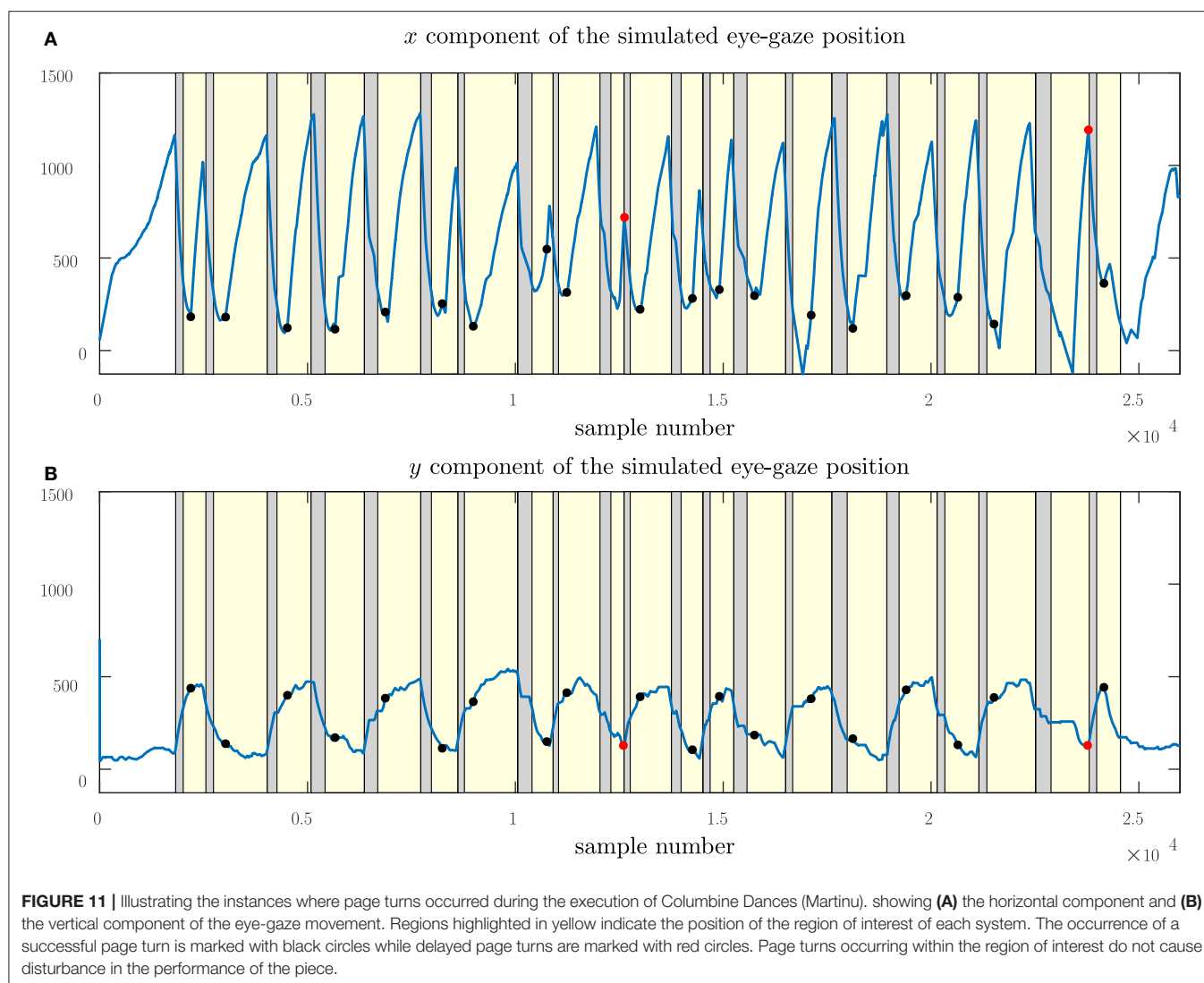




that occur before the region of interest are likely to disturb the subject by occurring too soon, when the subject is transitioning between systems. On the other hand, page turns that occur too late within the region of interest will delay the page turn, causing

the subject to wait for the page turn to occur. In **Figure 11**, we can observe that of the 21 page turns required to perform Columbine Dances, two of these page turns occurred just at the end of the region of interest, introducing an undesired pause





in the music. The remaining 19 page turns occurred within the region of interest and can thus, be considered as successful page turns. **Table 1**, documents the total number of successful, early and late page turns for the 15 selected pieces. From this table, we note that out of 289 page turns, only 5 page turns were delayed, resulting in a 98.3% successful page turns. The delays observed in the Columbine Dances are due to written tempo change instruction from a slow section to a faster section. In these cases, although the Kalman filter model did adapt to the change in the tempo, the adaptation was not sufficiently quick. The other three delays are mostly due to the score flattening approach adopted from (Bonnici et al., 2017). These pieces had repeat marks within the first half of the system, resulting in very short systems where the image of the written score was cut short to allow for the insertion of the repeated section at the next system. In these cases, our page-turning model required the subject to spend more time within the system before executing a page turn. Adjusting the type-setting of the music through, for example, re-writing the flattened score in MusicXML, would ensure systems of more uniform lengths and hence, eliminate

this problem. Nevertheless, the delays incurred were of under 3 s in duration and thus, not unlike the delays experienced when manually adjusting page turns, with the added advantage that the subject can trigger the page turn without needing to remove their hands from the keyboard.

Figure 12 further shows the performance of the eye-gaze tracking under instances when the subject stops and restarts reading the same section of music. The results shown demonstrate that the Kalman filter model can react to such changes, allowing for the eye-gaze following to function even under changes in the subject's gaze from the expected reading model. Accommodating such changes is necessary as it allows the eye-gaze page-turning to function even under instances of practice time.

For comparison purposes, we performed a subset of the scores presented in **Table 1** using the audio-based, page turning function of the PhonicScore App⁸. The music was performed on a Yamaha Clavinova CLP545 digital piano. For purposes

⁸<http://phonicscore.com/>

TABLE 1 | The performance of the automated page turning on 15 different musical pieces, giving the total number of page turns required as well as the number of successful page turns, the number of late page turns and the number of early page turns as a percentage of the total number of page turns.

| Selected piece | Page turns | | | |
|---|------------|----------------|----------|-----------|
| | Total | Successful (%) | Late (%) | Early (%) |
| Columbine Dances, Puppets II, Martinu | 21 | 90.5 | 9.5 | 0 |
| Gavotte, Holberg Suite, Grieg | 20 | 100 | 0 | 0 |
| Gnosienne No. 1, Satie | 15 | 100 | 0 | 0 |
| Children's Corner Suite, Mvt. No. 6, Debussy | 18 | 100 | 0 | 0 |
| Impromptu in G flat, Schubert | 40 | 100 | 0 | 0 |
| Maple Leaf Rag, Joplin | 34 | 100 | 0 | 0 |
| Minuet, from Sonata No. 1, Beethoven | 20 | 95 | 5 | 0 |
| Moonshadows On The Mountain, Linn | 13 | 92.3 | 7.7 | 0 |
| My Father's Favorite, Doyle | 16 | 100 | 0 | 0 |
| Nocturne In C Sharp Minor, Chopin | 16 | 93.8 | 6.2 | 0 |
| Papillon Noir, Massenet | 15 | 100 | 0 | 0 |
| Prelude In C, from 48 Preludes and Fugues, Bach | 10 | 100 | 0 | 0 |
| Song For Sienna, Crain | 24 | 100 | 0 | 0 |
| Sundial Dreams, Kern | 27 | 100 | 0 | 0 |
| Total | 289 | 98.3 | 1.7 | 0 |

TABLE 2 | The performance of the automated page turning using PhonicScore on eight of the pieces given in **Table 1**.

| Selected piece | Page turns | | | |
|---|------------|----------------|----------|-----------|
| | Total | Successful (%) | Late (%) | Early (%) |
| Columbine Dances, Puppets II, Martinu | 14 | 85.7 | 0 | 14.3 |
| Gnosienne No. 1, Satie | 10 | 20.0 | 80.0 | 0 |
| Children's Corner Suite, Mvt. No. 6, Debussy | 12 | 58.3 | 0 | 41.7 |
| Maple Leaf Rag, Joplin | 23 | 56.5 | 26.1 | 17.4 |
| Nocturne in C Sharp Minor, Chopin | 11 | 63.6 | 18.2 | 18.2 |
| Prelude in C, from 48 Preludes and Fugues, Bach | 5 | 100 | 0 | 0 |
| Song For Sienna, Crain | 16 | 50 | 0 | 50 |
| Sundial Dreams, Kern | 18 | 61.2 | 38.8 | 0 |
| Total | 109 | 61.5 | 19.3 | 19.2 |

The table presents the total number of page turns required as well as the number of successful page turns, the number of late page turns and the number of early page turns as a percentage of the total number of page turns.

of evaluation, the CFX Grand Piano tone was used while the use of pedals was not allowed since any other tone, or the use of pedaling prevented PhonicScore from recognizing the notes being played. The scores were selected on basis of the availability of the music in MIDI and MusicXML file format which are the two file formats recognized by the app. **Table 2** gives the number of successful, late and early page turns experienced when using this application. It is important to note that PhonicScore is not restricted to half-page turns and the number of systems presented in a page depends on the density of the music. Overall, the application therefore requires fewer page turns per score. From **Table 2**, we note that this application has a larger quantity of late and early page turns than our eye-gaze tracking system. Moreover, in all instances, manual intervention was needed to place the cursor position in the correct place on the score. All late and early page turns occurred after the application was unable to match the audio signal with the correct place on the score. In instances of late page turns, the application was unable to pick-up where the user was playing and did not advance at all, whereas in early page turns, the application found matches in places ahead of the user's current position on the score, skipping ahead in the score. These observations further demonstrate the advantages of using eye-gaze tracking for page-turning.

6. CONCLUSION

In this paper we present an eye-gaze page turning system that allows performers to browse through the music while performing it without lifting the hands from the keyboard. To achieve this page turning system, we describe a simple reading model which describes the way a subject's eye-gaze progresses through the music score when reading and performing the music. This reading model makes assumptions about the reading velocity that are not necessarily strictly observed by the subject. Measurements of the subject's point-of-regard through eye-gaze trackers are therefore used to adjust the position on the score. However, we note that such a sensor introduces measurement noise and thus, we propose a Kalman-filter model to reach a balance between the reading model and the measurement data.

The resulting eye-gaze tracking allows us to create a robust page-turning system which, when paired with a half-page turning display allows constant update of the displayed page such that the subject always has fresh music to play from. The results obtained show that successful page turns occurs in 98.3% of the page turning instances. Furthermore, the page-turning system is robust to instances of re-starts and skips along the system being read.

In our model, we assume that, for the most part, the subject needs to look at the score to read the notes from the score. However, one can envisage instances when the player performs parts of the score from memory. In such instances, our proposed measurement interpolation prevents the Kalman filter model from diverging. Accuracy in the model can be increased if the proposed Kalman filter model is augmented to include a

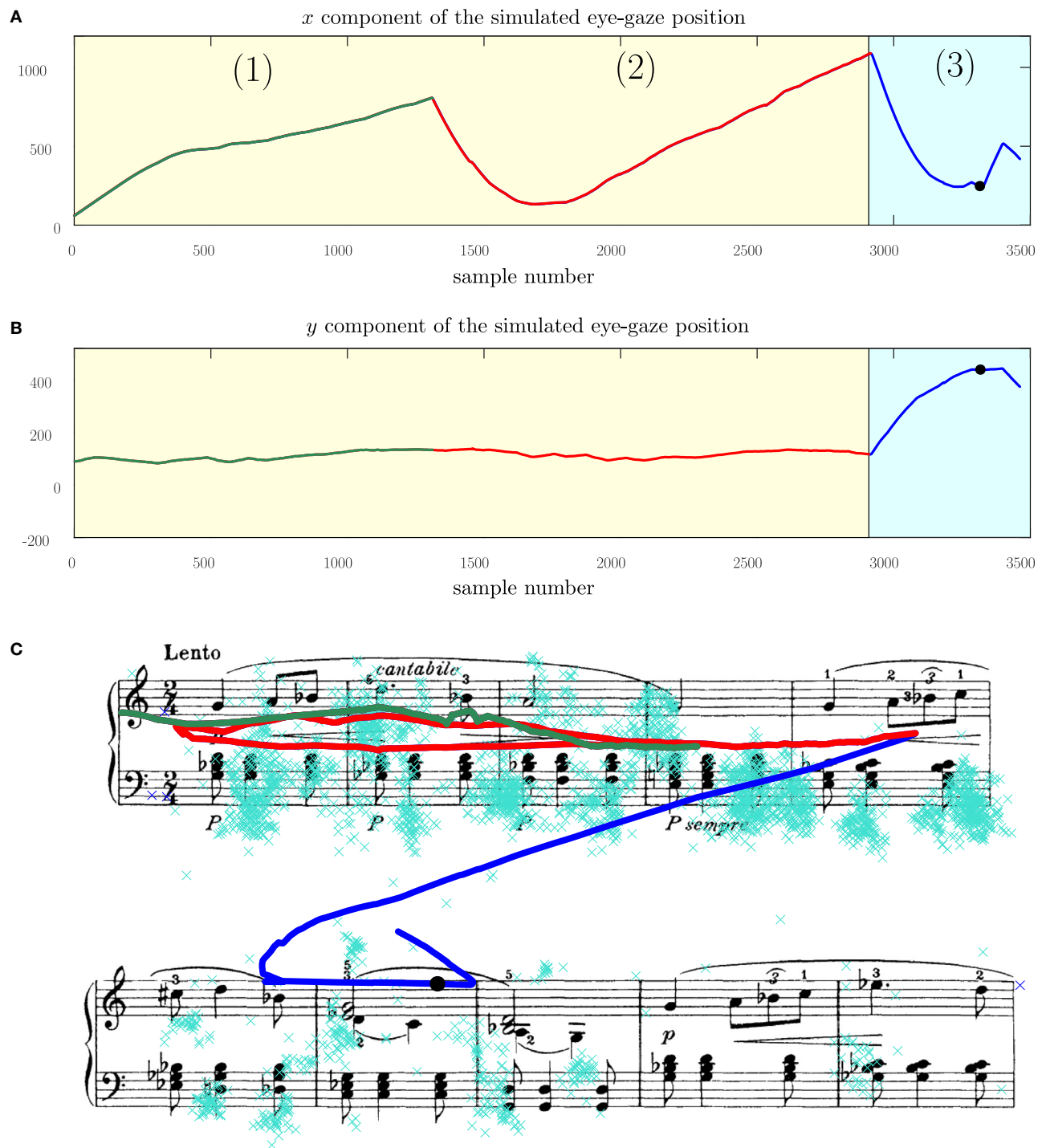


FIGURE 12 | Illustrating the performance of the page-turning under conditions of re-starts and skips, showing **(A)** the horizontal eye-gaze position, **(B)** the vertical eye-gaze position **(C)** the measured and Kalman filtered eye-gaze values on the score. (1) The subject starts by reading the music normally but at (2) stops and restarts the performance from the beginning of the system. The Kalman filter eye-gaze tracking model responds in kind and restarts from the beginning of the system too. The current system remains visible for the subject, causing no interruptions in the flow other than those intentionally introduced by the subject. At (3) the subject proceeds to the next system and the Kalman filter model detects this change. The subject plays the first, second, and third bars of this system, but then skips the fourth bar and goes straight to the fifth bar. The Kalman filter treats such a skip as noise in the measurement model and lags behind. However, the page-turning mechanism can sense that the subject has moved to the second system and can update the first system (not shown here). Thus, when the subject completes the second system, the page is refreshed and can proceed with performing the next system which would be displayed on top.

second measurement input, namely, sound measurement. The Kalman filter model would then combine the stochastic nature of the gaze and sound measurements to create a more robust score following.

DATA AVAILABILITY STATEMENT

The datasets generated for this study will not be made publicly available. The eye-gaze data required for the purpose of this study was not recorded during the study. The musical sheets used can be provided, but these are sheets which are readily and freely available from various music libraries.

REFERENCES

- Bell, B. (2012). "A better page turner for organists," in *Going Digital for Musicians* Online blog. Available online at: <https://goingdigitalmusician.wordpress.com/>
- Bell, T. C., Church, A., McPherson, J., and Bainbridge, D. (2005). "Page turning and image size in a digital music stand," in *International Computer Music Conference* (Barcelona).
- Blinov, A. (2007). *An interaction study of a digital music stand* (thesis). University of Canterbury, Christchurch, New Zealand.
- Bonnici, A., Cristina, S., and Camilleri, K. P. (2017). "Preparation of music scores to enable hands-free page turning based on eye-gaze tracking," in *Proceedings of the 2017 ACM Symposium on Document Engineering, DocEng '17* (New York, NY: ACM), 201–210. doi: 10.1145/3103010.3103012
- Cara, M., A. (2018). Anticipation awareness and visual monitoring in reading contemporary music. *Musicae Scientiae* 22, 322–343. doi: 10.1177/1029864916687601
- Chen, C., and Jang, J. S. R. (2019). An effective method for audio-to-score alignment using onsets and modified constant Q spectra. *Multimedia Tools Appl.* 78, 2017–2044. doi: 10.1007/s11042-018-6349-y
- Dannenberg, R. B. (2012). Human computer music performance. *Multimodal Music Process, Vol.3*, eds M. Muller, M. Goto, and M. Schedl (Berlin: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik), 121–134. doi: 10.4230/DFU.Vol3.11041.121
- Dannenberg, R. B., Gold, N. E., Liang, D., and Xia, G. (2014). Active scores: representation and synchronization in human computer performance of popular music. *Comput. Music J.* 38, 51–62. doi: 10.1162/COMJ_a_00239
- Dorfer, M., Arzt, A., and Widmer, G. (2016). "Towards score following in sheet music images," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)* (New York, NY).
- Dorfer, M., Arzt, A., and Widmer, G. (2017). "Learning audio-sheet music correspondences for score identification and offline alignment," in *International Society for Music Information Retrieval* (Suzhou). doi: 10.5334/tismir.12
- Gibbs, W. W. (2014). Hands-free sheet music [resources-hands on]. *IEEE Spectrum* 51, 27–28. doi: 10.1109/MSPEC.2014.6905480
- Graefe, C., Wahila, D., Maguire, J., and Dasna, O. (1996). "Designing the muse: a digital music stand for the symphony musician," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '96* (New York, NY: ACM), 436. doi: 10.1145/238386.238599
- Huovinen, E., Ylitalo, A. K., and Puurtinen, M. (2018). Early attraction in temporally controlled sight reading of music. *J. Eye Movement Res.* 11. doi: 10.16910/jemr.11.2.3
- Jin, Z. (2013). *Formal semantics for music notation control flow* (Master's thesis). Carnegie Mellon University, Pittsburgh, PA, United States.
- Laundry, B. A. (2011). *Sheet music unbound: a fluid approach to sheet music display and annotation on a multi-touch screen* (Master's thesis). University of Waikato, Hamilton, New Zealand.
- Maybeck, P. S. (1979). *Stochastic Models, Estimation and Control, Vol. 1*. London: Academic Press Inc.
- Murphy, K. P. (1998). *Switching Kalman filters*. Technical report. DEC/Compay Cambridge Research Labs.
- Nieweg, C. F., and Vaught, G. (2011). *Music Preparation Guidelines for Orchestral Music*. Major Orchestra Librarians' Association.
- Noto, K., Takegawa, Y., and K., H. (2019). "Adaptive score-following system by integrating gaze information," in *Proceedings of the 16th Sound and Music Computing Conference* (Málaga).
- Penttinen, M., Huovinen, E., and Ylitalo, A. K. (2015). Reading ahead: adult music students' eye movements in temporally controlled performances of a children's song. *Int. J. Music Educ.* 33, 36–50. doi: 10.1177/0255761413515813
- Puurtinen, M. (2018). Eye on music reading: a methodological review of studies from 1994 to 2017. *J. Eye Movement Res.* 11, 1–16. doi: 10.16910/jemr.11.2.2
- Ringwalt, D., Dannenberg, R. B., and Russell, A. (2015). "Optical music recognition for interactive score display," in *Proceedings of the International Conference on New Interfaces for Musical Expression, NIME 2015* (Baton Rouge, LA: The School of Music and the Center for Computation and Technology (CCT), Louisiana State University), 95–98.
- Rosemann, S., Altenmüller, E., and Fehle, M. (2016). The art of sight-reading: influence of practice, playing tempo, complexity and cognitive skills on the eye-hand span in pianists. *Psychol. Music* 44, 658–673. doi: 10.1177/0305735615585398
- Terasaki, S., Takegawa, Y., and Hirata, K. (2017). "Proposal of score-following reflecting gaze information on cost of DP matching," in *International Computer Music Conference Proceedings* (Shanghai).
- Vandemoortele, S., Feytaerts, K., De Bièvre, G., Reybrouck, M., Brône, G., and De Baets, T. (2018). Gazing at the partner in musical trios: a mobile eye-tracking study. *J. Eye Mov. Res.* 11. doi: 10.16910/jemr.11.2.6
- Wolberg, G., and Schipper, I. (2012). Page turning solutions for musicians: a survey. *Work* 41, 37–52. doi: 10.3233/WOR-2012-1242

ETHICS STATEMENT

The studies involving human participants were reviewed and approved by University of Malta. Written informed consent for participation was not required for this study in accordance with the national legislation and the institutional requirements.

AUTHOR CONTRIBUTIONS

AT was the primary contributor of the algorithms required to perform this research and carried out this research under the tutorship and supervision of both AB and SC.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Tabone, Bonnici and Cristina. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Creating Music With Fuzzy Logic

Rodrigo F. Cádiz^{1,2*}

¹ Faculty of Arts, Music Institute, Pontificia Universidad Católica de Chile, Santiago, Chile, ² Department of Electrical Engineering, School of Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile

Fuzzy logic is an artificial intelligence technique that has applications in many areas, due to its importance in handling uncertain inputs. Despite the great recent success of other branches of AI, such as deep neural networks, fuzzy logic is still a very powerful machine learning technique, based on expert reasoning, that can be of help in many areas of musical creativity, such as composing music, synthesizing sounds, gestural mappings in electronic instruments, parametric control of sound synthesis, audiovisual content generation or sonification. We propose that fuzzy logic is a very suitable framework for thinking and operating not only with sound and acoustic signals but also with symbolic representations of music. In this article, we discuss the application of fuzzy logic ideas to music, introduce the Fuzzy Logic Control Toolkit, a set of tools to use fuzzy logic inside the MaxMSP real-time sound synthesis environment, and show how some fuzzy logic concepts can be used and incorporated into fields, such as algorithmic composition, sound synthesis and parametric control of computer music. Finally, we discuss the composition of *Incerta*, an acousmatic multichannel composition as a concrete example of the application of fuzzy concepts to musical creation.

Keywords: fuzzy logic, computer music, machine learning, sound synthesis, parametric control, algorithmic composition

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Rinkaj Goyal,
Guru Gobind Singh Indraprastha
University, India
Arnaud Fadja Nguembang,
University of Ferrara, Italy

*Correspondence:

Rodrigo F. Cádiz
rcadiz@uc.cl

Specialty section:

This article was submitted to
Machine Learning and Artificial
Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 30 October 2019

Accepted: 07 July 2020

Published: 30 October 2020

Citation:

Cádiz RF (2020) Creating Music With
Fuzzy Logic. *Front. Artif. Intell.* 3:59.
doi: 10.3389/frai.2020.00059

1. INTRODUCTION

Music, although considered a science by many, is not an exact science, but rather a collection of qualities, ranging from the emotional to the intellectual in varying degrees (Suiter, 2010a). Several concepts in music are not absolute but rather relative and its terminology is not entirely precise. Many musical concepts do not possess an absolute meaning, and composers, with a few notable exceptions, do not specify every detail about how their musical creations should be converted into sound. For example, a slow tempo indication in a musical score can be interpreted very differently by different analysts or performers. Indeed, many musical attributes are described by imprecise (or fuzzy) concepts, such as *presto*, *forte*, *piano*, *andante*, or *allegro*. León and Liern (2012) provide the music of J.S. Bach as an example of such a fuzzy approach to composing, as in his music features, such as the instrumentation or the tempo are not explicitly stated in the scores.

Following the same line of argument, authors, such as Milicevic (1999) state that music, unlike language, is fuzzy, while others, such as León and Liern (2012) consider a musical score to be a truly fuzzy system, meaning that performers are required to execute very complex actions based on uncertain concepts written in the score. If we accept this premise, some aspects of fuzzy logic theory seem to be a natural way of predicting the aesthetic outcomes of music (Suiter, 2010a) and its structure.

Fuzzy logic (Kosko, 1993; McNeill and Freiberger, 1993; Cox, 1994; Bandemer and Gottwald, 1995; Klir and Yuan, 1995; Yen and Langari, 1999) is a branch of artificial intelligence specifically

designed to handle imprecise and vague concepts. Fuzzy logic can be conceived as a logical system based on a more general concept of truth, one that is not two-valued (true or false) and very appropriate for reasoning under uncertainty, by allowing different degrees of membership or several values of truth. In general, the application of fuzzy logic inference to a problem emulates some aspects of human reasoning, for example, the quantification of imprecise information or making decisions given unclear or partial data (Kosko, 1993).

Artificial intelligence aims to construct computational algorithms that can perform some level of reasoning and exhibit problem-solving skills similar to those of humans. Fuzzy logic has an additional objective: “to explore an effective trade-off between precision and the cost in developing an approximate model of a complex system or function” (Yen, 1999). However, perhaps one of the most important qualities of fuzzy logic concerning music-making is its capacity for modeling non-linear systems without the need of explicitly constructing a complex mathematical model. Indeed, as Suiter (2010a) states: “A significant feature of music is that the aesthetic outcome is often more than the sum of its technical elements. Indeed, what is the role of timber, attack, duration, decay, articulation, spatialization, register, texture, voicing, entries and timing, rhythm, tempo, or meter? What does musical form, structure, or process contribute? In fact, it is often the means and details of the interactions between the distinct elements which significantly influence the effectiveness of the whole work. This means music is, technically, a non-linear system.”

This article is structured as follows. First, we briefly introduce the main ideas and concepts behind fuzzy logic and its application, with an emphasis on the fuzzy approximation theorem and fuzzy inputs as latent spaces. Second, we conduct an updated survey of the utilization of fuzzy logic in musical applications. Third, we present the Fuzzy Logic Control Toolkit (FLCTK), a set of tools to generate musical content in the MaxMSP real-time sound synthesis environment. Fourth, we provide detailed examples of applications in sound synthesis, algorithmic composition, and many-to-many musical mappings. Fifth, we discuss some compositional aspects of *Incerta*, an acousmatic multichannel composition done in MaxMSP with the FLCTK. Finally, some conclusions and future lines of work are presented.

2. FUZZY LOGIC

Zadeh (1965) introduced the concept of fuzzy sets, which are different from standard sets in the sense that they operate with *multi-valued logic*. Compared to other, perhaps more popular, artificial intelligence techniques, fuzzy logic is simpler and more flexible, making it a very appealing tool for musical applications. Indeed, fuzzy logic systems have found applications in a great multiplicity of fields, notably engineering and control applications (Kosko, 1993; Klir and Yuan, 1995), but also in areas apparently unconnected, such as data analysis (Bandemer and Gottwald, 1995), economics, business, and finance (Von Altrock, 1997), sociology (Dimitrov and Hodge, 2002), or geology

(Demicco and Klir, 2004). Fuzzy logic algorithms can be easily found in everyday popular objects, such as cameras, camcorders, or washing machines, but also on unmanned vehicles, such as trains.

2.1. The Fuzzy Principle

Kosko (1993) coined the phrase *everything is a matter of degree* to emphasize a key element of fuzzy logic theory. In fuzzy logic, inputs and outputs are fuzzified, meaning that their values belong in varying degrees to several fuzzy sets. For example, if we consider the sound intensity range of 30–120 dB, and we want to determine whether a given intensity is low, medium, or high, does a value of 90 dB correspond to a high intensity? As it is closer to 120 than to 30 perhaps, but there are other values which are higher in intensity. Therefore, instead of assigning only one label to it, it is not a bad idea to consider a fuzzified version of this concept, one in which this particular value belongs in different degrees to both the medium and high intensity labels. In consequence, fuzzy sets are not exclusive, they allow *partial membership* of its elements. Unlike traditional crisp logic, where elements belong or do not belong to a particular set, in fuzzy logic an element of the set can be a member of it only partially. In this way, fuzzy logic handles uncertain terms and partial values of truth. Elements are not entirely black or white; they can acquire any shade of gray. Mathematically, this implies membership values between 1 and 0.

2.2. Fuzzy Sets

As we previously stated, a fuzzy set contains members to some degree (Kosko, 1993). Let F be a fuzzy set with an universe of discourse $X = \{x\}$, defined as the mapping $\mu_F(x): X \rightarrow [0, \alpha]$. The universe of discourse is the range of all possible real scalar values of some measurement or items of information that we want to fuzzify. This mapping assigns to each x a value in the range $[0, \alpha]$. When $\alpha = 1$ the set is called *normal*. A fuzzy set contains a distribution, also called *membership function*. When a distribution is of zero width, the membership function collapses to a singularity, which corresponds to the traditional case of a crisp set. If these singularities can only have one of two possibilities, they perform binary logic. μ_F is called the grade of membership or *degree of truth* of x . Fuzzy sets, although usually modeled after triangular or Gaussian distributions, can adopt any form, and no shape has been proven to be the best (Mitaim and Kosko, 2001).

2.2.1. Fuzzification and Defuzzification

Fuzzification and defuzzification are critical operations in fuzzy theory, as both of these operations connect the fuzzy set domain and the real value scalar domain (Roychowdhury and Pedrycz, 2001). Methods and techniques for fuzzification and defuzzification are an active line of research, and several approaches are constantly proposed in the literature. We will now illustrate one of the simplest strategies for fuzzification. **Figure 1** shows the fuzzification of the physical variable “intensity” which is often associated with loudness. Employing fuzzification, a variable or concept can be classified into one or several fuzzy sets. In this particular case, there are three fuzzy sets to which

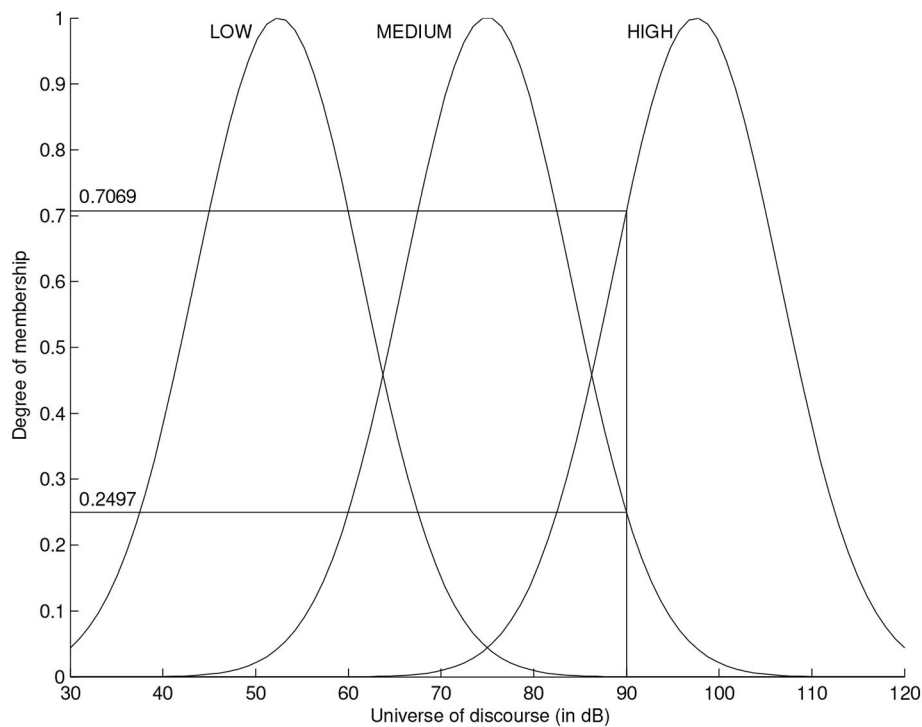


FIGURE 1 | Fuzzification of the concept “intensity.” The intensity level in dB is fuzzified into three fuzzy sets, labeled LOW, MEDIUM, and HIGH. The crisp value of 90 dB belongs to both the HIGH and MEDIUM sets in different degrees.

“intensity” can be classified into, denoted “LOW,” “MEDIUM,” and “HIGH.” The membership functions of these fuzzy sets are Gaussians and the universe of discourse X contains intensities between 30 and 120 dB. In this example, a 90 dB intensity level belongs 70.69% to the fuzzy set “HIGH,” 24.97% to the fuzzy set “MEDIUM” and 0% to the fuzzy set “LOW.”

2.2.2. Operations on Fuzzy Sets

Operations can be defined for fuzzy sets in the same way they are defined in traditional set theory and also in several different ways. The most important fuzzy operators and the way they are typically defined are:

- **Complement.** $\bar{\mu}(x) = 1 - \mu(x), x \in X$. The complement groups all the elements that do not reside in the set $\mu(x)$.
- **Scalar product.** $\mu(x) = S \cdot \mu_1(x), x \in X$. A fuzzy set can be multiplied by a scalar S .
- **Power.** $\mu(x) = [\mu_1(x)]^m, x \in X$. The power operation elevates a fuzzy set to a particular number m . The case $m = 2$ is known as the *concentration* of a fuzzy set.
- **Union.** $\mu_{\cup}(x) = \mu_1(x) \vee \mu_2(x) \vee \dots \vee \mu_n(x) = \max(\mu_1(x), \mu_2(x), \dots, \mu_n(x)), x \in X$. The union of two or more fuzzy sets joins all the elements of the universe of discourse that belong in some degree to any of those sets. This operation can be done with any triangular co-norm. In this particular implementation, we unite fuzzy sets by selecting the maximum values among them.

- **Intersection** $\mu_{\cap}(x) = \mu_1(x) \wedge \mu_2(x) \wedge \dots \wedge \mu_n(x) = \min(\mu_1(x), \mu_2(x), \dots, \mu_n(x)), x \in X$. The intersection of two or more fuzzy sets extracts all the elements of the universe of discourse that belong in some degree to all of those sets. This operation can be done with any triangular norm. In this particular implementation, we unite fuzzy sets by selecting the minimum values among them.

In set theory, both the intersection and union operators produce one set. This is also known as aggregation. In the case of crisp logic, the only way to aggregate one or more sets is by these two operations. However, in the case of fuzzy sets, aggregation can be achieved by several averaging operations, some of which are not necessarily symmetric. For example, it would be possible to specify different weights for each fuzzy set involved (Belohlavek and Klir, 2011). This type of aggregation seeks the averaging of several fuzzy sets into one, and it is not to be confused with the process of rule aggregation, which will be discussed shortly.

2.3. Fuzzy Systems

Fuzzy systems are model-free estimators, they estimate input-output functions where the inputs are fuzzified and the outputs defuzzified. They estimate a function, and can approximate one with any degree of accuracy, without an underlying mathematical model relating inputs to outputs. Fuzzy systems learn from experience codified into numerical or even linguistic data (Kosko, 1992). A general fuzzy system consists of a rule base, an inference

engine, and fuzzification and defuzzification stages (Klir and Yuan, 1995). It operates repeating a cycle of three steps:

1. Fuzzification. Input variables are converted into fuzzy variables.
2. Fuzzy inference engine. The fuzzified measurements are evaluated by the rule base, resulting in one or several fuzzy rules describing the universe of possible actions.
3. Defuzzification. The fuzzy outputs are converted into a single value or a vector.

2.3.1. Fuzzy Rules and Inference

One of the goals of fuzzy logic is to emulate the way humans reason, which is typically just by some imprecise rules and common sense. Most of the decisions humans take can be modeled after computer-like *if-then statements*, based on expert knowledge or common sense. However, fuzzy rules can also be learned from data (Kosko, 1992). One example is the FUZZEX algorithm which can learn rules from a corpus of data mapping inputs to outputs, in the same fashion that a neural network does (Finn, 1999).

Formally, a fuzzy rule is a conditional of the form IF X is A THEN Y is B , where A and B are fuzzy sets (Kosko, 1993). Typically, fuzzy systems contain a large rule base and the method by which the computation of the contribution of each rule is achieved is known as *aggregation*. There are two main aggregation strategies, one connecting the rules with AND operators, and another where they are connected by OR directives (Ross, 2010). In the first, the aggregated output is obtained by the intersection of all the individual rules, while in the latter the output is aggregated by the union of the contribution of all rules.

The process of aggregating all the rules in parallel is called fuzzy inference. Different inference methods can be employed depending on the task in question. One of the most popular is the Mamdani method, proposed in 1975 by Mamdani and Assilian (Ross, 2010). Several variants of this method exist, for example the min-max method where a fuzzy rule would have the form:

$$\text{IF } x_1 \text{ is } A_1^k \text{ AND } x_2 \text{ is } A_2^k \text{ THEN } y^k \text{ is } B^k \text{ for } k = 1, 2, \dots$$

where A_1^k and A_2^k are fuzzy inputs and B^k is the desired output. For r disjunctive fuzzy IF-THEN rules, the aggregated fuzzy output will be:

$$\mu_{B^k}(y) = \max_k [\min[\mu_{A_1^k}(\text{input}(1)), \mu_{A_2^k}(\text{input}(2)), \dots]]$$

for $k = 1, 2, \dots, r$

After inference, there comes defuzzification, a process to which several approaches exist (Ross, 2010). One of the most used ones is the centroid method, where the center of mass of the aggregated fuzzy output is computed as a scalar value.

Another widely used inference method is the TSK or Sugeno method proposed by Takagi, Sugeno, and Kang. In this method,

two inputs x, y and one output z are associated by a rule of the form:

$$\text{IF } x \text{ is } A \text{ AND } y \text{ is } B \text{ THEN } z \text{ is } z = f(x, y)$$

where $z = f(x, y)$ is a non-fuzzy function of the inputs x and y (Ross, 2010). This inference function can be any function that describes the output of the system within the fuzzy region that the particular fuzzy rule encompasses. One advantage of the TSK method over the Mamdani strategy is that it requires less computation time by avoiding the defuzzification stage, which can be computationally challenging if the rule base is large enough.

2.4. Fuzzy Logic vs. Deep Learning

Both neural networks and fuzzy systems are numerical frameworks used to estimate input-output functions without an underlying mathematical model of how inputs relate to outputs. In this sense, they are model-free estimators (Kosko, 1993). Both approaches have been proven to be universal approximators for any non-linear function to any degree of accuracy (Kosko, 1994; Ying, 1998).

Neural networks excel at learning and adapting under uncertainty scenarios. It is no surprise then that deep learning has emerged as perhaps the most important branch of AI due to its unprecedented capacity of learning data in an unsupervised manner and superb results in tasks of classification and estimation. However, due to the high complexity of some network architectures and the large amount of data that is needed for training, it is very hard to understand what is being learned or even why some systems work. Indeed, a large amount of current research in deep learning seeks to understand what are networks learning. In short, neural networks can do amazing jobs at the cost of inaccessible knowledge.

Fuzzy logic, on the contrary, is all about knowledge representation. It is very clear what is being learned and represented as all knowledge is encoded in the rules of the fuzzy system. There is no major mystery as to why fuzzy systems work. And these systems can also operate under uncertainty and require almost no data, besides a couple of examples or common sense to derive the rules from.

In the case where the number of inputs to a fuzzy logic system is significantly less than the number of outputs, then the system mimics the behavior of a latent space, in the sense that its rules, which depend only on a few inputs, are a compact representation of the dynamics of the many output parameters. However, the main difference with the typical latent spaces that can be found in auto-encoders and other types of neural networks is that this fuzzy latent space is constructed based on simple rules and it is not inferred or learned from data. In other words, this is a well-understood and totally determined latent space.

Fuzzy systems offer nice opportunities for creative applications, as they are able to mimic some characteristics of human reasoning. The parallel calculation of fuzzy rules generally reduces the calculation time compared to traditional deep learning techniques or mathematical approaches. Knowledge is encoded employing fuzzy rules that can easily be specified

as IF-THEN statements, with simple linguistic terms, using common sense, and they can be easily adjusted.

3. FUZZY LOGIC FOR MUSICAL APPLICATIONS: A SURVEY

Back in 2001, Landy (2001) signaled fuzzy logic as one of the important potential domains of development for the future's music world. We believe that fuzzy logic has not yet reached its full potential in the service of musical purposes, due perhaps to the exponential growth and development of deep learning and other AI techniques. However, fuzzy logic has found its way into several domains related to audio applications and musical creative activities, as we report now.

3.1. Acoustics, Psychoacoustics, and Digital Audio Processing

Demichelis et al. (1983) proposed an automatic recognition method of plosive consonants, by using a fuzzy model of the human speech perception and integration mechanisms. The rules of the system were designed taking into consideration prior research done by psychologists and phoneticians in the generation and perception of these types of consonants by the human brain, which can be characterized by several acoustic cues. The authors found out that the performance of the system drastically improved when more significant cues were added to the rule base.

Civanlar and Trussel (1986) designed an audio signal restoration method based on a fuzzy system that models a priori information. The authors combined exact knowledge about the signal to be restored with partial and incomplete information. The original signal and all reasonable solutions belong to a high degree to the feasibility set of possible solutions, while rejected solutions have lower membership degrees. The measure of this set gives an approximation for the quality of the solution. This method was shown to be successful in many restoration situations where other conventional techniques to that date had failed.

Kostek (1999) designed a fuzzy controller of a pipe organ. The system links the opening procedures of the pipe valves to the manner of depressing the keyboard. The proposed solution utilizes a velocity-sensitive MIDI keyboard, connected to a computer with a special fuzzy microprocessor card and a buffer to control an array of electromagnets. These, in turn, control the pipes. Inputs to the fuzzy controller were key number and velocity. The output can belong to the following membership functions: low current, medium current, and high current. The system produced one fuzzy output, associated with the current applied to the electromagnet coils, based on eleven rules.

Breining (2001) developed a fuzzy step-gain control procedure for adaptive filters to be used in acoustic echo cancellation situations. Many step-gain estimators become unreliable under adverse environments. This fuzzy logic-based controller used a step-gain estimator combined with a double-talk detector, resulting in a highly convenient and relatively simple method compared with traditional alternatives.

Meng et al. (2002) designed an analytic method to extend the sound impulse response of a room, using knowledge from extrapolation theory for band-limited sound signals. The evaluation of the method was conducted using a fuzzy clustering algorithm. The authors use similarity perceptual judgment tasks to compare the similarity between the extrapolated real impulse responses. Typically these kinds of perceptual measurements estimate a similarity matrix with methods, such as Kruskal's multidimensional scaling. However, considering that the terms similar or dissimilar are not entirely crisp concepts, fuzzy logic was added to transform the similarity matrix into a fuzzy clustering matrix.

Malcangi (2008) constructed a fuzzy audio-pattern recognition algorithm targeted for usage in very low-cost embedded systems, to automate human-machine interaction. This system was built on top of feature extraction algorithms and a rule base constructed by a self-learning process. The fuzzy logic recognition engine used membership functions according to the spectrum of a particular audio frame to be recognized. In consequence, if the audio pattern was, for example, a vocal utterance, then the membership function would be modeled according to the spectrum envelope of the stationary components of the speech. In such a manner, a set of membership functions covering all the stationary speech sounds to be recognized must be generated. By using a different dictionary of membership functions, other kinds of acoustic signals can be recognized by such a system.

Gonzalez-Inostroza et al. (2015) proposed a fuzzy-logic based equalizer for musical genres, by incorporating significant audio descriptors that allow for the recognition and description of diverse musical genres. These descriptors feed a fuzzy logic inference system, whose outputs are the required equalization levels for each frequency band. The rules of the system were derived from the analysis of a well-known music database encompassing ten different musical genres. Their approach works for songs that exhibit multiple genre characteristics, that are difficult to classify into one category, or that mix genres.

3.2. Music Listening, Emotion, and Analysis

Milicevic (1999) aimed to aid composers to create more appealing music for a wider public. Assuming that composers usually seek a positive cultural response to their music, the authors built a fuzzy adaptive and emotion-based music system that can reduce the internal fuzzy entropy of the compositions, making them more appealing to people and able to produce positive emotional responses while listening to it. They tried their system in the special case of computer music.

Friberg (2005) was able to use a fuzzy system for the analysis of the emotional expression and body movements in musical performances in real-time. Parameters, such as articulation, tempo, intensity, and motion descriptors were used as inputs to a fuzzy mapper able to translate these variables into one of three possible outputs: happiness, sadness, and anger, all related to emotion. The rule base was constructed considering qualitative data from former studies.

Yang et al. (2006) also developed a music emotion classification system based on fuzzy logic. They declare that

“due to the subjective nature of human perception, classification of the emotion of music is a challenging problem,” as feeling and emotion states provoked by music could be unequal to different people. Their approach estimates the likelihood that a given segment extracted from a song belongs to a particular category of emotion. Their system can measure emotional strength to track the variation of different musical emotions provoked by a song.

Maristany et al. (2016) have done soundscape quality analysis by means of fuzzy logic. They conducted a comparative analysis, based on surveys and psycho-acoustic estimations, in open locations around the city of Córdoba in Argentina. They found out that there is a non-linear relationship between these indicators and the audible qualities of the spaces. Fuzzy logic emerged then as a suitable tool to model this non-linearity, confirmed by the model performance and the perceptual outcomes from the users. The authors found that this approach can be applied not only to soundscapes, but in other studies where perception must be confronted with objective measurements.

Hasanzadeh et al. (2019) constructed a fuzzy cascade model designed to predict the emotional content of pieces of music using electroencephalographic (EEG) signals. Users listened to musical excerpts while their emotional appraisal response was estimated as a value along two emotional axes (valence and arousal). The proposed fuzzy model consists of parallel cascades with each cascade containing a single multi-input/single-output fuzzy logic-based system. The authors compared this approach to several alternative methods, including recurrent neural networks, and concluded that the fuzzy approach exhibited the best performance.

Kasinathan et al. (2019) developed a music recommendation system based on a fuzzy inference engine that considers user activities and emotion as part of the recommendation parameters. The authors describe that their fuzzy inference system can decide on music recommendations based on the user's music listening habits as well as expert knowledge about music genres and their effects on humans. The user's preference data is fed into the fuzzy system to obtain a decision that returns a score corresponding to the recommended music track. The top ten music tracks with the highest recommendation score are provided as recommendations for the user.

3.3. Music Information Retrieval and Performance

Orio and Pirro (1998) coded gestures made during interactive musical performances in real-time by a neuro-fuzzy system. One of the basic contributions of their work is using only two different levels for the codification of human gestures. These levels usually carry a significant amount of information about the performer's intentions. But additional information is carried by nuances of the gesture, and these can also be analyzed, capturing the detailed performance of each gesture. This two-level approach was applied in a system for interactive piano performances. Nuances were analyzed in terms of linguistic labels. This is where fuzzy logic plays an important role, given

its suitability to handle semantic expressions. Depending on the kind of desired performative nuances, fuzzy controllers were developed. Loudness and tempo are used as inputs, and the system then calculates the level of, for example, “urgency” of the musician. This information is later used by the system to musically respond in real-time to the human performer.

Usa and Mochida (1998) used a fuzzy system in the simulation of the gestures of an orchestra conductor. The system is capable of recognizing some of the most common conducting elements of conductors. In particular, the beat recognition system was built on top of a fuzzy model of actual orchestra musicians' recognition.

Liu and Huang (1998) implemented a system that discriminates news reports from broadcast ads or music in news programs based on the information contained in the audio signals. Four features were extracted from the audio data. Both a simple threshold and a fuzzy classifier were implemented to classify the audio data. In the case of the fuzzy classifier, descriptors were associated with fuzzy sets and the influence of each feature was combined to obtain the final classification decision. Results reported an improvement using the fuzzy classifier compared to the threshold-based system.

Weyde and Dalinghaus (2001) recognized rhythmic patterns with a neuro-fuzzy system, which determined grouping and group relations between two sequences (comparison) or within one sequence (analysis). The system makes use of knowledge and learning from data and it is open for the integration of different features and rules. The system defined by the rules can be trained to prefer certain interpretations over others by example.

Liu et al. (2002) propose a fuzzy system designed to classify and retrieve audio clips, inspired by the fuzzy nature of human perception. Various extracted features were used as input to a fuzzy system, whose outputs belonged to two types of classes. The rule base was constructed from characteristics extracted from the clips. The results show that the system can discriminate between speech and music and that it can be extended for the classification of more types of audio clips.

Monti and Sandler (2002) developed a system able to translate audio directly into MIDI data. The system contains a fuzzy inference system that achieves polyphonic note recognition as a part of the overall process. First, spectral peaks from a spectrogram are selected by the algorithm. Harmonically related peaks are grouped into note candidates. If a note candidate receives a good rating, it is transformed into a note hypothesis. Finally, the hypotheses that survived an activation time threshold become active notes. The fuzzy inference system takes the spectral peaks that were not selected into the note continuation process and creates new candidates. The new candidates are then evaluated by the inference system to become note hypotheses. The membership functions used in this system classify notes into low, middle, and high and take into consideration pitch, harmonic rate, and relative energy.

Leon and Liern (2010) modeled musical notes and tuning systems as fuzzy entities to integrate tuning theory and musical practice. The authors were able to combine different tuning systems into a simpler fuzzy model that reflects both the idea of proximity between different notes and whether their

configuration, in terms of a specific tuning system, is sufficiently similar for practical musical purposes.

Knudsen et al. (2019) propose that “to collaborate and co-create with humans, an AI system must be capable of both reactive and anticipatory behavior.” With this objective in mind, the authors considered a mixed human-robot duo, more precisely a piano and a virtual robotic drummer, and they designed a fuzzy logic-based system to determine the performance features of the drummer as a function of what the human pianist performs. While the system exhibited only limited anticipatory capabilities, the behavior of the drummer was judged to be satisfactory by musicians in initial evaluation experiments.

3.4. Musical Composition and Generation

Lee and Wessel (1993, 1995) were among the first researchers to incorporate a fuzzy reasoning system into the MAX real-time music programming language. They labeled their system MaxFUZ, and it implemented fuzzy variables, sets (limited to trapezoidal shapes), and both Mamdani and Sugeno rules. This was the first interactive fuzzy system that worked in real-time inside MAX to our knowledge.

Almost at the same time, Elsea (1995) utilized fuzzy logic features to tackle problems in the analysis and composition of music. In his work, pitches and dynamics were represented as fuzzy sets and fuzzy reasoning is used to produce chord inversions and sequence of chords. These ideas were implemented in software as external objects for Max/MSP, called L-objects, which provide fuzzy operations and manipulation of fuzzy sets.

Kiseliova et al. (2005) developed an interpretation fuzzy algorithm, based on top of a rule base designed by an experienced pianist. Their approach relies on both conventional and more advanced information decision strategies. This system, given a known piece of music, creates a MIDI-based interpretation of the piece. Their general objective is to transform a mechanical performance of a piece of music into a much more human-like interpretation by applying the knowledge of an expert performer in the form of a fuzzy rule base.

Cádiz (2004, 2006a) proposed a fuzzy logic system to convert visual information into sonic information and vice-versa. This model is useful to generate audiovisual content, given either the visual or sonic content in advance. Parameters in one domain are fuzzified and fed into a fuzzy inference engine that generates parameters in the other domain. This fuzzy mapping is inspired by the ideas of isomorphism and synaesthesia. Isomorphism determines whether two different modalities can be mapped onto each other based on the fact that perturbations into one of them consistently cause changes in the other, while synaesthesia occurs when a stimulation in one sensory modality automatically provokes a perceptual outcome in a secondary sensory modality when there is no direct stimulation to it.

Yilmaz and Telatar (2009) identified key areas where fuzzy logic can be used for the composition and generation of music: harmonization, orchestration, improvisation, and composition. They propose to focus on the harmonization with constraints as a way of tackling these three areas. In particular, they proposed a fuzzy feedback decision system designed to perform

accompanying tone generation dynamically. They applied this system to the particular problem of note-against-note two-voice counterpoint (Yilmaz and Telatar, 2010). Their method considers membership functions and rules that mimic some known rules of music theory, and their implementations provide feasible procedures when compared to those of established music theory.

Suiter (2010b) devised a conceptual framework for composing expressive music based on fuzzy logic, aimed toward reducing the number of musical decisions that a composer must make at the micro-level and focusing on those that contribute to expressiveness the most. A fuzzy system is used to trace the trajectory of all musical details of a composition, encompassing each element and their combinations.

López-Ortega and López-Popa (2012) developed a two-dimensional recursive fuzzy method assisting composition for MIDI-based musical works based on fractal structures. In their approach, notes evolve according to a particular fractal trajectory. Tempos and duration can remain fixed or they also can follow the fractal structure. Additionally, the set of produced pitches are translated into tones belonging to a previously determined musical scale.

Kuo et al. (2015) created a real-time emotion-based music accompaniment system through a fuzzy logic tempo controller, and an additional genetic evolutionary melody generation system. Harmonic chord progressions were generated using known music theory rules. For the fuzzy tempo controller, they used a range of 60 to 180 beats per measure, and the fuzzy output is used to adjust the current tempo compared to a target one.

Lucas et al. (2017) developed a method for representing human emotions in the context of human-machine musical composition based on fuzzy logic. A knowledge base of human-produced melodies and human-labeled emotions associated with them, in the form of a Markov chain process, is used to generate new melody patterns, which are later classified into emotions by a fuzzy classifier. These new melodies can be of later use to compose music with specific emotional targets in mind.

Guliyev and Memmedova (2019) modeled some compositional decisions as the requirement to construct relationships between controllable elements in music, in particular pitch, duration or amplitude, and a consequent evaluation. They established these relationships as sets of IF...THEN fuzzy rules with antecedents, the input parameters, and a consequent, the evaluation of the generated musical output. This method can be thought of as a “preference ordering” of the attributes of a particular piece of music.

3.5. Sound Synthesis

Miranda and Junior (2005) introduced a novel Markov fuzzy model for granular synthesis. While Markov chains control the temporal evolution of the sound, their fuzzy system defines the granular structure of the sound. In this sense, this method extends the idea of a grain into the concept of a fuzzy grain. A fuzzy grain contains several fuzzy parameters: frequency, amplitude, and membership values of each Fourier partial of the grain, or in other words, its weighted harmonic content.

Schatter et al. (2005) proposed a graphical user interface for the generation of electronic sounds with a synthesizer.

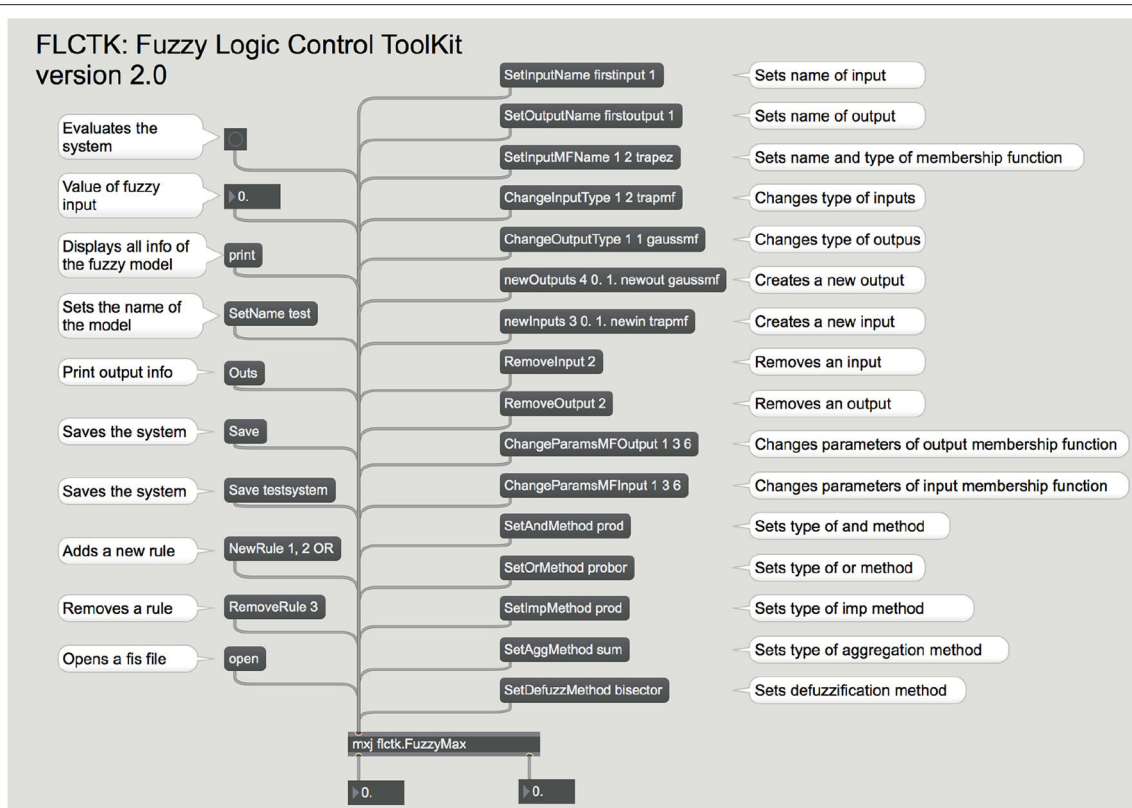


FIGURE 2 | Help screen of the Fuzzy Logic Control Toolkit (FLCTK) in the MaxMSP environment. Inputs, outputs and rules can be added and removed on-the-fly.

Twenty-three aural parameters were reduced to five parameters controlling the visual metaphor utilizing a fuzzy logic controller. The system allows knowledge-based mappings that are adaptable to each user. There are two modes of operation: in the manual operation mode, the system is used to record the parameter-input of the user, followed by the generation of fuzzy controllers. In the automatic mode, the system has to find parameter combinations, employing genetic algorithms.

Cádiz (2006b) has proposed an approach for the compositional control of computer music based on fuzzy logic. In this case, the control of the compositional process derives from the fuzzification of the synthesis parameters of interest, while the rule base can be specified at will by the composer, according to his objectives. The author provides five different applications of this type of synthesis control in the context of spectral synthesis, physical modeling, granular synthesis, particle-based synthesis, and audiovisual composition, exemplifying a significant number of situations in which such an approach gives suitable results.

Lucas and Pelaez (2019) implemented a granular synthesis method based on harmonic rules and fuzzy logic. In this method, each grain is positioned in a two-dimensional space arranged in the same fashion as the circle of fifths. A fuzzy logic prioritization algorithm is used to order the grains in the vicinity of a particular performing area inside this two-dimensional space. The algorithm takes frequency and energy levels of each grain

in the vicinity as inputs and produces a prioritization index as a result.

4. THE FUZZY LOGIC CONTROL TOOLKIT

The Fuzzy Logic Control Toolkit (FLCTK) (Cádiz and Kendall, 2006; Cádiz and Gonzalez-Inostroza, 2018) is a collection of software tools implemented in MaxMSP¹, a sound synthesis environment that allows for the design and usage of a generic fuzzy inference system in real-time. An important feature of this software is its capability to import and export fuzzy systems in the *fis* file format, a popular fuzzy logic specification used by MATLAB's Fuzzy Logic Toolbox². This common shared file format allows a user to design and troubleshoot a complete fuzzy system in MATLAB, export it as a *fis* file, and then import the same system into the FLCTK or vice-versa.

Figure 2 displays a screen-shot of the MaxMSP help patch of the FLCTK's external `flctk.Fuzzy` with all its options. This external object can load a fuzzy system or create one on-the-fly by sending messages to it. In this example, details of the fuzzy system in use are displayed in the Max window. Some messages can select specifics for implication, aggregation, fuzzification and defuzzification, number of rules, their weights, and whether AND

¹<http://cycling74.com>

²<https://www.mathworks.com/help/fuzzy/index.html>

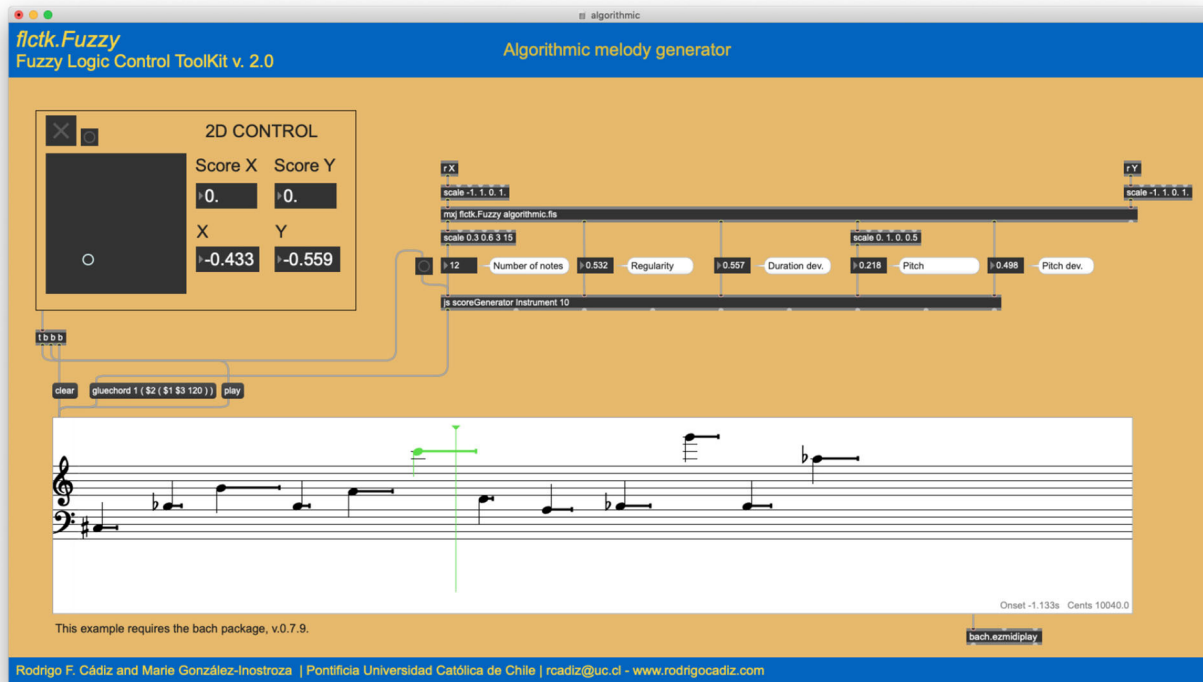


FIGURE 3 | Algorithmic melody generator example 1. The 2D coordinates $[-0.433, -0.559]$ of the latent space generate a 12-note melody with a large range of pitch variations.

or OR operators should be used. All fuzzy inputs and outputs can be defined with either triangular or Gaussian membership functions, and labels can be created for each of them.

This toolkit is very simple to use inside MaxMSP and a complete fuzzy system can be designed by sending the appropriate messages to the `fltk.Fuzzy` object. This object can be created based on an existing `fis` file designed off-line in MATLAB. In this case, the path to the `fis` file should be specified. Another option would be to initialize the object with several inputs and outputs, in which case the external will create all necessary fuzzy variables with a standard configuration using Gaussian membership functions and inputs and outputs consisting of five fuzzy sets each. This configuration can be altered after the system was created by sending modifier messages. Once inputs and outputs are created, rules can be added one by one by sending a message specifying the inputs and outputs involved in each rule, the aggregation method, and specific weight for each rule. Rules can be deleted and tested on the fly, to customize the system's behavior.

The FLCTK can be downloaded from its github website at <https://fltk.github.io/>. The package contains the source Java code, compiled code, help files and video examples, some of which are detailed below. Also, a standalone version, written from scratch in C++ and based on the Open Sound Control protocol (Wright et al., 2003) is in the works at the time this article was published.

5. EXAMPLES

We now provide four examples, developed by the author using the FLCTK, that illustrate the power of fuzzy logic for audio and music generation, in the specific domains of computer music and algorithmic composition, sound synthesis, and parametric control. These examples are purposely very simple, as they were designed to clearly show the effect of fuzzy logic when applied to very basic ideas. Illustrating videos of each of the examples can be found in the **Supplementary Material**.

For the algorithmic composition and parametric control examples, we utilize a bi-dimensional controller (shown in **Figures 3, 4, 8**) as a very simple control interface. The bi-dimensional controller has a square shape and a pointer (small circle) that tracks the coordinates of the mouse as the user moves it. Both axes have a range of 2.0 (from -1.0 to $+1.0$). The origin $(0,0)$ is located at the center of the square. The controller also accepts pointer coordinates via internal messaging. In this way, the controller can in turn be controlled not only by the mouse but by any kind of two-dimensional process. In the following examples, the coordinates of the controller are fed into custom fuzzy systems designed for each particular case. As these fuzzy systems contain more than two outputs, this controller behaves as a latent space, which is a compact representation of the high-dimensional output parameter space.

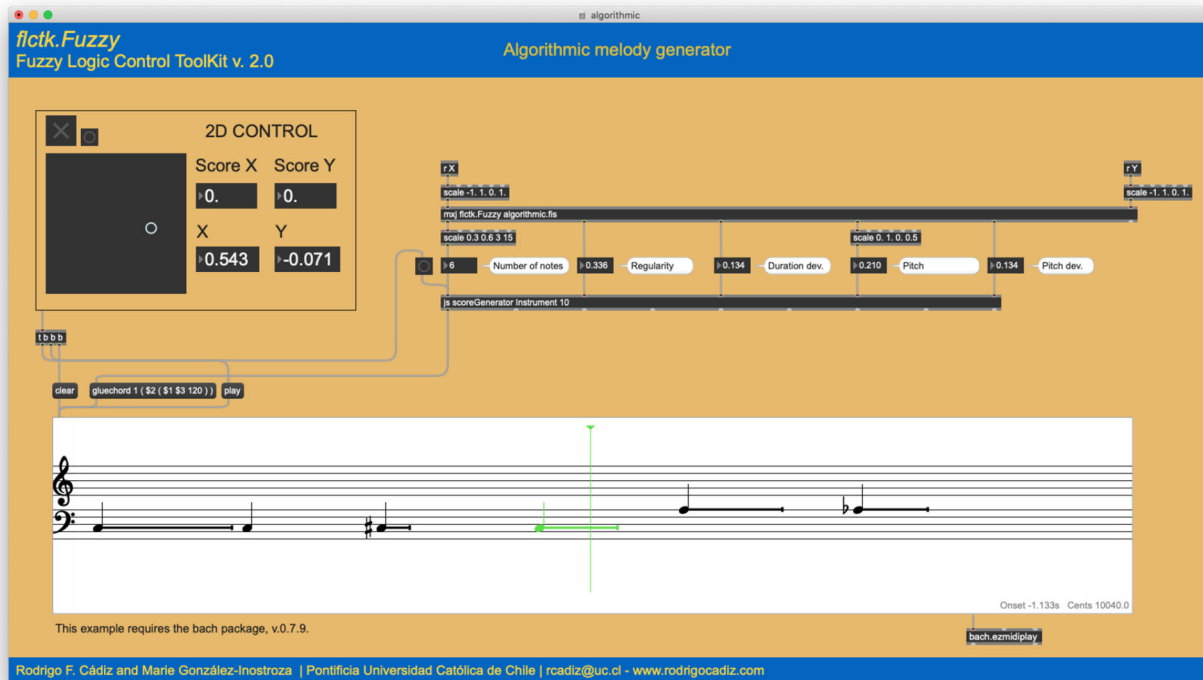


FIGURE 4 | Algorithmic melody generator example 2. The 2D coordinates [0.543, -0.071] of the latent space generate a six-note melody with a rather small range of pitch variation in the lower register.

TABLE 1 | Input and output variables for algorithmic composition example.

| Input variables | Output variables |
|-----------------|--------------------|
| X | Number of notes |
| Y | Regularity |
| | Duration deviation |
| | Pitch |
| | Pitch deviation |

Each one of these variables can belong to three fuzzy sets: *LOW*, *MEDIUM*, or *HIGH*. As the number of inputs is lower than the number of outputs, the input space is a latent space of the output space.

5.1. Algorithmic Composition

Algorithmic composition is simply the use of algorithms to compose music. This is a very common practice in the history of music, as “for centuries musicians have been proposing methods that can be considered as algorithmic in some sense, even if human creativity plays a key role” (Fernández and Vico, 2013). There is a great variety of algorithms that have been proposed for music composition, including simple recursive equations, chaotic systems, re-writing systems, and many others (Nierhaus, 2009; Edwards, 2011).

Algorithmic composition using fuzzy logic is proposed here as another alternative. Fuzzy logic, as we have previously discussed,

is flexible enough to be applied to many different composition-related contexts and situations. This particular example consists of the generation of a very simple melody, where the number of notes, their pitch range, and their duration are determined by a fuzzy inference system. The inputs to the system are the two outputs of the aforementioned bi-dimensional controller. The outputs are the number of notes, a regularity factor, a duration deviation factor, pitch, and a pitch deviation factor, as specified by **Table 1**. Each one of these variables can belong to three fuzzy sets: *LOW*, *MEDIUM*, or *HIGH*. As there are fewer inputs than outputs, the input space is a latent space of the output space.

This system contains nine fuzzy rules, detailed in **Table 2**. All rules have the same weight and are connected by AND operators. A “-” indicates that the value of the fuzzy variable can be anything. As shown in **Figure 3**, given the coordinates [-0.433, -0.559] of the latent space, the system generates a twelve-note melody with a large range of pitch variations. Another point in the latent space will produce a different output, as is displayed in **Figure 4**, where the coordinates [0.543, -0.071] of the latent space output a six-note melody with a rather small range of pitch variation in the lower register.

5.2. Sound Synthesis

An audio synthesis technique, based on fuzzy logic and the idea of sound particles, is presented as a second example of the application of fuzzy logic for music generation. This technique

TABLE 2 | Fuzzy rules for the algorithmic composition example.

| Rule | Inputs | | Outputs | | | | |
|------|--------|--------|---------------|------------|---------------|--------|------------|
| | X | Y | Num. of notes | Regularity | Duration dev. | Pitch | Pitch dev. |
| 1 | LOW | – | HIGH | MEDIUM | LOW | HIGH | MEDIUM |
| 2 | MEDIUM | – | HIGH | MEDIUM | – | LOW | LOW |
| 3 | HIGH | – | LOW | LOW | LOW | HIGH | LOW |
| 4 | – | LOW | MEDIUM | HIGH | HIGH | – | HIGH |
| 5 | – | MEDIUM | LOW | LOW | LOW | LOW | LOW |
| 6 | – | HIGH | LOW | LOW | HIGH | MEDIUM | HIGH |
| 7 | LOW | HIGH | LOW | MEDIUM | LOW | HIGH | MEDIUM |
| 8 | MEDIUM | HIGH | HIGH | MEDIUM | MEDIUM | LOW | HIGH |
| 9 | HIGH | HIGH | HIGH | HIGH | HIGH | HIGH | HIGH |

All rules have the same weight and are connected by AND operators. A “–” indicates that the value of the fuzzy variable can be anything.

TABLE 3 | Input and output variables for the single particle sound synthesis example.

| Input variables | Output variables |
|-----------------|--------------------|
| Time | Δ Frequency |
| Frequency | Δ Intensity |
| Intensity | |

has been shown to generate complex synthesis parametric trajectories by very simple means (Cádiz and Kendall, 2005). This example consist of a single sound particle (a sinusoidal oscillator) that possesses several fuzzy properties, labeled as time, frequency and intensity. These properties are fed into a fuzzy system that determines the temporal evolution of the particle. Each one of the fuzzy properties consists on several fuzzy sets or membership functions. **Table 3** displays all the inputs and outputs used in the example. Note that as time is included as an input, complex time-dependent behaviors or trajectories can be generated.

The time input variable can belong to seven fuzzy sets, labeled VERY SHORT, SHORT, MEDIUM SHORT, MEDIUM, MEDIUM LONG, LONG, and VERY LONG fuzzy sets. The frequency and intensity variables can belong to five sets: VERY LOW, LOW, MEDIUM, HIGH, or VERY HIGH. The outputs of the system are a change in both frequency and intensity. This means that, in this case, the fuzzy system is a closed-loop system, a very common design for automatic control applications. The outputs at each time step are used to recalculate the current frequency and intensity of the particle. The fifteen rules of this system are shown in **Table 4**. All rules have the same weight and are connected by AND operators. A “–” indicates that the value of the fuzzy variable can be anything. It is important to recall that in this example the fuzzy system is dependent on time. As time progresses linearly, the output variables frequency and intensity exhibit a highly non-linear behavior, as it can be observed in **Figure 5**.

This single particle model has been extended to many particles, as described in Cádiz and Kendall (2005). In the

many particle case, two additional fuzzy properties were added: spatial position and charge. **Figures 6, 7** show the frequency and intensity trajectories for a ten-particle system. In the figures, all particles shared the same initial conditions, except for random charges. The trajectories displayed in the figures are quite complex, with very different behaviors as time progresses. Sometimes they behave very chaotically and some other times, in this example most notably in the first 6 s, they follow smooth and apparently non-chaotic but rather well-defined trajectories. Some clustered groups can also be noticed. This kind of behavior is a consequence of the easiness of fuzzy logic to approximate non-linear dynamical systems.

5.3. Parametric Control

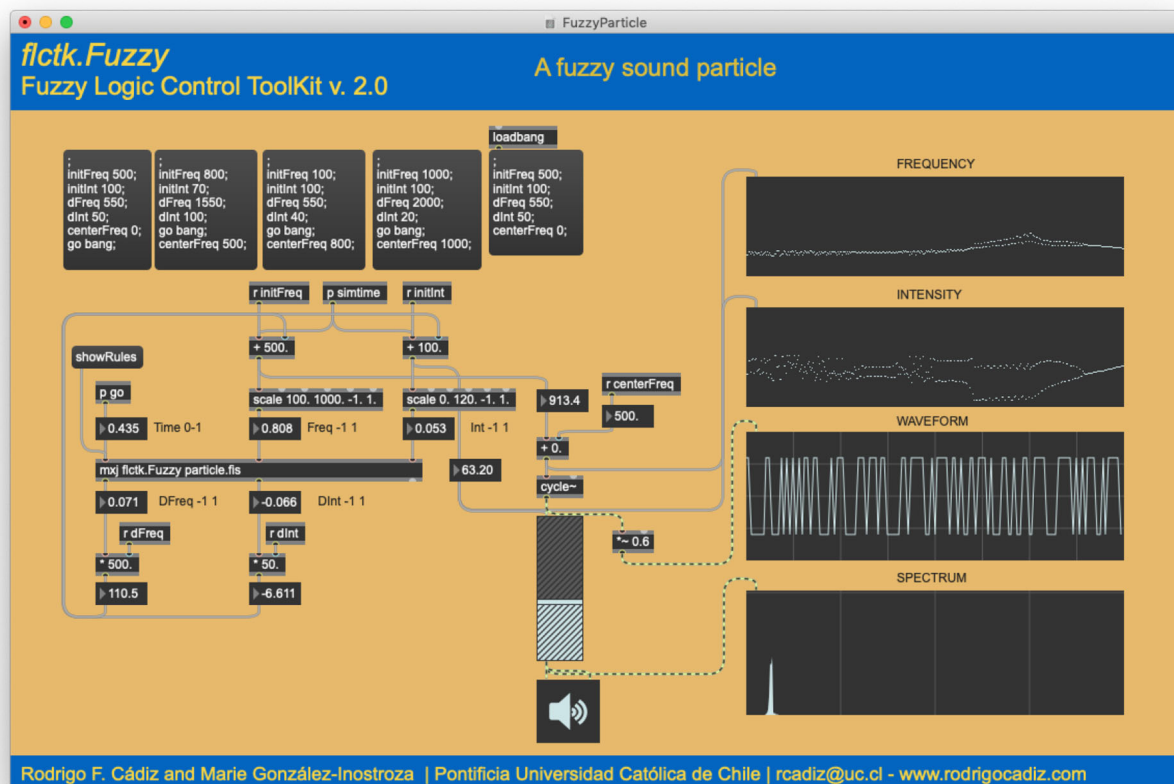
Granular synthesis (Dodge and Jerse, 1997) is inspired by the idea of sound particles or grains, similar in spirit to photons or particles of light. Iannis Xenakis in 1971 and Curtis Roads in 1978 were among the first to suggest granular synthesis as a viable computer music technique for producing complex sounds. This technique generates a high density of very short acoustic events or grains, resembling clouds, with a duration between 10 and 50 ms (Roads, 2004). These grain clouds typically range from several hundred to several thousand events per second. If sinusoidal functions or any pure synthesis methods are used to produce the grains, the technique is called granular synthesis, while if pre-recorded sounds constitute the grain material, people often call that granular processing. This technique often requires the user to control multiple parameters without any clear relation to what they are hearing (Wolek, 2005). This is a situation where a fuzzy logic-based control strategy could be useful.

This example consists on the control of a granulator, whose parameters are determined by a fuzzy inference system. In this specific case, granular synthesis is achieved using the `nw.grainpulse` object, written for Max/MSP by Wolek (2002). This object has five parameters to be controlled. The inputs to the system are the two outputs of the bi-dimensional controller used for the algorithmic composition example. The outputs are the pulse interval, buffer offset, duration, sample increment, and gain multiplier, as specified by **Table 5**. Each one of these variables can belong to three fuzzy sets: LOW, MEDIUM,

TABLE 4 | Fuzzy rules for the single particle sound synthesis example.

| Rule | Inputs | | | Outputs | |
|------|--------------|-----------|-----------|--------------------|--------------------|
| | Time | Frequency | Intensity | Δ frequency | Δ intensity |
| 1 | VERY SHORT | – | – | VERY LOW | VERY LOW |
| 2 | VERY LONG | – | – | VERY HIGH | VERY HIGH |
| 3 | – | MEDIUM | MEDIUM | MEDIUM | MEDIUM |
| 4 | – | LOW | – | HIGH | MEDIUM |
| 5 | – | HIGH | – | LOW | MEDIUM |
| 6 | – | – | LOW | MEDIUM | HIGH |
| 7 | – | – | HIGH | MEDIUM | LOW |
| 8 | – | MEDIUM | – | VERY LOW | VERY HIGH |
| 9 | – | – | MEDIUM | VERY HIGH | VERY LOW |
| 10 | SHORT | – | – | LOW | LOW |
| 11 | MEDIUM SHORT | – | – | HIGH | LOW |
| 12 | MEDIUM | – | – | LOW | HIGH |
| 13 | MEDIUM LONG | – | – | MEDIUM | MEDIUM |
| 14 | LONG | – | – | VERY HIGH | VERY HIGH |
| 15 | VERY LONG | VERY LOW | – | LOW | – |

All rules have the same weight and are connected by AND operators. A “–” indicates that the value of the fuzzy variable can be anything.

**FIGURE 5** | Particle sound synthesis example. In this case the fuzzy system is dependent on time. As time progresses in a linear fashion, the output variables frequency and intensity exhibit a highly non-linear behavior.

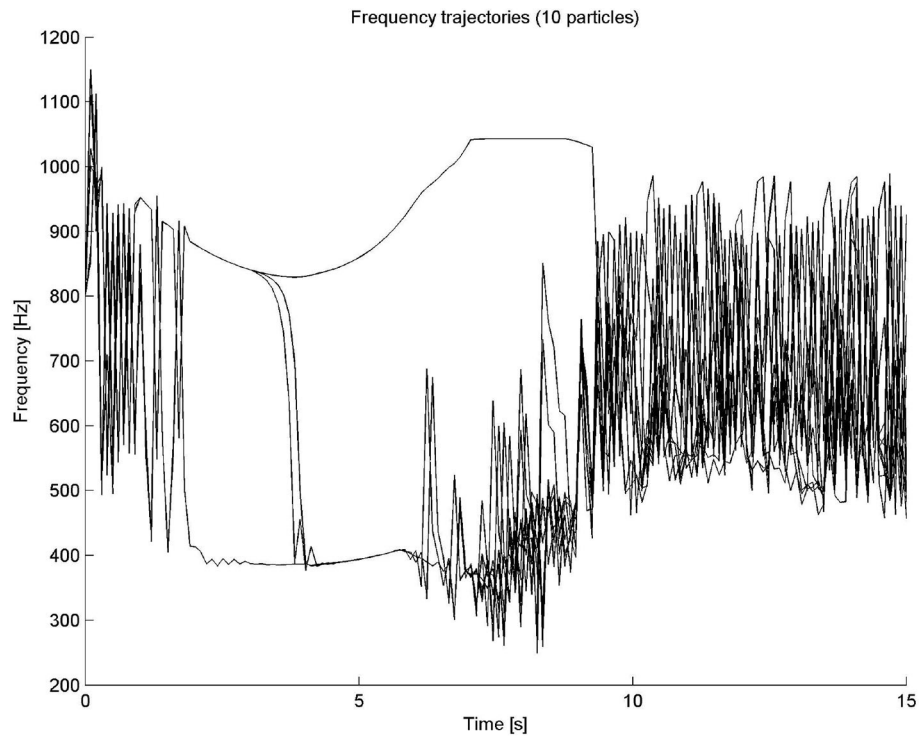


FIGURE 6 | Frequency trajectories in time for a 10-particle system. As it can be seen, highly complex behavior can be generated with a few simple if-then rules.

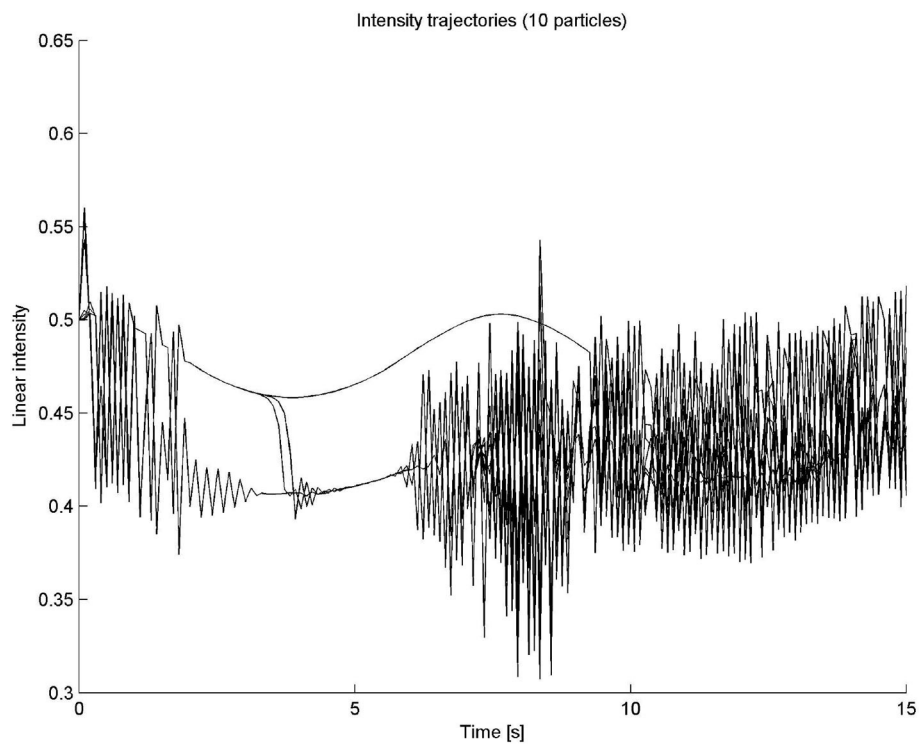


FIGURE 7 | Intensity trajectories in time for a 10-particle system. As it can be seen, highly complex behavior can be generated with a few simple if-then rules.

or HIGH. As there are more outputs than inputs, the input space is a latent space of the output space, as in the algorithmic composition example.

This system contains nine fuzzy rules, detailed in **Table 6**. All rules have the same weight and are connected by AND operators. A “–” indicates that the value of the fuzzy variable can be anything. Inputs and outputs of the fuzzy system are displayed on the right of **Figure 8**. As the latent space is explored in both the X and Y directions, the output variables exhibit different non-linear behavior. This allows the parametric control of five synthesis parameters with only two abstract parameters. The proposed fuzzy system effectively acts as a latent space generative model, one that can translate points from a two-dimensional parameter space into a five-dimensional space that acts directly on the sonic output, according to the nine rules of the system.

5.4. Many-to-Many Parametric Control

In computer music, sometimes the act of composing cannot be separated from the control of the synthesis process (Cádiz, 2006b). As a consequence, the compositional process can be strongly shaped by the nature of the synthesis technique that is being used. Gerhard and Hepting (2004) propose to think of composition as an exploration of a multidimensional parameter

space, where a particular configuration of parameters can be represented as a point in that space. The parameters are initially de-contextualized, meaning that they only offer a possible set of future musical ideas, and compositions often means to map or re-map these parameters until targets or musical constraints are satisfied. This parameter-based approach to composition allows the composer to explore a high dimensional space of musical possibilities and essentially pick trajectories in that space that are aesthetically relevant. Dahlstedt (2001) and Gerhard and Hepting (2004) have proposed several options for this composition strategy.

As these parameter spaces become larger, more specialized tools are needed. In particular, supervised neural network methods have been often used to generate a model of the mapping from controller inputs to synthesis outputs, using training datasets consisting of examples of input/output pairs (Fiebrink et al., 2009). These kinds of networks can learn a continuous function mapping, no matter how many dimensions are involved. For these reasons, Fiebrink and Cook (2010) developed the Wekinator, a free and cross-platform open-source software application that supports interactive design and application of real-time supervised learning systems for many-to-many parametric gestural control of music.

Since version 2.0 of the FLCTK this kind of many-to-many control can also be achieved with fuzzy logic in real-time. As inputs, outputs and rules can be added on-the-fly, high dimensional parametric gestural control can be achieved with a regular fuzzy system. Please see the example video in the **Supplementary Material** for a better understanding of this on-the-fly mode. In the video, rules are added in real-time to map five inputs into four outputs controlling a sound synthesis algorithm. This is a straightforward way of learning directly from data. As can be observed in the video, desired inputs can be specified along with their desired corresponding outputs and these data pairs can be encoded on a specific rule. Instead of specifying these data points in real-time moving faders, it would be straightforward to add a functionality to the FLCTK to learn them directly from a file on disk.

TABLE 5 | Input and output variables for parametric control of granular synthesis example.

| Input variables | Output variables |
|-----------------|------------------|
| X | Pulse interval |
| Y | Buffer offset |
| | Duration |
| | Sample increment |
| | Gain multiplier |

Each one of these variables can belong to three fuzzy sets: LOW, MEDIUM, or HIGH. As the number of inputs is lower than the number of outputs, the input space is a latent space of the output space.

TABLE 6 | Fuzzy rules used in the granular synthesis example.

| Rule | Inputs | | Outputs | | | | |
|------|--------|--------|----------------|---------------|----------|-------------|------------|
| | X | Y | Pulse interval | Buffer offset | Duration | Sample inc. | Gain mult. |
| 1 | LOW | – | HIGH | MEDIUM | LOW | HIGH | MEDIUM |
| 2 | MEDIUM | – | HIGH | MEDIUM | – | LOW | LOW |
| 3 | HIGH | – | LOW | LOW | LOW | HIGH | LOW |
| 4 | – | LOW | MEDIUM | HIGH | HIGH | – | HIGH |
| 5 | – | MEDIUM | LOW | LOW | LOW | LOW | LOW |
| 6 | – | HIGH | LOW | LOW | HIGH | MEDIUM | HIGH |
| 7 | LOW | HIGH | LOW | MEDIUM | LOW | HIGH | MEDIUM |
| 8 | MEDIUM | HIGH | HIGH | MEDIUM | MEDIUM | LOW | HIGH |
| 9 | HIGH | HIGH | HIGH | HIGH | HIGH | HIGH | HIGH |

All rules have the same weight and are connected by AND operators. A “–” indicates that the value of the fuzzy variable can be anything.

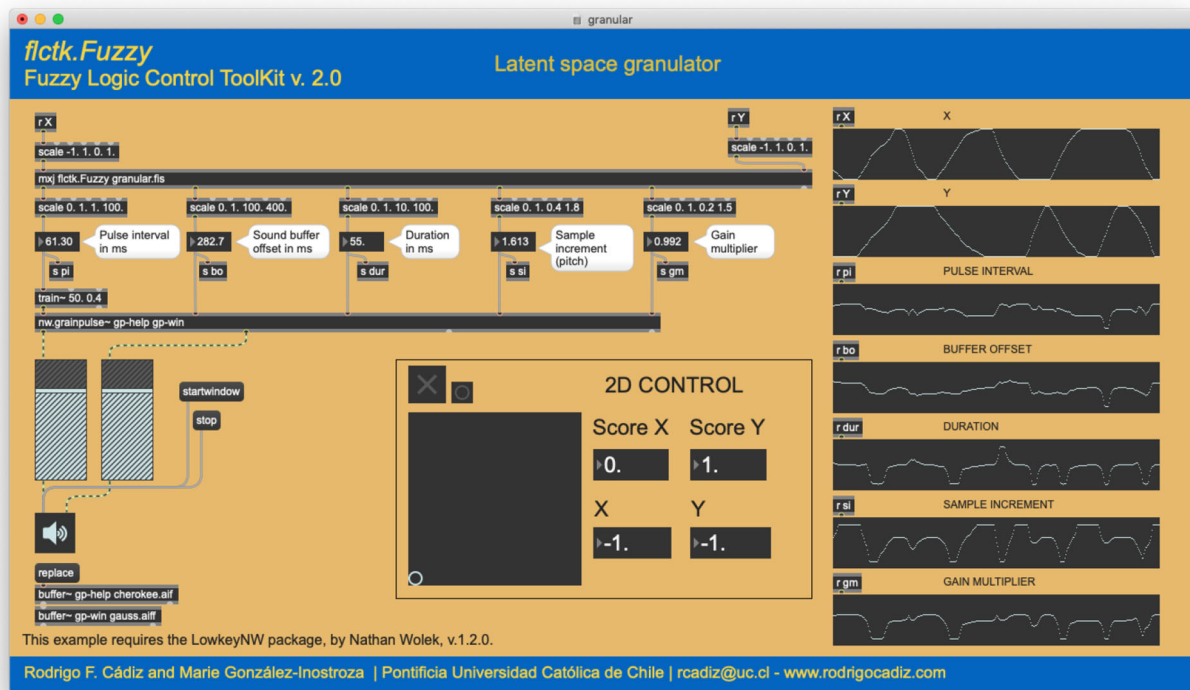


FIGURE 8 | Latent space granulator example. Inputs and outputs of the fuzzy system are displayed on the right. As the latent space is explored in both the X and Y directions, the output variables exhibit different non-linear behavior. This allows the parametric control of five synthesis parameters with only two abstract parameters.

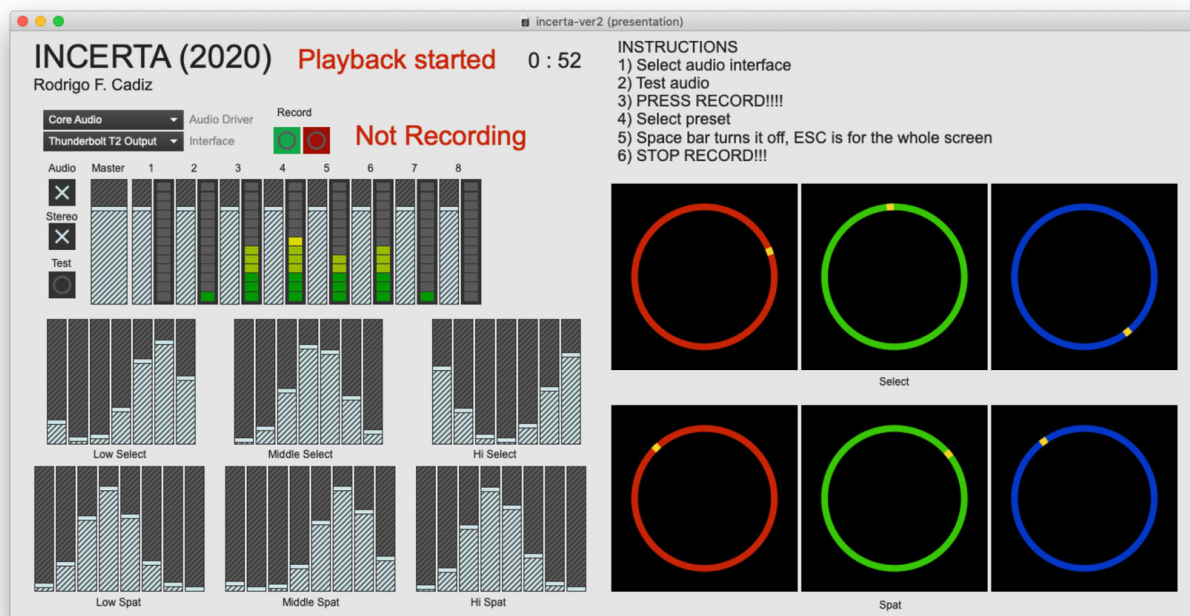


FIGURE 9 | Screenshot of the main interface of *Incerta* at time 0:52. The six Gaussian curves for the control of the sound material selection and spatialization can be seen at the bottom left. Circles on the right displays the rotation angle that is used to specify the mean value of each Gaussian curve.

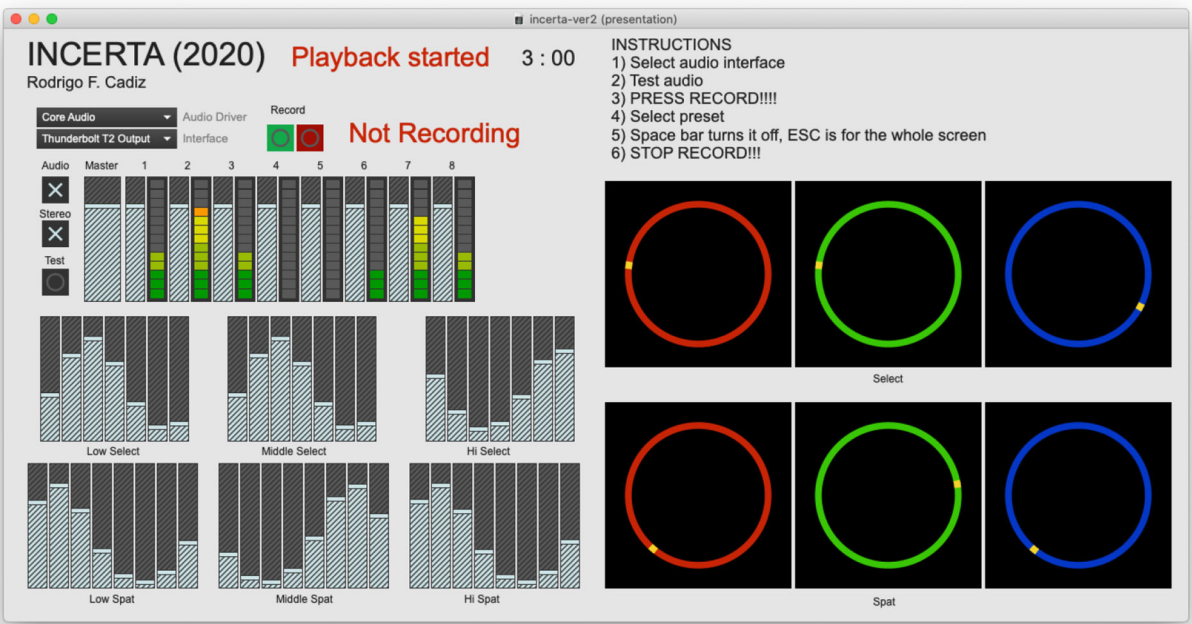


FIGURE 10 | Screenshot of the main interface of *Incerta* at time 3:00. The six Gaussian curves for the control of the sound material selection and spatialization can be seen at the bottom left. Circles on the right displays the rotation angle that is used to specify the mean value of each Gaussian curve.

TABLE 7 | Input and output variables for *Incerta*, an acousmatic composition for eight channels.

| Input variables | Output variables |
|------------------------------------|--|
| Low pitch selection angle (2) | Change in low pitch selection angle (1) |
| Middle pitch selection angle (3) | Change in middle pitch selection angle (2) |
| High pitch selection angle (4) | Change in high pitch selection angle (3) |
| Low spatial selection angle (5) | Change in low pitch spatial angle (4) |
| Middle spatial selection angle (6) | Change in middle pitch spatial angle (5) |
| High spatial selection angle (7) | Change in high pitch spatial angle (6) |
| Time (1) | Selection curve standard deviation (7) |
| | Spatial curve standard deviation (8) |

Each of the variable numbers have been assigned a number in parenthesis, as shown in **Table 8**. There are seven inputs and eight outputs in totals.

6. INCERTA: AN ACOUSMATIC MULTI-CHANNEL FUZZY COMPOSITION

Incerta is an 8-min acousmatic multi-channel composition created in MaxMSP with the FLCTK. *Incerta* is a latin word that could be translated into English as *vague*, in direct relation to the ability of fuzzy logic to handle uncertain data using vague concepts. The gist of the composition is very simple: twenty-one separate tracks of audio are presented in both temporal and spatial order according to a fuzzy logic inference engine.

The fuzzy system handles both the temporal and spatial presentation of the material across time. The twenty-one audio tracks are separated into three different groups, according to their pitch content, ranging from low-frequency textures to high pitches ones. Each group is presented at a given time on a specific spatial location.

Both the selection of individual sound files and spatial position in an eight-speaker system are determined by the selection of a specific Gaussian curve that specifies the amplitudes of a group of faders, as shown in **Figures 9, 10**. There are three curves for each pitch content (low, medium, and high) and three additional curves for the circular spatial position of each group. The mean of each curve is controlled by an angle variable in such a way that the faders overlap circularly. The Gaussian curves can also be made wider or thinner, and thus affecting a different number of faders, by controlling their standard deviation.

The fuzzy system takes the rotation angle of each of the six Gaussian curves as inputs and also a time variable that allows for time-based behavior as time progresses. In total, there are seven inputs to the system. The outputs of the system are the change that each angle should experience at the next time step and two variables that control the standard deviation of the selection and spatial curves. This is an example of a closed feedback system, where some of the outputs of the system affect the inputs at the next time step.

The fuzzy variables used in this composition are described in **Table 7** and can take the following values: Very Short (VSh), Short (Sh), Medium Short (MSh), Medium (M), Medium

TABLE 8 | Fuzzy rules for *Incerta*, an acousmatic composition for eight channels.

| Rule | Inputs | | | | | | | Outputs | | | | | | | |
|------|--------|-----|-----|-----|-----|-----|-----|---------|-----|-----|-----|-----|-----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | VSh | – | – | – | – | – | – | SI | SI | SI | SI | SI | SI | VL | VL |
| 2 | Sh | – | – | – | – | – | – | FC | FCC | FCC | FC | FCC | FC | L | L |
| 3 | MSh | – | – | – | – | – | – | FCC | SI | MC | SI | MC | SI | M | M |
| 4 | M | – | – | – | – | – | – | SI | SI | SI | SI | SI | SI | H | H |
| 5 | MLa | – | – | – | – | – | – | FC | FCC | FC | FCC | FC | FCC | VH | VH |
| 6 | La | – | – | – | – | – | – | FC | FCC | FC | FCC | FC | FCC | M | M |
| 7 | VLa | – | – | – | – | – | – | SI | SI | SI | SI | SI | SI | VL | VL |
| 8 | – | FCC | – | – | – | – | – | FC | – | – | – | FC | – | – | – |
| 9 | – | MCC | – | – | – | – | – | SI | – | – | – | – | – | – | – |
| 10 | – | SI | – | – | – | – | – | MC | – | – | – | – | – | – | – |
| 11 | – | MC | – | – | – | – | – | FC | – | – | – | – | – | – | – |
| 12 | – | FC | – | – | – | – | – | FCC | – | – | – | FC | – | – | – |
| 13 | – | – | FCC | – | – | – | – | – | FC | – | – | FCC | – | – | – |
| 14 | – | – | MCC | – | – | – | – | – | FC | – | – | – | – | – | – |
| 15 | – | – | SI | – | – | – | – | – | – | – | – | – | – | – | – |
| 16 | – | – | MC | – | – | – | – | – | FCC | – | – | – | – | – | – |
| 17 | – | – | FC | – | – | – | – | – | FCC | – | – | FCC | – | – | – |
| 18 | – | – | – | FCC | – | – | – | – | – | MC | – | FC | – | – | – |
| 19 | – | – | – | MCC | – | – | – | – | – | FC | – | – | – | – | – |
| 20 | – | – | – | SI | – | – | – | – | – | FCC | – | – | – | – | – |
| 21 | – | – | – | MC | – | – | – | – | – | MCC | – | – | – | – | – |
| 22 | – | – | – | FC | – | – | – | – | – | SI | – | – | – | – | – |
| 23 | – | – | – | – | FCC | – | – | – | FCC | – | MC | MCC | – | – | – |
| 24 | – | – | – | – | MCC | – | – | – | – | – | MC | – | – | – | – |
| 25 | – | – | – | – | SI | – | – | – | – | – | FCC | – | – | VH | VL |
| 26 | – | – | – | – | MC | – | – | – | – | – | MCC | – | – | – | – |
| 27 | – | – | – | – | FC | – | – | – | – | – | MCC | MC | – | – | – |
| 28 | – | – | – | – | – | FCC | – | – | – | – | – | FC | – | – | – |
| 29 | – | – | – | – | – | MCC | – | – | – | – | – | FC | – | – | – |
| 30 | – | – | – | – | – | SI | – | – | – | – | – | FCC | – | VL | VL |
| 31 | – | – | – | – | – | MC | – | – | – | – | – | FCC | – | – | – |
| 32 | – | – | – | – | – | FC | – | FCC | – | – | – | FCC | – | – | – |
| 33 | – | – | – | – | – | – | FCC | – | – | – | – | – | FC | – | – |
| 34 | – | – | – | – | – | – | MCC | – | – | – | – | – | MC | – | – |
| 35 | – | – | – | – | – | – | SI | – | – | – | – | – | FC | VH | VH |
| 36 | – | – | – | – | – | – | MC | – | – | – | – | – | FCC | – | – |
| 37 | – | – | – | – | – | – | FC | – | – | – | – | – | MCC | – | – |

All rules have the same weight and are connected by AND operators. A “–” indicates that the value of the fuzzy variable can be anything. Variable names are specified in **Table 7**. The fuzzy values that the variables can take are: Very Short (VSh), Short (Sh), Medium Short (MSh), Medium (M), Medium Large (MLa), Large (La), Very Large (VLa), Fast counter-clockwise (FCC), Medium counter-clockwise (MCC), Slow (SI), Medium clockwise (MC), Fast clockwise (FC), Very Low (VL), Low (L), High (H), and Very High (VH).

Large (MLa), Large (La), and Very Large (VLa) for time, Fast counter-clockwise (FCC), Medium counter-clockwise (MCC), Slow (SI), Medium clockwise (MC), Fast clockwise (FC) for rotation angles and Very Low (VL), Low (L), High (H), and Very High (VH) for standard deviations.

The rules for each system were created based on musical criteria, as shown in **Table 8**. In this approach to composition, most of the composer’s work deals with the design and tuning of the fuzzy inference rules. Once the rules are established, the piece unfolds in real-time as the composer specified. Rules were

designed in order. First, time dependence is established. Then, one rule for each possible fuzzy value of each one of the inputs is provided. This design methodology produces thirty-seven rules in total. Of course, these rules can be tweaked and fine-tuned to obtain specific desired behavior, but changing these rules too much would result perhaps in a different composition.

As time progresses the state of the whole fuzzy system changes, as it can be seen by comparing **Figure 9** with **Figure 10**, which corresponds to the same instance of the piece at different times, 0:51 and 3:00, respectively. The position of each of the rotating

circles is different, resulting in a different sonic output at those specific times. Another very interesting aspect of this approach is that the initial point of each input variable determines a different outcome. Even though there is some time dependence, the fact that there are closed loops in the system results in fuzzy outputs that are highly dependent on the initial conditions. As this composition is based on pre-generated sonic material, this complex behavior of the fuzzy system does not result in a totally different piece for different starting points, but there are indeed noticeable differences from one version to another. In this sense, this composition does not have a unique final format, but as many formats as there are initial conditions, which is infinite in theory.

The fuzzy system used in this piece can produce complex dynamic behavior, as it can be observed in the accompanying videos of three different performances or instances of the piece. The time evolution of each variable is distinct and the overall behavior of the piece is not the same. This is due to the thirty-seven inference rules encoded on the system. Videos of each of the *Incerta* performances can be found in the **Supplementary Material**.

7. DISCUSSION AND CONCLUSIONS

The provided examples show that the power of fuzzy systems lies in the parallel computation of very simple rules. A mathematical model is not needed to approximate any system, no matter how complex it could be. Fuzzy systems are, in general, much simpler to construct and use than other AI techniques, such as deep neural networks. They do not require a large amount of training or extremely large data sets. Rather, a few if-then like fuzzy inference rules, inspired by expert knowledge or common sense, are usually enough to develop interesting systems for musical creation. Fuzzy systems are very suitable tools for the control of high dimensional parameter spaces, as it could be observed from the algorithmic composition and parametric control examples, where five parameters could be successfully addressed with only two control dimensions. Also, because fuzzy systems can approximate any non-linear process, it is easy to create complex behavior, something highly valuable in creative endeavors.

Fuzzy logic is also a powerful way to implement non-linear mappings and intuitive control of non-intuitive synthesis parameters. However, one of the weaknesses of a fuzzy logic approach to parametric composition would be the time required to appropriately design adequate rules for the inference system. In engineering control applications, these rules are derived from expert knowledge or machine learning processes, where the rules are derived from trained data. In artistic applications, these rules constitute the heart of the underlying parameter mapping and it becomes really hard to select appropriate rules for a specific desired output when the parameter space is highly dimensional, which is often the case. Rule specification becomes an art form in itself, and it requires time and the development of expert knowledge specific to this kind of composition. In creative applications, when designing the fuzzy variables and rules, it is not necessary to worry about stability or controllable issues, the items on which control engineers spend most of their time. On the contrary, instability could be something very appealing to a composer.

The FLCTK constitutes a powerful and simple approach to the compositional control of computer music, as demonstrated by the examples described in this article. It has been successfully implemented in a variety of situations: algorithmic composition, particle-based synthesis, and granular synthesis control, and in the composition of a whole piece entitled *Incerta*. Overall, the FLCTK is a simple way of designing and implementing fuzzy logic inference systems inside MaxMSP. Its compatibility with MATLAB's fuzzy logic toolbox also allows this environment to be used in the design and test stages of the fuzzy models.

Finally, we would like to encourage the use of fuzzy systems as an alternative to the current trend of using deep learning and generative models for musical creation. Both approaches can complement each other. However, one big difference between these approaches is knowledge representation. In neural networks, it is sometimes very hard to understand what the knowledge captured by the network is. In fuzzy logic, it is very clear what is being learned and represented as all knowledge is encoded in the rules of the system, even if the rules were learned directly from data. This is a major difference between these approaches, and for some types of music, a fuzzy approach could be better suited than a purely data-based one.

DATA AVAILABILITY STATEMENT

All computer code and software tools generated for this study are included in the article/**Supplementary Material**.

AUTHOR CONTRIBUTIONS

The author confirms being the sole contributor of this work and has approved it for publication.

FUNDING

This research was funded by Fondecyt Grant No. #1161328, ANID, Government of Chile.

ACKNOWLEDGMENTS

The author would like to thank Marie González-Inostroza for her work in the Fuzzy Logic Control Toolkit v2.0.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/frai.2020.00059/full#supplementary-material>

Video S1 | Algorithmic composition example.

Video S2 | Sound synthesis example.

Video S3 | Parametric control example.

Video S4 | Many-to-many parameter control example.

Video S5 | *Incerta* performance 1.

Video S6 | *Incerta* performance 2.

Video S7 | *Incerta* performance 3.

REFERENCES

- Bandemer, H., and Gottwald, S. (1995). *Fuzzy Sets, Fuzzy Logic, Fuzzy Methods With Applications*. Chichester; New York, NY: J. Wiley.
- Belohlavek, R., and Klir, G. J., (Eds.). (2011). *Concepts and Fuzzy Logic*. Cambridge, MA: The MIT Press.
- Breining, C. (2001). A robust fuzzy logic-based step-gain control for adaptive filters in acoustic echo cancellation. *IEEE Trans. Speech Audio Process.* 9, 162–167. doi: 10.1109/89.902282
- Cádiz, R., and Kendall, G. (2005). “A particle-based fuzzy logic approach to sound synthesis,” in *Proceedings of the Conference on Interdisciplinary Musicology (CIM05)* (Montréal, QC:), 10–12.
- Cádiz, R. F. (2004). “A fuzzy logic model for compositional approaches to audiovisual media,” in *Proceedings of the International Computer Music Conference* (Miami, FL), 1–8.
- Cádiz, R. F. (2006a). A fuzzy-logic mapper for audiovisual media. *Comput. Music J.* 30, 67–82. doi: 10.1162/comj.2006.30.1.67
- Cádiz, R. F. (2006b). *Compositional control of computer music by fuzzy logic* (Ph.D. thesis), Northwestern University, Evanston, IL, United States.
- Cádiz, R. F., and Gonzalez-Inostroza, M. (2018). “Fuzzy logic control toolkit 2.0: composing and synthesis by fuzzyfication,” in *Proceedings of the International Conference on New Interfaces for Musical Expression* (Blackburn, VA), 398–402.
- Cádiz, R. F., and Kendall, G. S. (2006). “Fuzzy logic control tool kit: real-time fuzzy control for Max/MSP and Pd,” in *Proceedings of the International Computer Music Conference* (New Orleans, LA), 341–346.
- Civanlar, M. R., and Trussel, H. J. (1986). Digital signal restoration using fuzzy sets. *IEEE Trans. Acoust. Speech Signal Process.* 34, 919–936. doi: 10.1109/TASSP.1986.1164875
- Cox, E. (1994). *The Fuzzy Systems Handbook: A Practitioner's Guide to Building and Maintaining Fuzzy Systems*. Boston, MA: AP Professional.
- Dahlstedt, P. (2001). “Creating and exploring huge parameter spaces: interactive evolution as a tool for sound generation,” in *Proceedings of the International Computer Music Conference* (Havana), 1–8.
- Demico, R., and Klir, G. J. (2004). *Fuzzy Logic in Geology*. Amsterdam; Boston, MA: Elsevier Academic Press.
- Demichelis, P., Mori, R. D., Laface, P., and O' Kane, M. (1983). Computer recognition of plosive sounds using contextual information. *IEEE Trans. Acoust. Speech Signal Process.* 31, 359–377. doi: 10.1109/TASSP.1983.1164067
- Dimitrov, V., and Hodge, B. (2002). *Social Fuzziology. Study of Fuzziness of Social Complexity*. Heidelberg; New York, NY: Physica-Verlag.
- Dodge, C., and Jerse, T. A. (1997). *Computer Music: Synthesis, Composition, and Performance. 2nd Edn.* New York, NY; London: Schirmer Books; Prentice Hall International.
- Edwards, M. (2011). Algorithmic composition: computational thinking in music. *Commun. ACM* 54, 58–67. doi: 10.1145/1965724.1965742
- Elsea, P. (1995). *Fuzzy Logic and Musical Decisions*. Santa Cruz, CA: Technical report. Available online at: <http://artsites.ucsc.edu/ems/Music/research/FuzzyLogicTutor/FuzzyTut.html>
- Fernández, J. D., and Vico, F. (2013). AI methods in algorithmic composition: a comprehensive survey. *J. Artif. Intell. Res.* 48, 513–582. doi: 10.1613/jair.3908
- Fiebrink, R., and Cook, P. R. (2010). “The Wekinator: a system for real-time, interactive machine learning in music,” in *Proceedings of The Eleventh International Society for Music Information Retrieval Conference* (Utrecht).
- Fiebrink, R., Cook, P. R., and Trueman, D. (2009). “Play-along mapping of musical controllers,” in *Proceedings of the International Computer Music Conference* (Montréal, QC), 61–64.
- Finn, G. D. (1999). Learning fuzzy rules from data. *Neural Comput. Appl.* 8, 9–24. doi: 10.1007/s005210050003
- Friberg, A. (2005). “A fuzzy analyzer of emotional expression in music performance and body motion,” in *Proceedings of Music and Music Science* (Stockholm).
- Gerhard, D., and Hepting, D. (2004). “Cross-modal parametric composition,” in *Proceedings of the International Computer Music Conference* (Miami, FL), 1–8.
- Gonzalez-Inostroza, M., de La Cuadra, P., and Cádiz, R. F. (2015). “Fuzzy equalization of musical genres,” in *Proceedings of the International Computer Music Conference* (Denton, TX), 134–137.
- Guliyev, J., and Memmedova, K. (2019). Fuzzy logic-based compositional decision making in music. *Adv. Intell. Syst. Comput.* 896, 741–745. doi: 10.1007/978-3-030-04164-9_97
- Hasanzadeh, F., Annabestani, M., and Moghimi, S. (2019). Continuous emotion recognition during music listening using EEG signals: a fuzzy parallel cascades model. *arXiv* 1910.10489.
- Kasinathan, V., Mustapha, A., Firdaus Che Abdul Rani, M., Sau Tong, T., and Azlina Abd Rahman, N. (2019). Heartbeats: music recommendation system with fuzzy inference engine. *Indonesian J. Electric. Eng. Comput. Sci.* 16, 275–282. doi: 10.11591/ijeecs.v16.i1.pp275-282
- Kiseliova, T., Kiendl, H., and Rambintsoa, Y. (2005). “Fuzzy rules in computer-assisted music interpretation,” in *Proceedings of the International Computer Music Conference* (Barcelona), 1–4.
- Klir, G. J., and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ: Prentice Hall PTR.
- Knudsen, P., Miranda, L. D., and Saffiotti, A. (2019). Anticipation in collaborative music performance using fuzzy systems: a case study. *arXiv* 1906.02155.
- Kosko, B. (1992). *Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice Hall.
- Kosko, B. (1993). *Fuzzy Thinking. The New Science of Fuzzy Logic*. New York, NY: Hyperion.
- Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Trans. Comput.* 43, 1329–1333. doi: 10.1109/12.324566
- Kostek, B. (1999). *Soft Computing in Acoustics: Applications of Neural Networks, Fuzzy Logic and Rough Sets to Musical Acoustics*. Heidelberg: Physica-Verlag.
- Kuo, P.-H., Li, T.-H. S., Ho, Y.-F., and Lin, C.-J. (2015). Development of an automatic emotional music accompaniment system by fuzzy logic and adaptive partition evolutionary genetic algorithm. *IEEE Access* 3, 815–824. doi: 10.1109/ACCESS.2015.2443985
- Landy, L. (2001). From algorithmic jukeboxes to zero-time synthesis: a potential A-Z of music in tomorrow's world (a conference provocation). *Organ. Sound* 6, 91–96. doi: 10.1017/S1355771801002023
- Lee, M., and Wessel, D. (1995). “Soft computing for real-time control of musical processes,” in *IEEE International Conference on Systems Man and Cybernetics*, Vol.3 (Vancouver, BC), 2748–2753. doi: 10.1109/ICSMC.1995.538581
- Lee, M. A., and Wessel, D. (1993). “Real-time neuro-fuzzy systems for adaptive control of musical processes,” in *International Computer Music Conference* (Tokyo), 172–175. doi: 10.1117/12.165048
- Leon, T., and Liern, V. (2010). “Fuzzy logic helps to integrate music theory and practice,” in *2010 IEEE World Congress on Computational Intelligence, WCCI 2010* (Barcelona), 1683–1687. doi: 10.1109/FUZZY.2010.5584652
- León, T., and Liern, V. (2012). Mathematics and soft computing in music. *Soft Comput. Hum. Soc. Sci.* 273, 451–465. doi: 10.1007/978-3-642-24672-2_23
- Liu, M., Wan, C., and Wang, L. (2002). Content-based audio classification and retrieval using a fuzzy logic system: towards multimedia search engines. *Soft Comput.* 6, 357–364. doi: 10.1007/s00500-002-0189-3
- Liu, Z., and Huang, Q. (1998). “Classification of audio events in broadcast news,” in *1998 IEEE Second Workshop on Multimedia Signal Processing* (Redondo Beach, CA: IEEE), 364–369. doi: 10.1109/MMSP.1998.738963
- López-Ortega, O., and López-Popa, S. I. (2012). Fractals, fuzzy logic and expert systems to assist in the construction of musical pieces. *Expert Syst. Appl.* 39, 11911–11923. doi: 10.1016/j.eswa.2012.02.089
- Lucas, P., Astudillo, E., and Peláez, E. (2017). Human-machine musical composition in real-time based on emotions through a fuzzy logic approach. *Des. Comput. Intell.* 664, 143–159. doi: 10.1007/978-3-319-44735-3_8
- Lucas, P., and Peláez, E. (2019). “A granular synthesis strategy based on musical harmony theory through a fuzzy logic approach,” in *2019 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2019* (Guayaquil), 1–6. doi: 10.1109/LA-CCI47412.2019.9036760
- Malcangi, M. (2008). Fuzzy logic-based audio pattern recognition. *AIP Conf. Proc.* 1060, 225–228. doi: 10.1063/1.3037059
- Maristany, A., López, M. R., and Rivera, C. A. (2016). Soundscape quality analysis by fuzzy logic: a field study in Cordoba, Argentina. *Appl. Acoust.* 111, 106–115. doi: 10.1016/j.apacoust.2016.04.013
- McNeill, D., and Freiburger, P. (1993). *Fuzzy Logic*. New York, NY: Simon & Schuster.

- Meng, Z., Sakagami, K., Morimoto, M., Bi, G., and Alex, K. C. (2002). Extending the sound impulse response of rooms using extrapolation. *IEEE Trans. Speech Audio Process.* 10, 167–172. doi: 10.1109/TSA.2002.1001981
- Milicevic, M. (1999). “Designing an AI emotion-based adaptive fuzzy system for evaluation of the computer music,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4 (Tokyo: IEEE), 323–326. doi: 10.1109/ICSMC.1999.812421
- Miranda, E. R., and Junior, A. M. (2005). “Granular synthesis of sounds through Markov chains with fuzzy control,” in *Proceedings of the International Computer Music Conference* (Barcelona: International Computer Music Association), 1–4.
- Mitaim, S., and Kosko, B. (2001). The shape of fuzzy sets in adaptive function approximation. *IEEE Trans. Fuzzy Syst.* 9, 637–656. doi: 10.1109/91.940974
- Monti, G., and Sandler, M. (2002). “Automatic polyphonic piano note extraction using fuzzy logic in a blackboard system,” in *Proceedings of the International Conference on Digital Audio Effects* (Hamburg), 39–44.
- Nierhaus, G. (2009). *Algorithmic Composition: Paradigms of Automated Music Generation*. Vienna: Springer Science & Business Media.
- Orio, N., and Pirro, C. D. (1998). “Controlled refractions: a two-levels coding of musical gestures for interactive live performance,” in *Proceedings of the International Computer Music Conference* (Ann Arbor, MI), 1–4.
- Roads, C. (2004). *Microsound*. Cambridge, MA: MIT Press.
- Ross, T. J. (2010). *Fuzzy Logic With Engineering Applications*. 3rd Edn. Chichester: John Wiley.
- Roychowdhury, S., and Pedrycz, W. (2001). A survey of defuzzification strategies. *Int. J. Intell. Syst.* 16, 679–695. doi: 10.1002/int.1030
- Schatter, G. G., Züger, E., and Nitschke, C. (2005). “A synaesthetic approach for a synthesizer interface based on genetic algorithms and fuzzy sets,” in *Proceedings of the International Computer Music Conference* (Barcelona: International Computer Music Association), 1–4.
- Suiter, W. (2010a). The promise of fuzzy logic in generalised music composition. *IFIP Adv. Inform. Commun. Technol.* 333, 118–127. doi: 10.1007/978-3-642-15214-6_12
- Suiter, W. (2010b). “Toward algorithmic composition of expression in music using fuzzy logic,” in *Proceedings of the International Conference on New Interfaces for Musical Expression* (Sydney), 319–322.
- Usa, S., and Mochida, Y. (1998). A multi-modal conducting simulator. *J. Jpn. Soc. Fuzzy Theory Syst.* 10, 707–716. doi: 10.3156/jfuzzy.10.4_127
- Von Altrock, C. (1997). *Fuzzy Logic and Neurofuzzy Applications in Business and Finance*. Upper Saddle River, NJ; New York, NY: Prentice-Hall.
- Weyde, T., and Dalinghaus, K. (2001). “Recognition of musical rhythm patterns based on a neuro-fuzzy system,” in *Proceedings of the Conference on Artificial Neural Networks in Engineering* (St. Louis, IL), 1–6.
- Wolek, N. (2002). “A granular toolkit for cycling74’s Max/MSP,” in *SEAMUS 2002 National Conference* (Iowa City, IA: University of Iowa), 1–17.
- Wolek, N. (2005). *A simplified interface for granular processing based on perceptual research* (Ph.D. thesis), Northwestern University, Evanston, IL, United States.
- Wright, M., Freed, A., and Momeni, A. (2003). “Open sound control: state of the art 2003,” in *Proceedings of the International Conference on New Interfaces for Musical Expression* (Montréal, QC), 153–159.
- Yang, Y. H., Liu, C. C., and Chen, H. H. (2006). “Music emotion classification: a fuzzy approach,” in *Proceedings of the Annual ACM International Conference on Multimedia* (Santa Barbara, CA), 81–84. doi: 10.1145/1180639.1180665
- Yen, J. (1999). Fuzzy logic—a modern perspective. *IEEE Trans. Knowl. Data Eng.* 11, 153–165. doi: 10.1109/69.755624
- Yen, J., and Langari, R. (1999). *Fuzzy Logic: Intelligence, Control, and Information*. Upper Saddle River, NJ: Prentice Hall.
- Yilmaz, A. E., and Telatar, Z. (2009). “Potential applications of fuzzy logic in music,” in *IEEE International Conference on Fuzzy Systems* (Jeju Island), 670–675. doi: 10.1109/FUZZY.2009.5277385
- Yilmaz, A. E., and Telatar, Z. (2010). Note-against-note two-voice counterpoint by means of fuzzy logic. *Knowl. Based Syst.* 23, 256–266. doi: 10.1016/j.knosys.2010.01.007
- Ying, H. (1998). General SISO Takagi-Sugeno fuzzy systems with linear rule consequent are universal approximators. *IEEE Trans. Fuzzy Syst.* 6, 582–587. doi: 10.1109/91.728456
- Zadeh, L. A. (1965). Fuzzy sets. *Inform. Control* 8, 338–353. doi: 10.1016/S0019-9958(65)90241-X

Conflict of Interest: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Cádiz. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



On the use of AI for Generation of Functional Music to Improve Mental Health

Duncan Williams^{1,2,3*}, Victoria J. Hodge^{1,2} and Chia-Yu Wu⁴

¹Digital Creativity Labs, University of York, York, United Kingdom, ²Department of Computer Science, University of York, York, United Kingdom, ³School of Science, Engineering and Environment, University of Salford, York, United Kingdom, ⁴Department of Electronic Engineering, University of York, York, United Kingdom

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Gus Xia,
NYU Shanghai, China
Alexandra Bonnici,
University of Malta, Malta

*Correspondence:

Duncan Williams
d.a.h.williams@salford.ac.uk

Specialty section:

This article was submitted to Machine Learning and Artificial Intelligence, a section of the journal Frontiers in Artificial Intelligence

Received: 13 September 2019

Accepted: 19 October 2020

Published: 19 November 2020

Citation:

Williams D, Hodge VJ, Wu C-Y (2020)
On the use of AI for Generation of
Functional Music to Improve
Mental Health.
Front. Artif. Intell. 3:497864.
doi: 10.3389/frai.2020.497864

Increasingly music has been shown to have both physical and mental health benefits including improvements in cardiovascular health, a link to reduction of cases of dementia in elderly populations, and improvements in markers of general mental well-being such as stress reduction. Here, we describe short case studies addressing general mental well-being (anxiety, stress-reduction) through AI-driven music generation. Engaging in active listening and music-making activities (especially for at risk age groups) can be particularly beneficial, and the practice of music therapy has been shown to be helpful in a range of use cases across a wide age range. However, access to music-making can be prohibitive in terms of access to expertise, materials, and cost. Furthermore the use of existing music for functional outcomes (such as targeted improvement in physical and mental health markers suggested above) can be hindered by issues of repetition and subsequent over-familiarity with existing material. In this paper, we describe machine learning approaches which create functional music informed by biophysiological measurement across two case studies, with target emotional states at opposing ends of a Cartesian affective space (a dimensional emotion space with points ranging from descriptors from relaxation, to fear). Galvanic skin response is used as a marker of psychological arousal and as an estimate of emotional state to be used as a control signal in the training of the machine learning algorithm. This algorithm creates a non-linear time series of musical features for sound synthesis “on-the-fly”, using a perceptually informed musical feature similarity model. We find an interaction between familiarity and perceived emotional response. We also report on subsequent psychometric evaluation of the generated material, and consider how these - and similar techniques - might be useful for a range of functional music generation tasks, for example, in nonlinear sound-tracking such as that found in interactive media or video games.

Keywords: mental health, emotional states, feedback, biophysiological sensors, generative music, machine learning, artificial intelligence, algorithmic composition

INTRODUCTION

There is increasing evidence that mindfulness can form a positive contributor to mental health and general wellbeing (Baker and Bor, 2008; Economides et al., 2018). In this work we describe the design and evaluation of a system combining machine learning (ML) approaches with biophysiological metering and psychological evaluation of two descriptors which we consider to be at discrete ends of an affective space with positive mental health states at one side of the space (mindfulness, calmness, etc.), and negative mental states at the other side of the space (fear, anger, etc.) (Chambers et al., 2009).

The distinction between affective state, emotion, and mood, is complex, and is generally drawn between the duration of the response (Calvo et al., 2009). Various models of affective state exist, including models with dimensions for positivity and activation strength, such as the circumplex model of affect (Russell, 1980). This model places valence (as a measure of positivity) and arousal (as a measure of activation strength) on the horizontal and vertical axes respectively. Emotional descriptors (e.g., happy, sad, angry) can be mapped on to this space. Other models exist, for example, multidimensional models which also include, for example, dimensions for dominance. This type of model might be useful when delineating between very intense and very negative emotional descriptors, such as the difference between anger and fear—both intense, and negative, but one being a more dominant response and the other more passive. Often, individual emotional descriptors can be plotted across these types of spaces (Williams et al., 2014). In the case of this work, we consider a general model with two specific descriptors as approximately at either end of a scale—mindful, and afraid. However, the descriptors themselves are open to debate and could certainly form the subject of further work. We intend to explore the use of AI to generate music intended to elicit differing emotional states in an abstract emotional space and to examine biophysiological markers in a synchronous manner.

Existing work has shown that there are responses to music in both the central and peripheral nervous system (in other words, both physiological, and neurological responses) (Aldridge, 2005; Calvo et al., 2009). When listening to enjoyable music, the listeners pupils may dilate, or they might experience a change in heart rate, blood pressure, and skin conductivity (Blood et al., 1999; Daly et al., 2015). Measurement of galvanic skin response (GSR) has been shown to be a robust metric for analysis of emotional responses to music (Shrift, 1954; Vanderark and Ely, 1993; Daly et al., 2015).

Thus, there is a potential crossover between mental state, physiological reaction, and musical stimulation. Chambers (Chambers et al., 2009) showed that states of mindfulness have correlations in GSR (otherwise known as electrodermal activity, or skin conductivity), heart rate variability, and the ratio of alpha and beta waves in electroencephalographic measurement. The electroencephalograph (EEG) is a technique for metering electrical activity from the scalp used to infer patterns of brain activity. Bondolfi (Bondolfi, 2013) and Economides et al. (Economides et al., 2018) proposed that proactive training and entrainment of mental states might

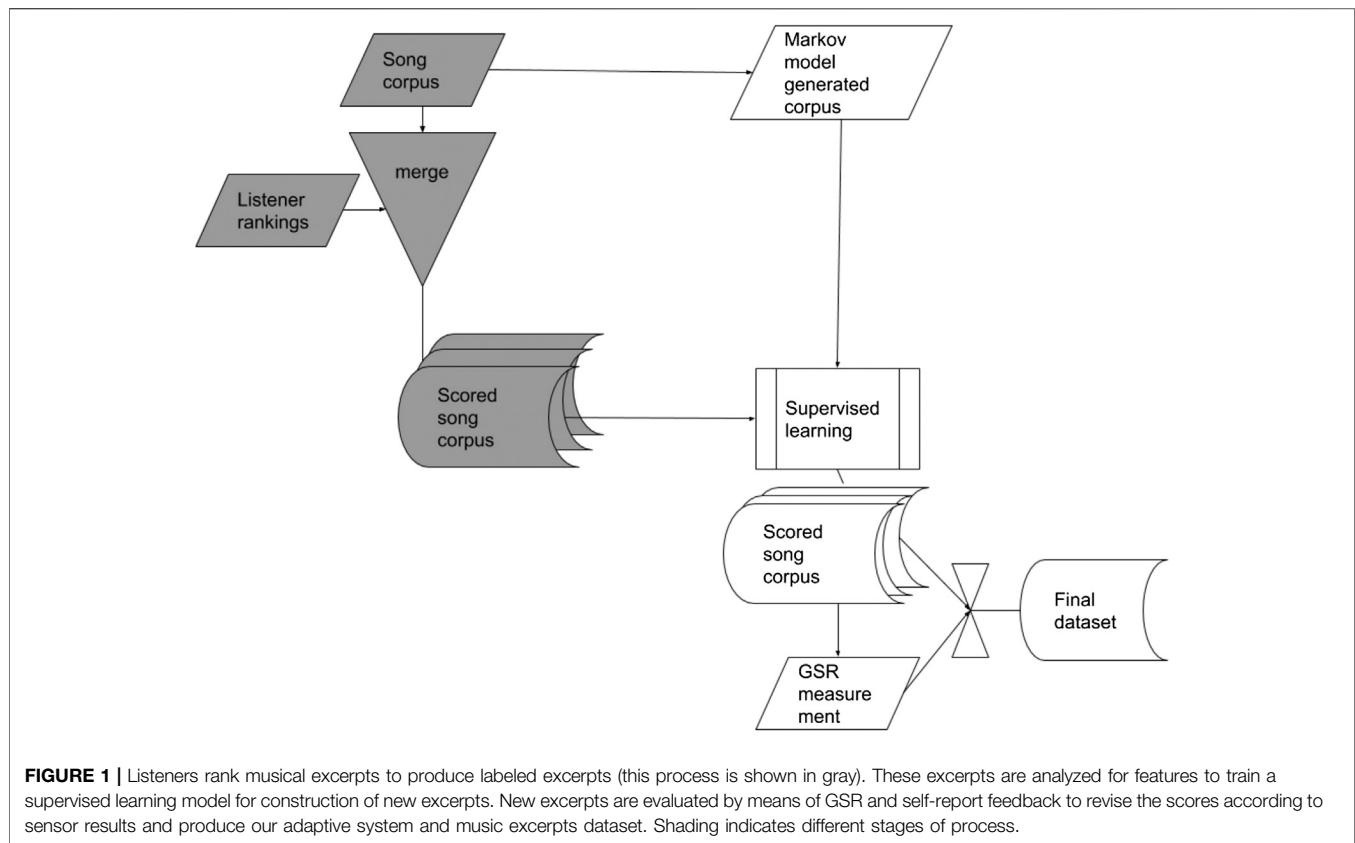
thus contribute to therapeutic treatment and physiological improvement.

We aim to harness these findings to create a machine learning based music-training system to encourage a change in affective state as measured through biophysiological correlations. For example, mood-based regulation (becoming less afraid or anxious) might be a useful mental health target for the user. Beyond mental health this type of system could have applications in the creative industries, for example, in film, television, or video games (Knox et al., 2008; Williams et al., 2015a), in which case, the viewer or player might be subjected to targeted mood disruption (i.e., rather than being calmed by the musical stimulus, the listener might preferably be deliberately excited, or even scared in the case of some types of gameplay).

In this paper we draw on previous experimental work documented in (Williams et al., 2019a; Williams et al., 2019b), and more widely, machine learning, a field of computer science covering systems that learn “*when they change their behavior in a way that makes them perform better in the future*” (Witten et al., 2016). These systems learn from data without being specifically programmed. Kim et al. (Kim et al., 2010) and Laurier and Herrera (Laurier and Herrera, 2012) give a literature overview of detecting emotion in music and focus on the music representations. Laurier and Herrera also provide an analysis of the machine learning algorithms used by systems in their survey. Classification algorithms used in the literature include C4.5, Gaussian mixture models, k-nearest neighbor, random forest, support vector machines, (Kim et al., 2010; Laurier and Herrera, 2012; Mostafavi et al., 2013). Regression techniques include Gaussian mixture model regression, multiple linear regression, partial least-squares regression and support vector regression.

ML has been used to retrieve music by mood and ML analyses found the personalized approach more consistent than a general approach (Mostafavi et al., 2013). An example is supervised learning. In supervised learning, the algorithm learns from a set of labeled inputs. It then generates a model associating the inputs with their respective labels or scores, and then classifies (or predicts) the likely label for previously unseen examples using the learned model. We use supervised learning to label newly generated material with potential emotional descriptors in the work documented in this paper.

Real-world testing of systems using bio-signal mappings in music generation contexts has become an emerging field of research, partly due to recent advances in portability, wearability, and affordability of biosensors. For example, Huang and Cai (Huang and Cai, 2017) generate simple music for specific emotional states using Markov chains. The Markov chains are used to generate music while the user wears a heart-rate sensor to monitor their bio-physiological response to the created music. The system was able to generate emotionally responsive music in a limited trial considering basic emotional descriptors. We have developed another such system, which assumes lower skin conduction variability as a correlate for positive affective state. It attempts to generate emotionally congruent music as a training tool to promote positive affective states in the context of mindfulness. In the future,



this system could also work in reverse by using skin conductance variability as a control signal to inform musical feature mapping for non-linear music generation.

While the physical and mental health benefits of music use have increasingly been reported upon (including improvements in cardiovascular health (Szmedra and Bacharach, 1998)), reduction of dementia in elderly populations (Vink et al., 2003), stress reduction (Knight and Rickard, 2001) and so on), the use of existing music to target such outcomes can be problematic due to the influence of familiarity, or repetition of stimulus materials (Kim, 2011; Ladinig and Schellenberg, 2012). Thus a major focus of this work is to evaluate a system for the automatic generation of new musical materials with functional aims (improvement of listener affective state on a case-by-case basis as determined by self-report or biophysiological correlate). We therefore aim to:

- (1) Measure musical features according to a similarity model from the human-labeled dataset and use these to inform Markov-model generation of new music to be evaluated by a supervised learning algorithm
- (2) Evaluate the success of the supervised learning algorithm using self-report and biophysiological measurement of GSR

We hypothesize that music generated by the automated algorithm may be able to influence self-reported emotional

state and GSR, when compared to music which listeners may already be familiar with from a corpus of popular film music.

MATERIALS AND METHODS

GSR is used as a marker of psychological arousal and as an estimate of emotional state to be used as a control signal in the training of the ML algorithm. This algorithm creates a non-linear time series of musical features for sound synthesis “on-the-fly,” using a perceptually informed musical feature similarity model.

We use the system described in (Williams et al., 2017) to create functional music informed by biophysiological measurement across two case studies, with target emotional states at opposing ends of a Cartesian affective space (a dimensional emotion space with points ranging from descriptors from relaxation, to fear).

The system detects the user’s current emotional level and the ML algorithm picks musical pieces to influence their future emotional level to achieve their desired mood. This whole process requires musical pieces that have an associated emotional label (score) to allow the selection of appropriate pieces. We use two tasks to achieve this, illustrated in **Figure 1**. Firstly we analyze a corpus for musical features using classification and regression. We use a multi-feature music representation combining analysis of symbolic musical feature data from a MIDI file, which represents the structure of

the melody, chords, rhythm and other musical properties concerning timing, dynamics, and pitch, with Mel-Frequency Cepstral Coefficients features (Logan, 2000) obtained from the entire piece to represent the timbral quality of the instrumentation. This dual representation is more flexible and richer than either MIDI or signal-based audio content features alone. We only use numerical data features to describe each piece and perform feature selection to identify the most significant set of features as described in (Hodge et al., 2016). Using this reduced set of significant features, the ML model can predict the likely emotional state score that a human listener would ascribe to newly input music pieces by determining the similarity between pieces using their respective sets of features.

We train the supervised learning algorithm to expand on a human-labeled corpus, which was labeled by means of a survey of 53 participants using a Qualtrics on-line survey (www.qualtrics.com). Each participant evaluated four musical excerpts, two calm or positive (N1 and N2) and two anxious or negative excerpts (S1 and S2), in a bipolar ranking across six pairs choosing the positive in each pair {N1vsS1, S2vsN2, S1vsN2, N1vsS2, S1vsS2 and N1vsN2}. The survey presented an initial question to allow the user to familiarize themselves with the format and then presented the six questions. The Qualtrics questionnaire allowed us to specify that each track played in full to each participant to ensure that the participant adapted fully to the track. We randomized the order of presentation of the questions (pairs of tracks) to each of the participants to reduce contextual effects. Participants were not required to answer every question in order to complete the evaluation. The algorithmic composition system uses hidden Markov models (HMM) to create new music to provide sufficient quantities of labeled pieces for the system to operate. We use a transformative algorithm based on a second order Markov-model with a musical feature matrix. New material is formed of permutations of the HMM with deliberate feature constraints following the procedure described in (Williams et al., 2015b; Williams et al., 2017). This allows discrete control over five parameters in a 2-dimensional model.

Human experiments are only feasible on a small set of music pieces as n pieces of music require $n!$ comparisons and enough human survey participants to provide enough responses for each of the $n!$ comparisons. Using human participants to generate a sufficiently large database of labeled pieces for this work would be very time consuming and complex. From our Qualtrics analyses we were able to train our ML model using supervised learning to map the musical sequences to scores where the sequences are represented by features as described previously. Our generative system can be used to create new musical sequences according to the likelihood of a particular affective state occurring after the current and preceding states measured in the listener and these can be scored by the trained ML model.

We then analyze the listener's GSR to select music which exercises the most influence of the listener's affective state. We incorporate a feedback loop to adapt the corpus scores according to the user's affective response, selecting musically consistent pieces and removing pieces that do not influence the user's emotional level (in essence a fitness function). We then compare the listener's GSR signal, the emotional tag they

describe after listening and the calmness level of the piece the participant is listening to. To analyze GSR, we used the Shimmer3 wireless GSR + Unit¹ which has been validated for use in biomedical-oriented research applications. This device needs to be calibrated on each use to establish a baseline skin conductance signal, which varies due to many factors including skin dryness, nervousness (due to unfamiliarity with the experimental procedure) and ambient temperature. The captured reading for each user under analysis is their skin conductance response while undertaking the listening exercise, minus their individual skin conductance response baseline. After listening to each piece, the users completed a questionnaire describing the emotion they felt while listening which we compared to the GSR data.

RESULTS

Responses to the musical stimuli suggest that listeners found it relatively easy to discriminate the affective states between stimuli, which were rendered using different synthesized timbres. Generally speaking, shorter durations and larger pitch ranges were considered lower in positivity (for example, "more anxious") than longer durations with a more restricted pitch range, regardless of the musical timbres. 58.1% of participants thought S2 was more negative than S1 while 54.6% felt N1 was more negative than N2.

For S1, 94.5 and 93.2% of participants rated it more negatively than N1 and N2 respectively. For S2, 88.1 and 89.1% of participants rated it more negatively than N1 and N2 respectively. Yet, 58.1% of the participants rated S2 more negatively than S1 despite S2 having created more positive report than S1 when compared to the positive stimuli. Similarly, for N1, 4.6 and 10.9% rated it more negatively than S1 and S2, respectively, while for N2, 6.8 and 11.9% rated it more negatively than S1 and S2, respectively. This presents a similar contradiction as for the negative stimuli as N1 has lowest reported positivity yet was rated more negatively than N2 by 54.6% of participants.

From these comparisons, we were able to attain sufficient data that we can calculate a ranked order (score) for the pieces from these pairwise comparisons (Wauthier et al., 2013). In order for the biofeedback based evaluation to be feasible, we then use the supervised learning generated corpus to provide a large enough quantity of stimuli.

Our analyses revealed that there is a direct correlation between the reported negativity of a musical piece, the user's GSR readings and the emotions they describe feeling in a questionnaire survey conducted after listening. Users display elevated GSR for negative pieces which they also labeled congruently in the questionnaire and lower GSR and appropriate labels for calmer pieces. We also find an interaction between familiarity of existing material in the corpus, and the perceived emotional response.

¹<http://www.shimmersensing.com/products/shimmer3-wireless-GSRsensor>

DISCUSSION

Our experiments highlighted that familiarity influences individual affective responses both in self-report and in GSR. For this reason, we have attempted to focus on generating novel music to create functional music which responds to a listener's biophysiological state rather than invoked or evoking memories (and removing some need to consider the influence of familiarity might affect listener responses).

Overall, we saw an increased GSR in each music excerpt, regardless of whether the excerpts were generated by the HMM model or not. We conclude that GSR is a suitable detection tool to evaluate emotional responses. DES-based self-report was used to allow listeners to report on different categories of emotions (Lane et al., 1990). However, the two measurements do not have consistent results when considered in response to music that listeners described themselves as being familiar with (samples of famous film music). Nevertheless, the emotional responses to generated music excerpt *g1* showed consistent results with both self-reporting and GSR. Thus we consider there may be an interaction between music and familiarity (perceived emotions). In self-reports, familiarity has insignificant effects. Conversely, in GSR data, there are differences in the simple effect of music between unfamiliar and familiar tracks. Familiar movie soundtracks also have higher GSR amplitude than unfamiliar ones but lower negative self-reports.

Hence, to induce calm states of mind reliably we believe further work should focus on unfamiliar music composed using artificial intelligence based approaches. Our main aim for this work is to develop a music generator for music therapy use that produces music which induces specific emotions in the listener but the approach described here might also be suitable in the design of a more generic music generator capable of inducing specific emotions in the audience, specifically when functional music with non-linear duration would be useful (e.g., videogame sound-tracking and the creative industries more broadly construed).

CONCLUSIONS AND FURTHER WORK

This work suggests that generative music technology has the potential to produce infinite soundtracks in sympathy with a listener's bio-signals, and in a biofeedback loop. There are promising applications for linking music with emotions, especially in the creative industries art and therapy, and particularly for relaxation. Enhancement of well-being using music and the emotions music induces is becoming an emerging topic for further work. We have applied a system for musical feature analysis from MIDI features and Mel-Frequency Cepstral Coefficients features (Logan, 2000) to train a supervised learning algorithm with listener responses to a corpus of training material. We use this algorithm to influence the generation of a larger corpus by means of a Hidden Markov Model algorithmic composition engine, and then analyzed the complete corpus by testing listener GSR and self-report in a DES evaluation. GSR is used as a marker of

psychological arousal and as an estimate of emotional state to be used as a control signal in the training of the ML algorithm. This algorithm creates a non-linear time series of musical features for sound synthesis "on-the-fly", using a perceptually informed musical feature similarity model. These small case studies, with target emotional states at opposing ends of a Cartesian affective space (a dimensional emotion space with points ranging from descriptors from positive descriptor states such as calmness, to negative descriptors such as fear), show us an interaction between familiarity and perceived emotional response. We believe further work involves three major challenges.

- (1) The extraction of meaningful control information from signals emanating from the body.
- (2) Design of generative and performative music technology in order to respond to such information.
- (3) Consideration of the ways in which such technology can be best deployed depending on the intended end-use; for example, in therapeutic contexts.

There is a tendency in human-computer interaction work for music generation to prioritize the technical implementation by focusing on increased speed or accuracy of a system, rather than the specific needs of the application. In a music therapy context, for example, one advantage of a functional music system is that it might be used by a patient with no musical ability and thereby potentially increases their own ability to express emotional states and have access to the pleasure of performing music with other people. Thus the use of biophysiological sensors is critical in the development of suitable systems for audio generation in the context of mindfulness or relaxation where improved affective state as part of mental health is an intended outcome. These are not trivial considerations in terms of application design, and subsequent evaluation. Methodologies for evaluating the success or failure of such systems remain a significant challenge for further work.

DATA AVAILABILITY STATEMENT

The datasets analyzed in this article are not publicly available. Requests to access the datasets should be directed to d.a.h.williams@salford.ac.uk.

ETHICS STATEMENT

The experiment was conducted with ethical approval from the University of York, Dept of Electronic Engineering review board. The patients/participants provided their written informed consent to participate in this study.

AUTHOR CONTRIBUTIONS

DW and VH contributed conception and design of the study; DW composed the musical sequence database; CW led the human

analyses under guidance from DW and VH, VH developed the Qualtrics survey, CW and VH performed the statistical analyses; DW wrote the first draft of the article; VH and CW wrote sections of the article. All authors contributed to article revision, read and approved the submitted version.

REFERENCES

- Aldridge, D. (2005). *Music therapy and neurological rehabilitation: performing health*. London UK: Jessica Kingsley Publishers, 272.
- Baker, F., and Bor, W. (2008). Can music preference indicate mental health status in young people? *Australas. Psychiatr.* 16 (4), 284–288. doi:10.1080/10398560701879589.
- Blood, A. J., Zatorre, R. J., Bermudez, P., and Evans, A. C. (1999). Emotional responses to pleasant and unpleasant music correlate with activity in paralimbic brain regions. *Nat. Neurosci.* 2 (4), 382–387. doi:10.1038/7299.
- Bondolfi, G. (2013). [Depression: the mindfulness method, a new approach to relapse]. *Rev. Med. Suisse.* 9 (369), 32–39. doi:10.1176/appi.focus.20170039.
- Calvo, R. A., Brown, I., and Steve, S. (2009). “Effect of experimental factors on the recognition of affective mental states through physiological measures,” in Australasian joint Conference on artificial intelligence (Berlin, Heidelberg: Springer), 62–70.
- Chambers, R., Gullone, E., and Allen, N. B. (2009). Mindful emotion regulation: an integrative review. *Clin. Psychol. Rev.* 29 (6), 560–572. doi:10.1016/j.cpr.2009.06.005.
- Daly, I., Malik, A., Weaver, J., Hwang, F., Nasuto, S. J., Williams, D., et al. (2015). “Toward human-computer music interaction: evaluation of an affectively-driven music generator via galvanic skin response measures,” in 7th computer science and electronic engineering conference, September 24–25, 2015. Colchester, UK: IEEE, 87–92. doi:10.1109/CEEC.2015.7332705.
- Economides, M., Martman, J., Bell, M. J., and Sanderson, B. (2018). Improvements in stress, affect, and irritability following brief use of a mindfulness-based smartphone app: a randomized controlled trial. *Mindfulness* 9 (5), 1584–1593. doi:10.1007/s12671-018-0905-4.
- Hodge, V. J., O’Keefe, S., and Austin, J. (2016). Hadoop neural network for parallel and distributed feature selection. *Neural Network.* 78, 24–35. doi:10.1016/j.neunet.2015.08.011.
- Huang, C-F., and Cai, Y. (2017). “Automated music composition using heart rate emotion data,” in International conference on intelligent information hiding and multimedia signal processing (Cham, Switzerland: Springer), 115–120.
- Kim, J. (2011). Affective states, familiarity and music selection: power of familiarity. *Int. J. Art Tech.* 4 (1), 74–89. doi:10.1504/ijart.2011.037771.
- Kim, Y. E., Schmidt, E. M., Migneco, R., Morton, B. G., Richardson, P., et al. (2010). Music emotion recognition: a state of the art review. *Int. J. Arts Technol.* 4, 74–89.
- Knight, W. E. J., and Rickard, N. S. (2001). Relaxing music prevents stress-induced increases in subjective anxiety, systolic blood pressure, and heart rate in healthy males and females. *J. Music Ther.* 38 (4), 254–272. doi:10.1093/jmt/38.4.254.
- Knox, D., Cassidy, G., Scott, B., and Macdonald, R. A. R. (2008). “Music emotion classification by audio signal analysis: analysis of self-selected music during game play,” in Proceedings of the 10th international conference on music perception and cognition, Sapporo, Japan, August 25–29, 2008, 581–587.
- Ladinig, O., and Schellenberg, E. G. (2012). Liking unfamiliar music: effects of felt emotion and individual differences. *Psychol. Aesthet. Creat. Arts.* 6 (2), 146–154. doi:10.1037/a0024671.
- Lane, R. D., Quinlan, D. M., Schwartz, G. E., Walker, P. A., and Zeitlin, S. B. (1990). The levels of emotional awareness scale: a cognitive-developmental measure of emotion. *J. Pers. Assess.* 55 (1–2), 124–134. doi:10.1207/s15327752jpa5501&2_12.
- Laurier, C., and Herrera, P. (2012). “Automatic detection of emotion in music: interaction with emotionally sensitive machines,” in *Machine learning: concepts, methodologies, tools and applications* Jyväskylä, Finland, August 12–16, 2009 (IGI Global), 1330–1354.
- Logan, B. (2000). Mel frequency cepstral coefficients for music modeling. *ISMIR*, 270:1–11.
- Mostafavi, A. C., Ras, Z. W., and Wiczorkowska, A. (2013). “Developing personalized classifiers for retrieving music by mood,” in Proc. Int. workshop on new frontiers in mining complex patterns, Prague, Czech Republic, September 27, 2013 (Cham, Switzerland: Springer International Publishing).
- Russell, J. A. (1980). A circumplex model of affect. *J. Pers. Soc. Psychol.* 39 (6), 1161–1178. doi:10.1037/h0077714.
- Shrift, D. C. (1954). *The galvanic skin response to two contrasting types of music*. Lawrence, KS: University of Kansas.
- Szmedra, L., and Bacharach, D. (1998). Effect of music on perceived exertion, plasma lactate, norepinephrine and cardiovascular hemodynamics during treadmill running. *Int. J. Sports Med.* 19 (1), 32–37. doi:10.1055/s-2007-971876.
- Vanderark, S. D., and Ely, D. (1993). Cortisol, biochemical, and galvanic skin responses to music stimuli of different preference values by college students in biology and music. *Percept. Mot. Skills.* 77 (1), 227–234. doi:10.2466/pms.1993.77.1.227.
- Vink, A. C., Bruinsma, M. S., and Scholten, R. J. P. M. (2003). Music therapy for people with dementia. *Cochrane Database Syst. Rev.* 4, doi:10.1002/14651858.CD003477.
- Wauthier, F., Jordan, M., and Jojic, N. (2013). “Efficient ranking from pairwise comparisons,” in International conference on machine learning (Thousand Oaks, CA: SAGE), 109–117.
- Williams, D., Kirke, A., Miranda, E. R., Roesch, E., Daly, I., and Nasuto, S. (2014). Investigating affect in algorithmic composition systems. *Psychol. Music.* 43 (6), 831–854. doi:10.1177/0305735614543282.
- Williams, D., Kirke, A., Eaton, J., Miranda, E., Daly, I., Hollowell, J., et al. (2015a). “Dynamic game soundtrack generation in response to a continuously varying emotional trajectory,” in Audio engineering society conference: 56th international conference: Audio for games, February 11–15, 2015. New York, NY: Audio Engineering Society.
- Williams, D., Kirke, A., Miranda, E., Daly, I., Hollowell, J., Weaver, J., et al. (2015b). Investigating perceived emotional correlates of rhythmic density in algorithmic music composition. *Trans. Appl. Percept.* 12 (3), 8. doi:10.1145/2749466.
- Williams, D., Kirke, A., Miranda, E., Daly, I., Hwang, F., Weaver, J., et al. (2017). Affective calibration of musical feature sets in an emotionally intelligent music composition system. *Trans. Appl. Percept.* 14 (3), 1–13. doi:10.1145/3059005.
- Williams, D., Hodge, V., Gega, L., Murphy, D., Cowling, P., and Anders, D. (2019a). “AI and automatic music generation for mindfulness,” in Audio engineering society conference: 2019 AES international conference on immersive and interactive audio, York, UK, March 27–29, 2019 (New York, NY: AES). Available at: <http://www.aes.org/e-lib/browse.cfm?elib=20439>.
- Williams, D., Wu, C-Y., Hodge, V., Murphy, D., and Cowling, P. (2019b). A psychometric evaluation of emotional responses to horror music. in Audio engineering society convention 146, Dublin, Ireland, March 21–23, 2019 (New York, NY: AES). Available at: <http://www.aes.org/e-lib/browse.cfm?elib=20270>.
- Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data mining: practical machine learning tools and techniques*. Burlington, MA: Morgan Kaufmann.

ACKNOWLEDGMENTS

This work was supported by the Digital Creativity Labs (www.digitalcreativity.ac.uk), jointly funded by EPSRC/AHRC/InnovateUK under grant no EP/M023265/1.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Williams, Hodge and Wu. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Evolving Musical Sight Reading Exercises Using Expert Models

Charlotte Pierce^{1*}, Tim Hendtlass², Anthony Bartel³ and Clinton J. Woodward²

¹Melbourne School of Engineering, The University of Melbourne, Parkville, VI, Australia, ²Faculty of Science, Engineering and Technology, Swinburne University of Technology, Hawthorn, VI, Australia, ³Faculty of Health, Arts and Design, Swinburne University of Technology, Hawthorn, VI, Australia

Sight reading skills are widely considered to be crucial for all musicians. However, given that sight reading involves playing sheet music without having seen it before, once an exercise has been completed by a student it can no longer be used as a sight reading exercise for them. In this paper we present a novel evolutionary algorithm for generating musical sight reading exercises in the Western art music tradition. Using models based on expert examples, the algorithm generates material suitable for practice which is both technically appropriate and aesthetically pleasing with respect to an instrument and difficulty level. This overcomes the resource constraint in using traditional practice exercises, which are exhausted quickly by students and teachers due to their limited quantity.

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Elad Liebman,
Other, Austin, United States
Steven Kemper,
The State University of New Jersey,
United States

*Correspondence:

Charlotte Pierce
charlotte.pierce@unimelb.edu.au

Specialty section:

This article was submitted to
Machine Learning and
Artificial Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 12 September 2019

Accepted: 17 November 2020

Published: 27 January 2021

Citation:

Pierce C, Hendtlass T,
Bartel A and Woodward CJ (2021)
Evolving Musical Sight Reading
Exercises Using Expert Models.
Front. Artif. Intell. 3:497530.
doi: 10.3389/frai.2020.497530

Keywords: expert models, musical sight reading, melody representation, music education, evolutionary algorithms

1 INTRODUCTION

Sight reading is widely believed to be a basic skill every musician should obtain (Spillman, 1990; Crozier, 2000; Lehmann and McArthur, 2002; McPherson and Gabrielsson, 2002; Galyen, 2005; Kopiez and In Lee, 2008), and is required at most levels of formal musical achievement in many countries (Ji, 2017; Australian Music Examinations Board, 2018a; The Associated Board of the Royal Schools of Music, 2018). It enables musicians to learn new music quickly, to rapidly expose themselves to a variety of repertoire and musical styles, and to become independent musical learners (Gregory, 1972). For students specifically, good sight reading skills allow them to dedicate more lesson time to musical interpretation rather than learning notes. For music teachers, sight reading is essential for demonstrating examples to their students.

As with most skills, practice is key to improving musical sight reading ability. This is shown by Kopiez and In Lee (2008), who found that there is a positive correlation between the time a person has spent practicing sight reading and their level of sight reading skill. As sight reading is the ability to perform a piece or phrase of music without having seen it before, as soon as a single exercise has been completed once by an individual it is no longer a sight reading exercise for them (Schulz, 2016). Currently, practice material for students preparing for formal music exams in the Western tradition is written by experts and disseminated to students through online stores and physical books. This is an ineffective approach. Access to expertize is limited, and practice material is consumed much faster than it is created. This means that students often exhaust the available resources before achieving competency. Given that both the quantity and quality of practice is key to gaining competence in sight reading (Kornicke, 1992; Banton, 1995; Galyen, 2005; Kopiez and In Lee, 2008; Tsangari, 2010), this resource constraint is a large barrier for musicians attempting to develop the skill.

To overcome this resource constraint, in this paper we present a novel evolutionary algorithm (EA) for generating monophonic sight reading exercises in the Western art music tradition. The goal

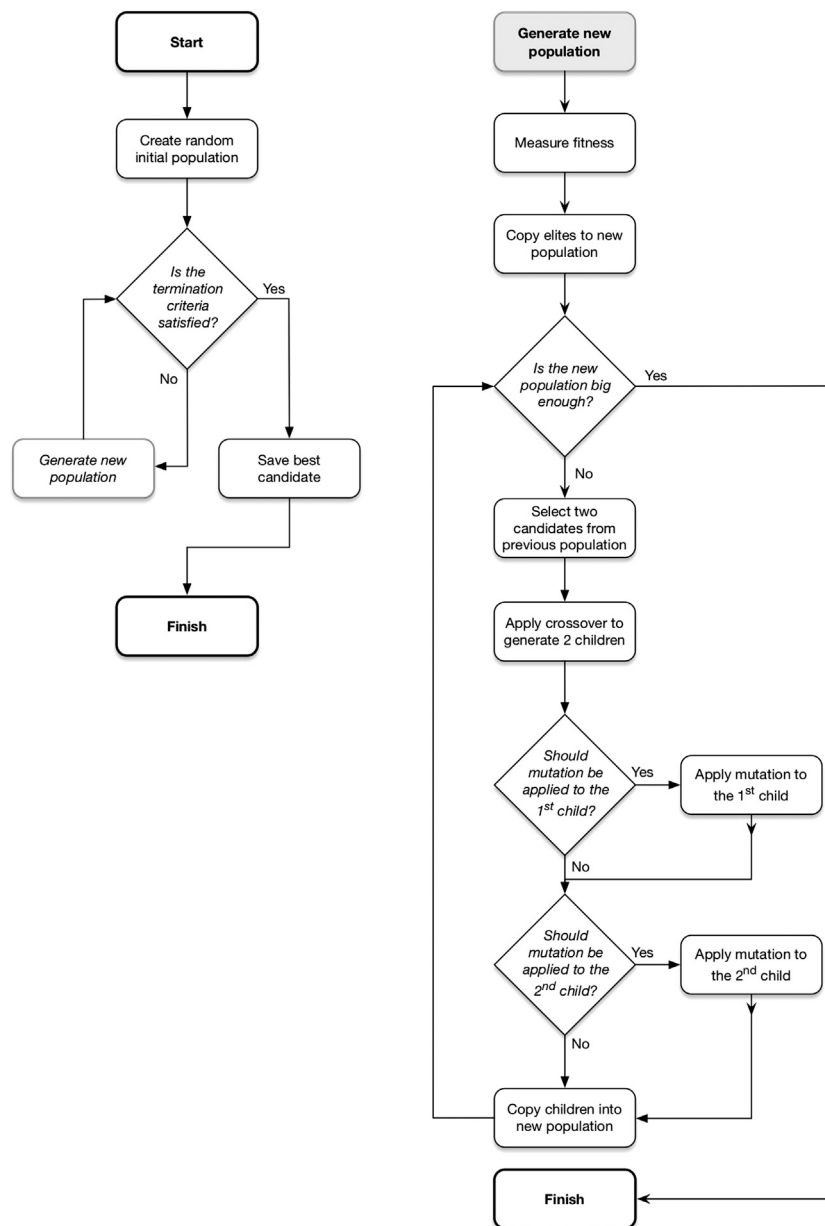


FIGURE 1 | The general process followed by an evolutionary algorithm. Recreated from Koza (2018). Note that the number of parent candidates selected and the number of children generated depends on the operator. This example shows two parents generating two offspring.

of the algorithm is to generate exercises which are both technically appropriate and aesthetically pleasing. It does so by using expert models of professionally-written sight reading exercises as templates for emulation.

There are four primary reasons for selecting EAs in this work. First, Biles (2007) notes that evolutionary approaches have been applied to melody generation problems more often than any other technique, and with more success. Secondly, the solution space when generating a melody is large, and evolutionary algorithms are well suited to navigating that space (Johnson et al., 2004). Thirdly, evolutionary algorithms allow for specific

goals to be set for a solution while still providing space for random elements and emergent behaviors to appear. This means that the solutions found by the algorithm are likely to maintain a higher level of variability compared to other methods, even when using identical configurations. Lastly, preliminary experiments using probability-bound random sampling indicated that simpler approaches were not able to satisfactorily handle the complexities of the problem.

Figure 1 shows the general process followed by EAs. An EA begins with a randomly generated set of candidate solutions, referred to as a ‘population’, then follows an iterative process until

some termination criteria is met. This iterative process involves generating a series of new candidate populations, each of which is based on the previous. The aim is that over time the populations will contain incrementally superior solutions to those in previous populations.

First, all candidates are measured for their suitability as a solution and assigned a corresponding numerical value (i.e., fitness value). The top n best or “elite” candidates (where n can be 0) are then directly copied into the next population without any alterations. The remainder of the new population is formed through the application of the genetic operators *crossover* and *mutation*. Crossover is a reproduction method used to create candidates by combining elements from two ‘parents’ from the previous population. The mutation operator applies small random changes to a candidate in order to introduce diversity into the population.

Once the new population has reached its target size, the termination criteria are checked. If they have been met, the candidate with the highest fitness over all iterations is returned as the solution. If not, the process is repeated.

The algorithm requires a number of aspects be defined:

- **Population size** The number of candidate solutions in a population. If this value is too small the algorithm may converge on a suboptimal solution due to lack of diversity. However, if this value is too large the algorithm may take an excessive amount of time to finish.
- **Termination criteria** When the algorithm should stop. It is typically a target fitness value, a specific number of iterations, a number of iterations without improvement, or a combination of the three. For example, target a fitness of 0.95, but if it hasn’t been reached within 1,000 iterations terminate the algorithm anyway.
- **Fitness function** A numerical measure for quantifying the suitability of a candidate solution. This dictates the likelihood that a candidate will be selected to be part of the next population.
- **Number of elites** The number of top candidates from the previous population that will be directly copied to the new population without any adjustments.
- **Genetic operators** How the *crossover* and *mutation* operators will be implemented.
- **Selection method** The method for selecting candidates for the crossover operator. Typically a function of each candidate’s fitness value.
- **Probability of mutation** How likely it is that candidates resulting from the crossover operator will be mutated.
- **Candidate representation** How each candidate is encoded.

Section 2 will describe the method used in this work. This includes the curation of a suitable set of expert models, the technique used to represent candidate solutions, and the experimental design. The results of this experimental design will be detailed in **Section 3**. Finally, **Section 4** will discuss the implications of these results and potential directions for future work.

TABLE 1 | Summary of expert-written example exercises extracted from published books.

| Book | Grade 1 | Grade 2 |
|---|---------|---------|
| Improve your sight-reading! (Harris, 1994) | 25 | 16 |
| Flute sight-reading (Selleck, 2012) | 12 | 12 |
| Sound at sight - sight reading pieces for flute; book 1 (Rae, 2007) | 20 | 20 |
| Flute specimen sight reading | 15 | 16 |
| The Associated Board of the Royal Schools of Music (1995) | | |
| Total | 72 | 64 |

2 METHOD

2.1 Building Expert Models

Four books of sight reading exercises were selected, representative of the curricula of the Australian Music Examinations Board (AMEB), Associated Board of the Royal Schools of Music (ABRSM), and Trinity College. Grade 1 and 2 exercises were extracted from each book, as summarized in **Table 1**. A expert model was derived from each exercise, capturing the following characteristics:

- Key and time signatures
- length,
- range,
- number of ties and rests
- ratio of notes to rests
- proportions of note lengths, rest lengths, and intervals, and
- melody shape (defined in **Section 2.3.2**).

These characteristics can be viewed as a whole to gain an appreciation of a “typical” exercise at the Grade 1 and 2 difficulty levels. They can also be considered individually to see a distilled view of the key characteristics of each individual exercise. In practice, this data will primarily be used at the level of an individual exercise, where the set of characteristics relating to one exercise is used to form a single expert model. This is discussed further in **Section 2.4.1**.

2.2 Exercise Representation

Many published works in the field of melody generation are not clear on how melodies are represented. However, two primary themes emerge: tree-based and sequential structures (Biles, 2007). This is true for works which both do and do not utilize evolutionary algorithms.

Sequential structures, such as that used by Acevedo (2004), represent melodies as ordered lists of musical elements. These elements are typically individual notes and rests, with each having a length and (where appropriate) a pitch. Pitches can be represented absolutely (e.g., C4), or as an offset from some epoch. Length can also be represented absolutely (e.g., crotchet), or as a start and end time offset from the start of the melody. In some cases, elements may also contain ornamental information such as dynamic or articulation markings. Melodies represented this way are read by examining the sequence of musical elements in order.

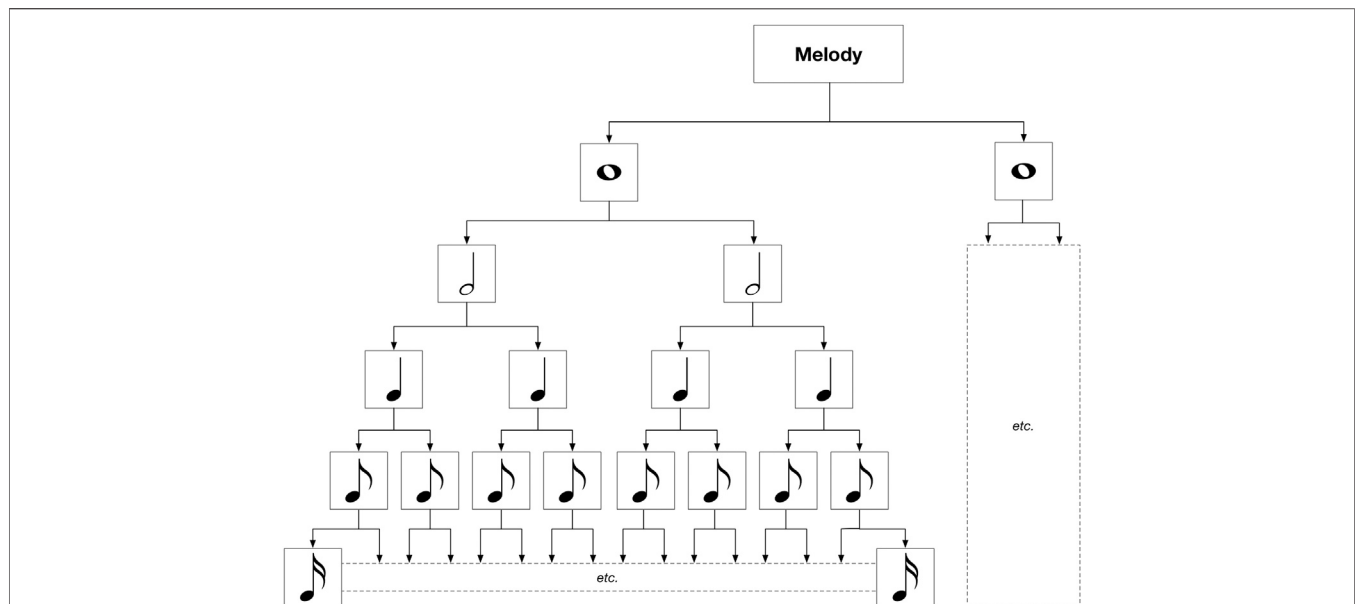


FIGURE 2 | The duration hierarchy typically used by tree structures which are designed to represent melodies in common time. The bottom row of semiquavers has been truncated for space, and the right-hand side of the tree is not shown to completion as it is identical to the left.

Regardless of the specific encoding scheme, sequential structures are not an ideal choice for an evolutionary algorithm. This is particularly true in this work, where the desired result is a melody of an explicit, fixed length. The practical reason for this lies in the crossover operator. This operator swaps sections of two parents to create two new candidates. When using a sequential structure, it is easy for the newly created candidates to have different lengths, simply by choosing asymmetrical crossover points. It is also easy for crossover to create candidates whose note and rest sequences do not fit neatly into entire bar lengths. For example, if two 4-bar parents were selected, while asymmetrical crossover could result in two children with 4-bar lengths it is much more likely to create two children with different lengths and partially complete bars (e.g., 3.4, and 4.6 bars). Symmetrical crossover would eliminate this problem, but would severely reduce the variety of new candidates that could be created, as the selected crossover points would always need to be symmetrical.

While not without fault, tree-based structures avoid these problems entirely. As such, they are used in this work.

2.2.1 Tree-Based Structures in the Literature

In the literature, tree-based solutions for melody representation typically follow a binary structure where each node represents a musical element with half the duration of its parent. This means that the structure of a melody tree adheres to the duration hierarchy shown in **Figure 2**, where the length of a note is entirely dependent on its depth within the tree. The root of the tree represents the entire melody, with nodes in the first layer representing individual bars. From this point onwards each additional layer of depth splits durations in two. For example, in a $\frac{4}{4}$ melody nodes with a depth of 1 would represent

semibreves, nodes with a depth of 2 would represent minims, nodes with a depth of 3 would represent crotchets, and so on. Within this structure only leaf nodes represent concrete musical elements that would be directly shown on a score. Internal nodes, at a minimum, serve to maintain the duration hierarchy. However, some implementations also assign some or all internal nodes special meaning in order to support additional functionality. When interpreting a melody tree, leaf nodes on the left are typically played before leaf nodes on the right.

Table 2 shows a comparison of the characteristics of the melody trees in the literature. All of the trees are able to represent monophonic melodies and dotted notes. Additionally, all three representations allow for subtrees from two different melodies to be swapped at any point, without breaking the tree structure.

The trees proposed by Rizo et al. (2003) and de León et al. (2016) both support simple time signatures—that is, time signatures such as $\frac{4}{4}$ and $\frac{2}{4}$ where measures can recursively be divided into equal halves without the need for dotted notes. However, neither of these trees support compound time signatures such as $\frac{6}{8}$ and $\frac{9}{8}$, where measures do not neatly fit into a binary structure. They also do not support simple time signatures such as $\frac{3}{4}$ which do not neatly divide into two.

Dahlstedt's tree is noted as supporting neither simple nor compound time. This is because the tree does not structure its nodes according to a duration hierarchy. Instead, it assigns the duration of nodes independently of one another. A time signature is applied to the melody when translating it to a score rather than within the tree itself, and the melody has no guarantee of fitting within the chosen time signature. This means that Dahlstedt's tree, unlike the trees of Rizo et al. (2003) and de León et al. (2016), also does not maintain a musical grammar. That is, it can not

TABLE 2 | A comparison of the features of the melody trees proposed in the literature.

| Feature | Rizo et al. (2003) | Dahlstedt (2007) | de León et al. (2016) |
|--------------------------|--------------------|------------------|-----------------------|
| Monophony | ✓ | ✓ | ✓ |
| Polyphony | ✗ | ✓ | ✗ |
| Dotted notes | ✓ | ✓ | ✓ |
| Tied notes | ✗ | ✓ | ✓ |
| Irregular divisions | ✗ | ✗ | ✗ |
| Simple time | ✓ | ✗ | ✓ |
| Compound time | ✗ | ✗ | ✗ |
| Maintain musical grammar | ✓ | ✗ | ✓ |
| Crossover anywhere | ✓ | ✓ | ✓ |

guarantee that the represented melody will fit neatly into any particular time signature.

None of the melody tree structures discussed explicitly support irregular divisions such as triplets or tuplets. This severely limits their representational capacities. Rizo's tree has an additional problem, in that it does not offer a mechanism for representing tied notes.

For representing sight reading exercises a melody tree would, at a minimum, need to support:

- monophonic melodies
- dotted notes
- tied notes
- triplets
- simple time signatures
- compound time signatures,
- enforceable musical grammar, and
- swapping of subtrees at any point.

Additional features that would be useful for representing melodies include support for

- polyphonic melodies
- multiple time signatures in one melody
- ornamental and stylistic markings (e.g., mordents, trills), and
- additional irregular divisions (e.g., duplets, any variation of the "x in the time of y" pattern).

These additional features are not necessary for the task of generating monophonic sight reading melodies of low level difficulties, and thus are left as future work.

None of the trees in the surveyed literature support the necessary combination of minimally viable features. As such, a novel melody tree was created that would meet this criteria. This novel tree is described in **Section 2.2.2**.

2.2.2 Designing a Novel Melody Tree

Several of the minimally viable features for a melody tree are already supported in existing trees. This is capitalized upon in this work by taking elements from existing trees where possible then adding the additional, missing functionality necessary for representing musical sight reading exercises.

Of the trees in the literature, that proposed by Rizo et al. (2003) offers the most desired features, thus will act as a starting point for a novel tree structure. The features covered by this tree include support for:

- monophonic melodies
- dotted notes
- tied notes
- simple time signatures,
- enforceable musical grammar, and
- swapping subtrees at any point.

This leaves two key features absent:

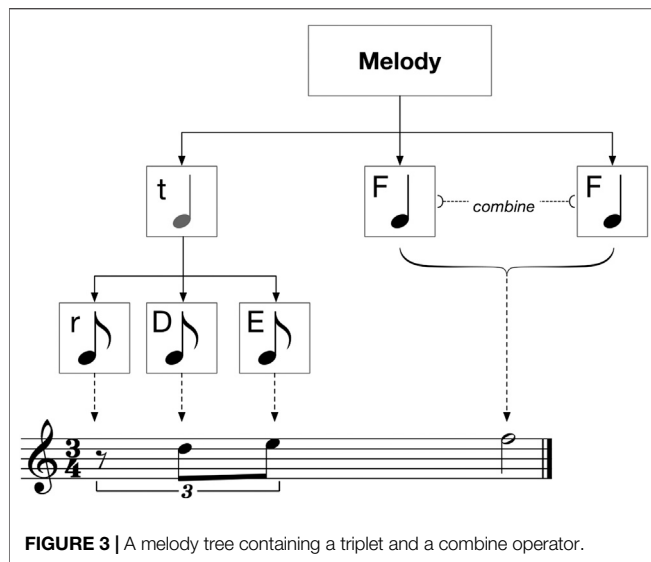
- support for compound time signatures
- Support for triplets

The implementation of these two features is discussed below.

2.2.2.1 Supporting Compound Time Signatures

The ability to swap subtrees at any point while enforcing musical grammar is entirely due to a tree following a duration hierarchy as described in **Section 2.2.1**. Unfortunately, this encourages a binary structure, which is not ideal for representing compound time signatures or other time signatures which do not neatly divide into two. For example, to represent a melody in $\frac{3}{4}$ time (a simple time signature that does not neatly divide into two), splitting bars equally would result in the first layer of nodes representing dotted crotchets. Continuing this pattern, the following layer would contain nodes representing half that value again—a dotted quaver. The next layer would represent dotted semiquavers, then dotted demisemiquavers, and so on. This pattern results in a structure where node lengths are unnecessarily complex, and individual nodes do not represent lengths commonly found in melodies (i.e., non-dotted lengths).

An alternative strategy might be to split any compound-lengthed node into two non-equal but more typical lengths. Returning to the example of a melody in $\frac{3}{4}$ time, this would result in the first layer being a combination of a minim and crotchet node. The next layer would then comprise two crotchet nodes (from splitting the minim) and two quaver nodes (from splitting the crotchet).



This approach presents two problems. Firstly, it breaks the duration hierarchy which requires that all nodes at the same depth have the same length. This complicates the continuation operator as no assumptions can be made regarding a node's length with respect to its depth. It also adds more complexity when swapping subtrees in ensuring nodes are reassigned the correct length given their new depths.

Secondly, it introduces a decision regarding which node should be left-most in the tree—the longer or shorter of the split? For example, when splitting a $\frac{3}{4}$ bar should the minim or crotchet node be left-most? This choice informs how elegantly a melody can be represented and how often continuation operators need to be used to form longer note lengths.

The solution to these problems is for bars of a time signature $\frac{n}{m}$ be split into n nodes of m length, where m indicates the number of that length note required to equal the length of a semibreve. So, $m = 1$ indicates a semibreve, $m = 2$ indicates a minim, $m = 4$ indicates a crotchet, and so on.

This strategy means that the second layer of a melody tree is non-binary, but the remainder is. Unfortunately, this gives rise to one problem. In order to successfully implement the crossover operator, it must be possible to swap any subtree from one melody with any subtree from another. This is an issue when the second layer of the tree is non-binary, as the parent node of a non-binary layer (i.e., the node representing a single bar) may be swapped with the parent node of a binary layer.

The solution to this problem is to remove the layer of 'bar' nodes entirely, meaning that the tree starts with a layer containing $n \times \text{number_of_bar}$ nodes of m length. This means that the first layer of the tree may contain many nodes, but every one of those nodes is binary and has children with exactly half of their length. Additionally, because the value of 'm' is taken from the time signature, these nodes are guaranteed to be of a length which can be recursively split into two equal, non-compound halves. Notes longer than m can still be represented through the use of one or several linked continuation operators.

2.2.2.2 Supporting Triplets.

Triplets are implemented with an internal node operator similar to the *split* and *continuation* operators used by de León et al. (2016). As shown in **Figure 3**, the triplet operator is placed on the first direct parent of the triplet leaf nodes. If all leaves within the triplet are of the same length, the triplet operator is placed one layer above. If the leaves within the triplet are of different lengths, the triplet operator is placed on the first common parent.

The triplet operator does not break the duration hierarchy, nor does it restrict the swapping of subtrees. If the triplet operator itself is selected to be swapped, the entire triplet is moved. If a subtree within the triplet is selected to swap, notes within that subtree will—assuming they are swapped to a non-triplet parent node—be interpreted as having a standard, non-triplet length. Conversely, the subtree swapped into its place will then be interpreted as part of the triplet.

2.3 Evolutionary Operators

2.3.1 Parent Selection

In this work, Pareto selection is used. Instead of considering a candidate's overall fitness value, Pareto selection examines fitness in terms of individual characteristics. This is a relative probability measure which is useful for situations where a single fitness value does not make sense (Horn et al., 1994; Fonseca and Fleming, 1995).

For example, consider the task of evolving a box with an appropriate width, depth, height, strength, and weight. Here, an overall or combined fitness value will not work, as perfection in one aspect of the box does not offset weakness in another aspect. That is, better fitness in height does not compensate for poor fitness in depth. Similarly, good fitness in width does not make up for poor fitness in strength. Pareto selection deals with this issue by considering the individual aspects of fitness. The probability that a candidate will be selected is based on the number of other candidates in the same population that it is superior to in every aspect. Using the box example, a candidate is only better than another candidate if it has a superior width, depth, height, strength, and weight. Once this value is known, **Eq. 1** can be used to determine the selection probability for a candidate.

$$\text{probability_of_selection}_i = \frac{(1 + W_i)}{\sum_{j=1}^n W_j}$$

W_i : the number of candidates the i^{th} candidate in a population is superior to in every aspect

n : the total number of candidates in the population

(1)

The task of evolving musical sight reading exercises benefits from the use of Pareto selection. An exercise needs to meet multiple criteria, both technical and esthetic. For example, an exercise at the Grade 1 level might need to use only crotchet lengths and have only one or two rests. Additionally, the melody might also need to meet esthetic criteria such as beginning and ending on the tonic note. As with the box problem, an overall fitness value does not work for these requirements. A good selection of note lengths

does not make up for a lack of esthetic qualities. Similarly, a melody sounding good does not make up for an absence of appropriate technical characteristics. As such, Pareto selection is an ideal solution.

2.3.2 Fitness Measures

The fitness measures are designed to guide the evolutionary process toward creating a melody with a specific set of characteristics. For this work six measures are used, each of which has an associated target value. A melody is assigned a score in the range [0..1] for each measure. This score is calculated using **Eq. 2** as the difference between the candidate's actual and target value for a fitness measure.

$$\text{fitness}_{fi} = 1.0 - \text{abs}(t_f - a_{fi}) \quad (2)$$

t_f : the target value for fitness measure f
 a_{fi} : the actual value for fitness measure f for candidate i

To illustrate this idea, return to the example of evolving a box. A target height for the box may be set as 10 cm. If a candidate box had a height of 10 cm it would receive a score of 1.0 for the 'height' measure. However, if the box had a height of 5 cm it would receive a score of 0.5. Similarly, if the box overshot the target with a height of 15 cm it would also receive a score of 0.5.

Each of the six fitness measures used in this work are based on counting a specific element within the melody. These counts are described as being either "time" or "count" based. A count-based measure takes the count as a raw value. For example, 6 notes in the melody are crotchets. Time-based measures take the raw count value and interpret it as a proportion of melody time. For example, in a 4 bar melody in $\frac{4}{4}$ time a count of 8 crotchets would be interpreted as 50% of the melody being crotchets, because the melody could potentially fit a total of 16 crotchets.

The six fitness measures used in this work are:

- **Target note lengths (time-based)** The proportion of melody time to be taken by each note length. For example, 50% of the melody time should be filled by crotchets; 25% of the melody time should be filled by quavers.
- **Target rest lengths (time-based)** The proportion of melody time to be taken by each rest length. For example, 25% of the melody time should be filled by crotchet rests.
- **Allowable lengths (count-based)** The acceptable lengths for notes and rests in the melody. For example, only use notes and rests with crotchet or quaver lengths.
- **Target intervals (count-based)** The proportion of each size of interval to include. Size is represented in scale degrees. For example, 50% of intervals should be 1 scale degree in size; 50% of intervals should be 2 scale degrees in size.
- **Allowable intervals (count-based)** The acceptable interval sizes to use in the melody. Size is represented in scale degrees. For example, only use intervals with sizes of 1 or 2 scale degrees.
- **Melody shape (count-based)** The number of segments in the melody containing three contiguous notes where the

itches move consistently up or down. For example, 80% of the melody segments should be shapely.

The target note proportions and target rest proportions should sum to represent exactly 100% of the melody time. Similarly, the target interval proportions should sum to represent 100% of the intervals. The allowable lengths and allowable intervals are derived automatically from the target note, rest, and interval proportions. For example, if target proportions are set for crotchet and quaver notes, and a target proportion is set for crotchet rests, the allowable lengths are crotchets and quavers. Similarly, if target proportions are set for intervals of size 1, 2, and 3, the allowable intervals are 1, 2, and 3.

The purpose of combining "target" and "allowable" measures rather than just using one or the other is to guide the algorithm toward rewarding the use of an "allowable" length more than an undesirable length, even if doing so breaks the target proportions. The "target" measures represent the ideal proportions of note lengths and intervals. However, if the algorithm is struggling to reach the ideal targets it is better to sacrifice melodies score with respect to the targets, in the interest of still only using "allowable" note lengths and intervals. This is because introducing "unallowable" note lengths or intervals is a much bigger problem for the overall fitness-for-purpose of an exercise than having slightly incorrect proportions. That is, if only crotchets and quavers are "allowable" and the algorithm can not form an exercise with the desired proportions of these note lengths, it is still better for the algorithm to use extra quavers and potentially lower the score for *target note proportions* than to introduce some other note length that is considered inappropriate.

The "melody shape" measure is illustrated further in **Figure 4**. In this example, the melody contains a total of ten segments. Note that segments containing rests or fewer than three notes are not counted. Of these ten segments, only four contain notes which move consistently up or down in pitch. Therefore, in this case the melody shape is $\frac{4}{10}$, or 0.4.

Consider the example target values for a melody set out in **Table 3**. These targets indicate that the evolutionary algorithm should attempt to create a melody made entirely of crotchets, most of which are notes (as opposed to rests). They specify a large target proportion for intervals with a size of 1, but also requests that some intervals of 0 and 2 scale degrees be used. The melody shape target is set to 0.3, meaning that 30% of the segments in the melody should have notes which move consistently up or down in pitch.

Now consider the melody shown in **Figure 5**. To calculate the fitness value for each measure we must calculate the actual proportions and numbers of notes, rests, and intervals, and compare them to the targets. For measures such as "target note/rest proportions", "target interval proportions", and "melody shape", the fitness value is calculated according to **Eq. 2**. The remaining fitness measures are calculated as raw counts, as their target is for 100% of the melody to fit within the assigned parameters. For example, the fitness for "allowable intervals" is calculated as the number of intervals of an allowable size divided

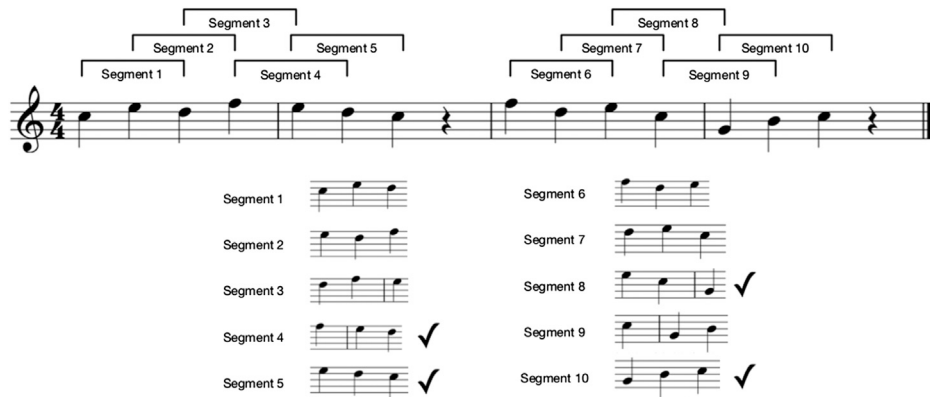


FIGURE 4 | Calculating the shape of a melody. Tick marks indicate the segments which are counted as having *shape*, as the pitches within the segment move consistently up or down.

TABLE 3 | Calculating the fitness of the melody in **Figure 5** against an arbitrarily selected set of targets. The final fitness value for each measure is in bold.

| Fitness measure | Target | Actual | Fitness |
|-------------------------|----------------|-------------------------------------|---|
| Target note proportions | 0.75 crotchets | Crotchets | $1 - \text{abs}\left(0.75 - \frac{13}{16}\right) = \mathbf{0.94}$ |
| Target rest proportions | 0.25 crotchets | $\frac{2}{16}$ crotchets | $1 - \text{abs}\left(0.25 - \frac{2}{16}\right) = \mathbf{0.88}$ |
| Allowable lengths | Crotchets | $\frac{15}{17}$ allowable Lengths | $\frac{15}{17} \approx \mathbf{0.88}$ |
| Target interval | 0.3 size 0 | $\frac{0}{13}$ intervals Of size 0 | $1 - \text{abs}(0.3 - 0) = 0.7$ |
| Proportions | 0.5 size 1 | $\frac{9}{13}$ intervals Of size 1 | $1 - \text{abs}\left(0.5 - \frac{9}{13}\right) \approx 0.81$ |
| | 0.2 size 2 | $\frac{3}{13}$ intervals Of size 2 | $1 - \text{abs}\left(0.2 - \frac{3}{13}\right) \approx 0.97$ |
| | | | $\frac{(0.7+0.81+0.97)}{3} \approx \mathbf{0.83}$ |
| Allowable intervals | 0, 1, and 2 | $\frac{12}{13}$ allowable Intervals | $\frac{12}{13} \approx \mathbf{0.92}$ |
| Melody shape | 0.3 | $\frac{6}{11}$ shapely Segments | $1 - \text{abs}\left(0.3 - \frac{6}{11}\right) \approx \mathbf{0.75}$ |



FIGURE 5 | An example melody.

by the total number of intervals in the melody. **Table 3** shows how each fitness measure would be calculated for this melody, given an arbitrarily selected set of targets.

2.3.3 Crossover

The implementation of crossover is reasonably straightforward. First, two parent candidates are selected using Pareto selection. Then, a node from the melody tree of each candidate is randomly picked. The subtrees starting from these nodes are taken from each parent and their positions are swapped.

Once the subtrees are swapped, the durations of their nodes are altered with respect to their overall depth within their new tree. For example, if a node is placed one layer higher in its new tree than in its old tree, its duration is doubled. Similarly, a node placed one layer lower would have its duration halved. No additional alterations are made.

This simple implementation is possible because the melody tree representation ensures that no matter which subtrees are swapped the resulting melody trees will still be grammatically correct. Additionally, the length of the melodies remains fixed

as the node durations are adjusted according to their new depths.

2.3.4 Mutation

As the candidates in this work are melodies, the potential mutations are musical in nature and specific to the domain. One alteration type is randomly selected from the following:

- **Change note type** Randomly select a leaf node. If the node represents a note, change it to a rest. If the node represents a rest, change it to a note with a random pitch.
- **Split node** Randomly select a leaf node. Change that node into an internal node with two randomly initialized children.
- **Reduce node** Randomly select a leaf node. Remove that node and its siblings and randomly reinitialize their parent.
- **Reinitialize note** Randomly select a node representing a note (not a rest). Reinitialize the node with a random pitch.
- **Add triplet** Randomly select any node within the tree. If the node is a leaf, change it to an internal node, add the triplet operator, and randomly initialize and add three children to it. If the node already has children, add the triplet operator and randomly initialize and add a third child to it.
- **Remove triplet** Randomly select any node within the tree that has a triplet operator attached. Remove the triplet operator then randomly select one of the node's children and remove the subtree from that point.
- **Add continuation** Randomly select any leaf node within the tree that does not already have a continuation operator attached. Add a continuation operator to the node. Then, if the next leaf node in the tree is not the same type, change it so that it is. For example, if the randomly selected node is a note and the next leaf node is a rest, change the rest to a note with the same pitch as the randomly selected node.
- **Remove continuation** Randomly select any node that has a continuation operator attached. Remove the operator.

Any change that is not possible is not considered when randomly selecting an alteration. For example, if a melody does not contain any triplets then “Remove triplet” will not be selected. The algorithm configuration also allows for individual mutation operators to be disabled regardless of whether they are possible or not for any given candidate. Additionally, the random elements of mutation (e.g., giving a note a new randomly selected pitch) are constrained with respect to a target range and key signature as dictated by an expert model.

2.4 Experimental Design

2.4.1 Overview

The design of the algorithm allows for exercises to be generated for any monophonic instrument. However, for consistency and comparability between results only a single instrument—the flute—is used in this experimental design. The flute was chosen as it is both monophonic and non-transposing. It is also a relatively popular instrument, meaning there are a large number of sight reading exercise books published for it. This is important as parameter sets for

the algorithm were derived from the characteristics of expert models, which are in turn derived from published books of exercises. This ensures that the targets set for the algorithm are both realistic and grounded in accepted, widely-used, professionally written examples.

Grades 1 and 2 were chosen as the difficulty levels with which to validate the algorithm's capabilities. The reason for selecting these earlier grades lies in the utility of the exercises. Formal exams at the Grade 1 level are a student's first exposure to musical sight reading, so naturally students at this level find large quantities of practice material useful. A wide variety of practice exercises is also useful at other early grade levels as students come to grips with sight reading techniques. Students studying later levels—typically Grade 5 and above—often require fewer practice exercises. There are likely two reasons for this. Firstly, students at this level should already have a solid foundation of sight reading skills, and thus require less practice material. Secondly, at these levels students can use entire pieces from lower grades as sight reading exercises, reducing the need for purpose written material.

As discussed in **Section 2.1**, the set of individual expert-written sight reading exercises extracted from published exercise books were transformed into a set of expert models, where each model corresponds to one published exercise. The reason for modeling exercises individually rather than as a collective is twofold. Firstly, targeting only the most typical characteristics of a group of exercises ignores the variety in musical and technical content in exercises of even the lowest level of difficulty. This severely limits the variety of solutions the algorithm can generate, as it would be constrained by a single expert model for any one difficulty level. The second reason relates to the purpose of the experimental design in exploring the capabilities of the algorithm. If the algorithm is tasked only with generating ‘typical’ sight reading exercises, the results only indicate how it performs generating typical sight reading exercises. To gain a wider understanding of the algorithm's performance, it should be tested on a broader range of input. Using models based on individual expert-written examples allows this to be done.

2.4.2 Algorithm Configurations

Given a configuration, the algorithm generates tree structures using an evolutionary approach, as described in **Section 2**. When the algorithm is finished the best candidate is converted into a text format supported by music21^a, which is then used to render the solution and save it in the MusicXML format.

A unique algorithm configuration was created for each expert model. Each of these configurations (i.e., parameter sets) covers three elements:

- Fixed characteristics,
- target characteristics, and
- evolutionary parameters.

The fixed characteristics represent targets for the algorithm which can be set or ‘hard-coded’ during initialization. These characteristics are:

- Length (number of bars)
- time signature
- key signature
- range
- use of ties, and
- use of rests.

There are two reasons for setting these characteristics as hard limits rather than targets that may or may not be achieved. Firstly, as discussed in **Section 2.2.2**, both the length and time signature are required to form the structure of the melody tree used to represent candidate solutions. Given this, they need to be fixed during initialization, and can not change during the evolutionary process without requiring fundamental alternations to the melody tree structure.

The second reason relates to the key signature, range, use of ties, and use of rests, in that setting non-negotiable limits on these characteristics significantly reduces the search space the algorithm needs to explore. If the key signature and range is known, random pitch selection can be restricted to a pool of pitches which are both in the given key signature and within the given range. As pitches outside the target range and key signature add no value to a candidate solution, this approach serves only to reduce the search space—it is highly unlikely to reduce the quality of the final solution, only the time needed to find it.

Similarly, if the melody of the expert model being used does not include any rests or ties, then rests and ties should not be introduced into the solution space. That is, mutations relating to rests and ties should not be applied, and the initial population of candidate solutions should not contain any rests or ties.

The target characteristics relate to specifying target values for each of the six fitness measures defined in **Section 2.3.2**. Unlike the fixed characteristics, these targets may not be perfectly met. Targets for the *target note lengths*, *target rest lengths*, *target intervals*, and *melody shape* measures can be found directly within an expert model. Targets for the remaining two measures—*allowable lengths* and *allowable intervals* can be inferred from these values. This was discussed in **Section 2.3.2**.

Finally, the evolutionary parameters relate to values which influence the evolutionary process but which are not specific to the chosen application of generating musical sight reading exercises. These remain static regardless of the expert model being used:

- Population size: 50
- Number of elites: 1
- Probability of mutation: 0.01 (i.e., 1%)
- Random function: Gaussian
- Selection method: Pareto
- Termination criteria: 100 generations with no improvement in the best candidate

Each configuration is also assigned a fixed random number generator seed, so that its output can be reproduced.

The parameters of the algorithm are intended to be specific enough to guide the evolutionary process, but still broad enough that there are many acceptable solutions for a given expert model.

TABLE 4 | Criteria for assigning each generated sight reading exercise a rating.

| Rating | Fit for purpose? | Percentage of melody violating ruleset |
|-----------|------------------|--|
| Very good | Yes | — |
| Good | Yes | ≤10% |
| Average | Yes | ≤25% |
| Bad | No | >25% |
| Very bad | No | Unplayable elements |

To show this, three configurations were created for each expert model, differing only in their random number generator seed. This means that after being executed with every configuration the algorithm will have generated three different sight reading exercises for each expert model. Comparing these results will indicate how consistent the algorithm is in finding acceptable solutions, and the similarities between solutions generated using the same set of targets.

2.4.3 Evaluating the Generated Exercises

Describing exactly what makes a musical sight reading exercise fit for purpose requires quantifying both the esthetics of the melody as well as its technical appropriateness with respect to a specific difficulty level and instrument. Either one of these tasks is uniquely challenging on its own. Some guidance can be found by examining existing published exercises. These mostly reveal the technical properties which are appropriate for each difficulty level, such as the acceptable note lengths, interval sizes, and use of syncopation. Additional guidelines can be inferred. For example, some sequences of notes are clearly unplayable. Other components, such as repeated large intervals between notes, are widely accepted as being difficult to play (Schoenberg, 1967).

Other factors to consider relate to the esthetics of the melodies. Musical esthetics are notoriously difficult to quantify and remain the subject of much debate and ongoing research. One reason for this is that musical rules tend to be derived empirically, in that they emerge by examining trends in common practice rather than being determined *a priori*. Another reason is that they are largely contextual, depending on the culture, genre, age, and purpose of the music in question. This means that not only are they open for discussion and interpretation, but they also evolve over time.

Fortunately, the application domain of the algorithm presented in this work naturally restricts the scope of the musical rules that need to be considered. The purpose of the proposed algorithm is to generate sight reading exercises which would be suitable for students preparing for formal musical examinations such as those facilitated by the Australian Music Examinations Board (AMEB) and the United Kingdom's Associated Board of the Royal Schools of Music (ABRSM). The curricula developed by these organizations strictly fall within Western Classical Music from the Common Practice period. This is a relatively well-documented period with a number of widely accepted musical guidelines for aesthetically pleasing melodic and harmonic structures. As the algorithm focuses on monophonic melodies, only guidelines relating to melodic structures need to be considered.

Additionally, sight reading exercises are uniformly short in length, meaning that complex concepts of musical form, which

describe formal structures for musical pieces to follow, do not apply.

Each generated exercise was evaluated against a ruleset containing 29 rules relating to technical appropriateness and melodic esthetics. Evaluation was performed by one person who has over 20 years of musical experience, and who has obtained formal musical qualifications in both performance and musical theory studies. The proportion of an exercise in violation of the ruleset was translated to a Likert quality rating on a five-point scale according to **Table 4**. This means that although the weighting of the rules is equal, their impact on the final score for a melody differs as some are easier to violate than others. Exercises assigned a “Very good”, “Good”, or “Average” rating are said to be “fit for purpose”, with the remainder being ‘unfit for purpose’.

Each exercise is also given a rating based on whether it can be improved or upgraded with a small number of alterations. An exercise is said to be “improved” if it was already fit for purpose and becomes more so as a result of the changes. Alternatively, an exercise is said to be ‘upgraded’ if the changes transform it from being unfit to being fit for purpose. Currently, changes are made manually when assessing each exercise.

When determining potential changes, at most 5% of the melody can be altered. Acceptable alterations include changes to note pitches, note and rest lengths, and note and rest placements. Only changes which could be represented algorithmically should be used, as it is intended that these changes could be incorporated into the algorithm in the future. Once a small set of changes is made, an exercise is evaluated again with respect to the ruleset and a new Likert rating is assigned.

The post-alteration ratings serve a dual purpose. They show the potential of the algorithm to be improved, and serve to highlight the biggest problems currently preventing the generated exercises from being more fit for purpose. This information could be used to drive future work.

The evaluation ruleset was derived from an examination of relevant literature and published expert-written examples (i.e., expert models) of sight reading exercises. As well as relating to the technical appropriateness of an exercise, the melodic esthetics of an exercise, or both, each individual rule can be further categorized as relating to one of four facets:

- Note/Rest selection

Rules relating to the length, pitch, and location selected for each note and rest in the melody.

- Intervals

Rules relating to the size and placement of intervals.

- Melodic structure

Rules relating to the shape and form of the melody.

- Rhythmic structure

Rules relating to the sequences of note and rest lengths in the melody.

Many of the rules refer to the “strong” beats of a melody. Beats which are seen as “strong” depend on the time signature. In time signatures where bars can be divided into two equal parts, the first beat and the beat half-way through are strong (e.g., beats 1 and 3 in $\frac{4}{4}$ bars are strong). In all other time signatures the first beat of the bar is strong.

Some rules also refer to scale degrees, either numerically or in roman numerals. In this case, the scale degrees should be interpreted with respect to the target key signature of the generated exercise (e.g., scale degree 3 in C major would indicate the pitch E).

Many of the rules are written generally, for example “An exercise should only use rest lengths seen in expert models of the same difficulty level.” In order to implement the ruleset a reasonable sample of expert models need to be collected. For the application of the algorithm presented in this work these expert models are those for the flute described in **Section 2.1**. The ruleset can easily be translated to evaluate exercises for other monophonic instruments by replacing this set of expert models with another specific to the instrument being considered.

Table 5 summarizes the ruleset, including the source from which each rule was derived. The individual rules are defined as follows:

- Rest proportions

No more than 10% of the melody should be made up of rests, unless a larger proportion is present in the target expert model.

- Note lengths

An exercise should only use note lengths seen in expert models of the same difficulty level. For example, Grade 1 exercises for the flute are expected to only use minim, crotchet, quaver, semiquaver, and semibreve length notes.

- Rest lengths

An exercise should only use rest lengths seen in expert models of the same difficulty level. For example, Grade 1 exercises for the flute are expected to only use crotchet, quaver, and semiquaver length rests.

- Tied notes

Grade 1 exercises for the flute should not contain any tied notes.

Grade 2 exercises for the flute should contain at most 5% tied notes.

If the target expert model contains a larger proportion of tied notes than those listed here, the maximum percentage of tied notes an exercise can contain is that of the target expert model.

- Interval sizes

TABLE 5 | The origin of each rule in the ruleset for evaluating algorithmically generated sight reading exercises. Note that there are no rules for evaluating just the melodic esthetics of rhythmic structures, only technical appropriateness alone or technical appropriateness and melodic esthetics combined.

| Rules Evaluating . . | Facet | Rule | Origin |
|--|----------------------------|---|---|
| <i>Technical Appropriateness</i> | <i>Note/Rest selection</i> | 1. Rest proportions | Expert models |
| | | 2. Note lengths | Expert models |
| | | 3. Rest lengths | Expert models |
| | | 4. Tied notes | Expert models |
| | <i>Intervals</i> | 5. Interval sizes | Expert models |
| | | 6. Interval proportions | Expert models; Miller (2005) |
| <i>Melodic Esthetics</i> | <i>Melodic Structure</i> | 7. Key signature | Expert models |
| | <i>Rhythmic Structure</i> | 8. Time signature | Expert models |
| | | 9. Playability | Expert models |
| | <i>Note/Rest selection</i> | 10. Note placement | Perricone (2000) |
| | | 11. Tonic repetition | Laitz (2008); Perricone (2000) |
| | | 12. Opening note | Laitz (2008); Perricone (2000); Australian Music Examinations Board (2018b) |
| 13. Phrase endings | | Expert models; Goetschius (2009) | |
| 14. Peak note | | Australian Music Examinations Board (2018b) | |
| <i>Intervals</i> | | 15. Tritones | Expert models; Perricone (2000); Goetschius (2009); Aldwell and Cadwallader (2018); Schoenberg (1967) |
| | | 16. Augmented and diminished intervals | Expert models; Perricone (2000); Goetschius (2009); Aldwell and Cadwallader (2018); Schoenberg (1967) |
| <i>Melodic structure</i> | | 17. Closing intervals | Expert models; Laitz (2008) |
| | | 18. Interval resolutions | Goetschius (2009) |
| | | 19. Melodic direction | Goetschius (2009) |
| | 20. Contextualizing leaps | Goetschius (2009); Perricone (2000); Kwalwasser (1955); Schoenberg (1967) | |
| | 21. Peak placement | Australian Music Examinations Board (2018b) | |
| | – | – | |
| <i>Technical Appropriateness and Melodic Esthetics</i> | <i>Note/Rest selection</i> | 22. Placement of long notes | Expert models; Goetschius (2009) |
| | | 23. Placement of rests | Expert models; Goetschius (2009) |
| | | 24. Target key signature | Expert models; Miller (2005) |
| | <i>Intervals</i> | 25. Gap placement | Expert models; Laitz (2008); Miller (2005) |
| | | 26. Leaps | Laitz (2008) |
| | <i>Melodic structure</i> | 27. Repetition | Expert models; Australian Music Examinations Board (2018b); Miller (2005) |
| <i>Rhythmic structure</i> | 28. Length | Expert models; Goetschius (2009) | |
| | 29. Syncopation | Expert models; Miller (2005) | |

An exercise should only use intervals seen in expert models of the same difficulty level. For example, Grade 1 exercises for the flute are expected to only use intervals up to a size of 7.

- Interval proportions

At least 90% of the intervals should be between 0 and 3 in size, inclusive (i.e., between a unison and a fourth).

At least 50% of the intervals should be between notes only one scale degree apart (i.e., a second).

7 Key signature

An exercise should only be written in a key signature seen in expert models of the same difficulty level. For example, Grade 1 exercises for the flute are expected to only be written in the keys of C, F, G, and B \flat major, or A, D, and E minor.

8 Time signature

An exercise should only be written in a time signature seen in expert models of the same difficulty level. For example, Grade 1 exercises for the flute are expected to only be written in $\frac{4}{4}$, $\frac{2}{4}$, or $\frac{3}{4}$.

9 Playability

Each bar of the melody should only contain sequences of note lengths seen in expert models of the same difficulty level. For example, a bar in a Grade 1 exercise for the flute in $\frac{4}{4}$ time can contain two minims in a row, but would not be filled with a string of semiquavers.

10 Note placement

Strong notes from the target key (i.e., 1, 3, 5) should be placed on at least 50% of the strong beats in the melody.

11 Tonic repetition

At least 10% of the strong beats in the melody should be filled with a tonic note.

12 Opening note

The opening pitch of an exercise should be 1, 3, or 5.

13 Phrase endings

The note before a rest should be at least crotchet length.

14 Peak note

The highest note in the melody should be used no more than 3 times.

15 Tritones

Tritones should never be used.

16 Augmented and diminished intervals

Augmented and diminished intervals should never be used.

17 Closing interval

The melody should end with $2 \rightarrow 1$, $7 \rightarrow 1$, $4 \rightarrow 1$, or $5 \rightarrow 1$.

18 Interval resolutions

After a jump (i.e., an interval greater than a fourth), instability should always be resolved.

- should resolve to 3.
- should resolve to 1 or 3.
- should resolve to 5.
- should resolve to 1.

19 Melodic direction

If the melody is moving up in pitch, it should not change direction on pitch 7.

If the melody is moving down in pitch, it should not change direction on pitches 4 or 6.

20 Contextualizing leaps

The melody should switch direction after a leap (i.e., an interval greater than a fourth).

If there are two leaps in a row, the first should be larger.

21 Peak placement

The peak should fall within the middle 50% of the melody.

22 Placement of long notes

80% of notes longer than a crotchet should be placed on strong beats of the bar.

23 Placement of rests

80% of rests should be placed on weak beats of the bar.

24 Target key signature

All notes should have pitches from the target key signature.

25 Gap placement

There should be no more than 3 intervals of a third or more in sequence, unless the sequence forms an arpeggio.

26 Leaps

There should be no more than two intervals of a fourth or more in a row.

27 Repetition

A self-similar structure should not be repeated exactly more than twice in a row. If the structure is transposed when repeated, it is not considered to be repeated exactly.

28 Length

An exercise should be of a length, in bars, seen in expert models of the same difficulty level. For example, Grade 1 exercises for the flute are expected to only be 4, 8, 12, 14, or 16 bars long.

29 Syncopation

Grade 1 exercises for the flute should contain no syncopation. Grade 2 exercises for the flute can contain up to 10% syncopation.

3 RESULTS

3.1 Most Typical Characteristics

3.1.1 Overview

A preliminary examination of the generated exercises can be done by comparing the most typical values for a set of measured characteristics between the expert-written and algorithmically-generated exercises. These results, presented in **Sections 3.1.2 and 3.1.3**, show how closely the generated exercises were able to match the characteristics of the expert-written exercises. This is a good indication of the fitness for purpose of the results.

3.1.2 Grade 1

The Grade 1 generated exercises almost exactly match the most typical characteristics of the expert-written Grade 1 exercises, as seen in **Table 6**. Compared to the expert-written exercises, the

TABLE 6 | Typical characteristics of expert-written and generated Grade 1 sight reading exercises. Ratios and proportions are represented in terms of time. Differences between the expert-written and generated exercises are highlighted in bold. Characteristics marked with “*” are fixed and not expected to change.

| Characteristic | Typical Value(s) | |
|-------------------------|---|--|
| | Expert-written exercises | Generated exercises |
| Key signature* | F major, C major | F major, C major |
| Time signature* | $\frac{4}{4}$ | $\frac{4}{4}$ |
| Exercise length* | 8 bars | 8 bars |
| Range | 14 semitones (one octave and one tone) F4 → G5 | 12 semitones (one octave) F4 → F5 |
| Note lengths | 90–100% crotchets 0–10% quavers 0–5% minims, dotted minims, semiquavers | 90–100% crotchets 0–10% quavers 0–5% minims, dotted minims, semiquavers, semibreves |
| Rest lengths | 95–100% crotchets 0–5% quavers, semiquavers | 90 –100% crotchets 0– 10 % quavers, semiquavers |
| Ratio of notes to rests | 90% notes: 10% rests | 90% notes: 10% rests |
| Intervals | 95–100% gaps of 1 scale degree | 95–100% gaps of 1 scale degree |
| (As scale degrees) | 0–5% gaps of 0 or 2–7 scale degrees | 0–5% gaps of 0 or 2–7 scale degrees |

TABLE 7 | Typical characteristics of expert-written and generated Grade 2 sight reading exercises. Ratios and proportions are represented in terms of time. Differences between the expert-written and generated exercises are highlighted in bold. Characteristics marked with “*” are fixed and not expected to change.

| Characteristic | Typical Value(s) | |
|-------------------------|---|---|
| | Expert-written exercises | Generated exercises |
| Key signature* | G major, A minor, F major | G major, A minor, F major |
| Time signature* | $\frac{43}{44}$ | $\frac{43}{44}$ |
| Exercise length* | 8 bars | 8 bars |
| Range | 12 semitones (one octave and one tone) G4 → A5 | 12 semitones (one octave) G4 → G5 |
| Note lengths | 80–90% crotchets 0–10% quavers 0–10% minims 0–5% dotted minims, semiquavers, dotted Crotchets, dotted quavers, semibreves | 75 –90% crotchets 0–10% quavers 0–10% minims 0–5% dotted minims, semiquavers, dotted Crotchets, dotted quavers, semibreves |
| Rest lengths | 95–100% crotchets 0–5% quavers, semiquavers, minims | 85–100% crotchets 0– 15 % quavers, semiquavers, minims |
| Ratio of notes to rests | 95% notes: 5% rests | 95% notes: 5% rests |
| Intervals | 85–95% gaps of 1 scale degree | 85–95% gaps of 1 scale degree |
| (As scale degrees) | 0–10% gaps of 0 or 2 scale degrees 0–5% gaps of 2–7 scale degrees | 0–10% gaps of 0 or 2 scale degrees 0–5% gaps of 2–7 scale degrees |

generated exercises often exhibit a slightly smaller range, the most common highest note falling one full tone from G5 to F5. The generated exercises also include some semibreves, which were not seen in the expert-written exercises. They also exhibit slightly fewer crotchet rests and slightly increased proportions of quaver and semiquaver rests.

3.1.3 Grade 2

Table 7 shows that, compared to the expert-written exercises, the generated Grade 2 exercises exhibit only slight differences in their most typical characteristics. As with the Grade 1 exercises, the most common highest note drops a full tone, this time from A5 to G5. The typical proportions of crotchet note lengths increase in range from 80–90% to 75–90%. This is compensated for by increases in other note lengths, but not enough to alter the most typical proportions. The typical proportions of rest lengths also change. Crotchets are

used less frequently, a drop which is compensated for by an increased use of quavers, semiquavers, and minims.

3.2 Target Characteristics

3.2.1 Pitch Range

Compared to the expert-written exercises, the generated exercises exhibit a greater variety of ranges and an increased use of smaller ranges (i.e., ranges less than 12 semitones in size). However, overall, the pitch ranges of the expert-written and algorithmically-generated exercises are similar.

Although the range for each exercise was fixed with respect to a particular expert model, the algorithm does not enforce that the specified range be used to its limits, only that all selected pitches must fall within that range. It is for this reason that the range sizes and spreads differ between the expert-written and algorithmically-generated exercises.

3.2.2 Proportion of Notes vs. Rests

The amount of time in each exercise that should be filled by notes and rests is not specified exactly in the expert models, but can be inferred from the target proportions of specific note and rest lengths. That is, if the target proportion for crotchet notes is 0.5, the target proportion of quaver notes is 0.25, and target proportion of crotchet rests is 0.25, it can be inferred that the target proportion of notes is 0.75.

In the generated exercises there were no cases where an exercise contained rests if its corresponding expert model contained no rests. This is because, as discussed in **Section 2.4.2**, rests were not introduced at any stage of the algorithm's execution if there were no rests in the expert model used to derive the algorithm parameters. For cases where both note and rest target proportions were provided, the generated exercises regularly matched the given target proportions exactly.

3.2.3 Note Lengths

The generated exercises closely emulate the target note lengths extracted from the expert models. However, at both the Grade 1 and 2 difficulty levels there were note lengths in some generated exercises that were not present in any of the expert-written exercises. For Grade 1 the only unallowable note length used was a semibreve. This is not of significant concern given that semibreves are valid note lengths, and not unheard of at a Grade 1 level even though they are not present in the sample of expert-written Grade 1 exercises used in this work. The unallowable note lengths at the Grade 2 level, however, represent more of an issue. These were notes such as doubly dotted quavers and hemidemisiquavers, which are rarely if ever seen at even the highest difficulty levels. However, very few generated exercises contained such note lengths.

3.2.4 Rest Lengths

As with the note lengths, the proportions of rest lengths in the generated and expert-written exercises are reasonably close.

However, at both difficulty levels the generated exercises exhibited some rest lengths that were not present in the expert-written exercises. For example, some Grade 1 generated exercises contained minim rests, which were not in any of the expert-written Grade 1 exercises. Some exercises also contained rests with lengths that would rarely be seen at any difficulty level, such as doubly dotted semiquavers.

Overall, the proportions of rest lengths are more variable in the generated exercises compared to the expert-written exercises. This indicates that the generated exercises were not always able to match the target rest proportions exactly. They were, however, able to come close. Additionally, the spread of proportional values in the generated exercises is close to those of the expert-written exercises.

The generated exercises were not able to as closely match the target rest lengths as they were the target note lengths. It is important to note that this is most likely a side effect of the exercises containing significantly fewer rests than notes. A low number of rests within each exercise means that discrepancies between the actual and target rest proportions are amplified simply because each individual rest represents a relatively large proportion of the overall rest time. This results in cases where a single rest length being of an "unallowable" length can have a large effect on the overall rest proportions. For

TABLE 8 | Summary of fitness values for each grade level of generated exercises. Shows that at least one exercise reached the maximum value for each fitness measure (i.e., 1.0), and that the average fitness values for each measure were high at every difficulty level.

| Fitness measure | Grade | Minimum | Maximum | Average | SD |
|---------------------|-------|---------|---------|---------|------|
| Target note lengths | 1 | 0.53 | 1.0 | 0.95 | 0.11 |
| | 2 | 0.66 | 1.0 | 0.96 | 0.08 |
| Target rest lengths | 1 | 0.96 | 1.0 | 0.95 | 0.01 |
| | 2 | 0.94 | 1.0 | 0.99 | 0.01 |
| Allowable lengths | 1 | 0.65 | 1.0 | 0.99 | 0.04 |
| | 2 | 0.95 | 1.0 | 0.99 | 0.01 |
| Target intervals | 1 | 0.82 | 1.0 | 0.95 | 0.05 |
| | 2 | 0.92 | 1.0 | 0.97 | 0.03 |
| Allowable intervals | 1 | 0.94 | 1.0 | 0.99 | 0.01 |
| | 2 | 0.96 | 1.0 | 0.99 | 0.01 |
| Melody shape | 1 | 0.8 | 1.0 | 0.99 | 0.02 |
| | 2 | 0.88 | 1.0 | 0.99 | 0.01 |

^a<http://web.mit.edu/music21/>.

example, if an exercise was given a target of containing 4 crotchet rests, having one of those rests generated as two quaver rests means that 25% of the rest time is filled by an "unallowable" length, and the target rest proportion was only 75% met. This situation is much less likely to happen with note proportions, simply because significantly more time within each exercise is filled by notes.

3.2.5 Intervals

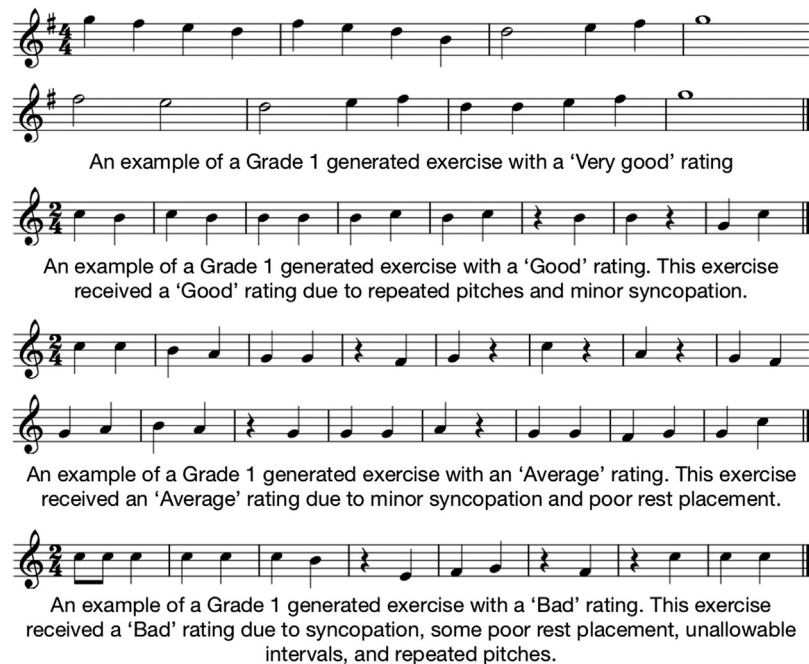
At each of the difficulty levels, the proportions of intervals exhibited in the generated exercises closely match the target proportions set by the expert models.

In both grade levels the use of intervals with a size of 0 is higher in the generated exercises than in the expert-written exercises. However, as this increase in use is only slight it is not overly concerning. Compared to the expert-written exercises, the generated exercises exhibit greater variation in interval proportions. As with the rest length proportions, this is most likely an indication that the generated exercises were not always able to exactly match the target proportions. They were, however, able to come close.

3.3 Fitness of the Generated Exercises

As shown in **Table 8**, the generated exercises consistently achieved high fitness values on every fitness measure. At each grade level there was at least one exercise that achieved a perfect score on each of the fitness measures. The average fitness value for each measure was consistently high, ranging between 0.95 and 0.99 inclusive. Similarly, the standard deviations of the fitness values were consistently low, indicating that little variance was exhibited across the fitness values for each measure. The minimum values are more variable, both across fitness measures and within the same fitness measure across different grade levels. However, given the high average fitness and low standard deviations, such minimum values represent outliers rather than trends.

Overall, there are no fitness measures on which the generated exercises scored consistently better or worse. This is true both when comparing different fitness measures within a grade level, and when comparing fitness values on the same measures across different grade levels.



An example of a Grade 1 generated exercise with a 'Very good' rating

An example of a Grade 1 generated exercise with a 'Good' rating. This exercise received a 'Good' rating due to repeated pitches and minor syncopation.

An example of a Grade 1 generated exercise with an 'Average' rating. This exercise received an 'Average' rating due to minor syncopation and poor rest placement.

An example of a Grade 1 generated exercise with a 'Bad' rating. This exercise received a 'Bad' rating due to syncopation, some poor rest placement, unallowable intervals, and repeated pitches.

FIGURE 6 | Examples of Grade 1 generated exercises assigned each of the Likert quality ratings. An example of an exercise rated as "Very bad" is not provided as none of the Grade 1 exercises were assigned this rating.

Given the application domain, it is likely that some of the fitness measures may have conflicting goals. That is, an increase in one fitness value may be directly related to a decrease in another. For example, consider an exercise which meets its target proportion of notes and rests, but contains two unallowable rest lengths (e.g., two quaver rests instead of a single crotchet rest). If one or both of those unallowable rests were changed to notes, the "Allowable lengths" fitness of that exercise would improve. However, the extra notes would mean that the exercise no longer meets the target note and rest proportions, thus decreasing its fitness in the "Target note lengths" and "Target rest lengths" measures. The set of results presented in this work indicate that the fitness measures do not influence one another, given that the generated exercises achieved consistently high fitness scores. As such, potentially conflicting goals among the fitness measures can be considered not to be an issue.

3.4 Fitness for Purpose of the Generated Exercises

3.4.1 Grade 1

The majority of the Grade 1 generated exercises are "fit for purpose", with over 60% being assigned a "Very good", "Good", or "Average" rating. This proportion increases to almost 80% when small repairs are made to some exercises.

Figure 6 provides examples of generated Grade 1 exercises assigned each of the Likert ratings. None of the exercises at this difficulty level were assigned a "Very bad" rating, so no example has been given. Reasons are provided for each example's rating.

3.4.2 Grade 2

Approximately half of the Grade 2 generated exercises are fit for purpose. This is roughly 15% less than the Grade 1 generated exercises. The proportion of Grade 2 exercises rated as "fit for purpose" increases once small repairs are made, reaching approximately the same percentage as the Grade 1 generated exercises before repairs (i.e., roughly 60%). This drop in the number of "fit for purpose" exercises is an expected result due to the increased complexity of the Grade 2 exercises compared to Grade 1.

The major difference between the Grade 1 and 2 ratings is the presence of "Very bad" exercises. These account for around 10% of the Grade 2 generated exercises, a proportion which does not change after repairs are made. This indicates that the "Very bad" exercises can not be upgraded in fitness for purpose, or even improved to a "Bad" rating. Given that the criteria for a "Very bad" rating is unplayable elements within an exercise, it is not surprising that a small number of changes could not resolve these issues. For reference, examples of generated Grade 2 exercises assigned each Likert rating are provided in Figure 7.

4 DISCUSSION AND FUTURE WORK

4.1 Current Capabilities of the Evolutionary Algorithm

These results show that the proposed EA is capable of emulating the characteristics of expert-written sight reading exercises at the Grade 1 and 2 difficulty levels, and that it is able to do so in a way



An example of a Grade 2 generated exercise with a 'Very good' rating



An example of a Grade 2 generated exercise with a 'Good' rating. This exercise received a 'Good' rating due to repeated pitches.



An example of a Grade 2 generated exercise with an 'Average' rating. This exercise received an 'Average' rating due to poor placement of larger intervals.



An example of a Grade 2 generated exercise with a 'Bad' rating. This exercise received a 'Bad' rating due to excessively long tied notes.



An example of a Grade 2 generated exercise with a 'Very bad' rating. This exercise received a 'Very bad' rating due to the unplayable sequence in bar 6.

FIGURE 7 | Examples of Grade 2 generated exercises assigned each of the Likert quality ratings. Note that the "Very bad" example was given this rating due to the rhythm in bar 6 being uncharacteristically difficult for the Grade 2 difficulty level.

that is generally fit for purpose. This is a particularly promising outcome given the relatively simplistic and general nature of the fitness measures. The capabilities of the algorithm to produce appropriate sight reading practice material is additionally supported by the Likert quality ratings, which show that the generated exercises also conform to the expectations of musical esthetics and technical appropriateness defined by the field of music theory.

Although the results indicate the potential of the algorithm, there are areas in which the algorithm's capabilities can be improved. When examining the Likert quality ratings of an exercise with respect to its characteristics there appear to be no links. That is, the quality of an algorithmically generated exercise is not related to its key signature, time signature, length, note proportions, or interval proportions. This indicates that the primary difficulty in applying the algorithm to generate exercises at higher difficulty levels will be in managing the overall increase in musical and technical complexity. Even at the difficulty levels currently examined (i.e., Grades 1 and 2), higher quality results

should be possible were this complexity to be better modeled and incorporated into the evolutionary process.

For example, the presence of rests in an exercise affects its overall structure. If not placed carefully within a sequence of notes, rests can cause unintended syncopation or awkward breaks in phrasing. The existence of a rest also affects the measurement of intervals within an exercise, as two notes separated by a rest are not considered to be part of an interval during fitness calculations. Given that rests become more frequent in number and length at later difficulties, these issues will become more prominent.

The pitch range of exercises also grows with the difficulty level. For example, exercises at the Grade 2 level cover a greater range of pitches than exercises at the Grade 1 level. An increase in pitch range increases the solution space. This is because there are simply more potential pitches to select, thus more potential for an algorithm to select pitch sequences which are aesthetically or technically inappropriate. Naturally, this increase in the size of the solution space also causes an increase in the difficulty of algorithmically generating fit for

purpose solutions. This problem is compounded when considering other musical artifacts, as the size of the solution space similarly increases with additions to the sets of allowable note lengths and intervals. The interaction of these elements also needs to be considered. For example, increasing the allowable intervals in an exercise where only crotchet note lengths are allowed would increase the overall solution space. If additional note lengths were also to be allowed, the solution space would increase exponentially, not linearly. This is because some interval sequences, which would have been appropriate between crotchets, would not be appropriate with shorter note lengths.

These issues were expected, particularly given the general approach taken to measuring fitness in this work. Further efforts in modeling the requirements of higher difficulty exercises and incorporating those models into the algorithm would help to manage the expanding size of the solution space, thus the algorithm's capacity to generate more complex exercises.

One potential drawback of the Likert ratings currently presented in the results is that they were all measured using a single rater. To check for consistency more robustly it would be ideal to have the same exercises rated using the same framework by multiple experts.

4.2 Future Directions for Algorithmic Development

When developing an algorithm for generating musical sight reading exercises, there are trade-offs to be made. One key decision is whether the algorithm will be specific or general. For example, a specific algorithm might only generate Grade 2 exercises focusing on breath control for the clarinet. Alternatively, a general algorithm might aim to generate exercises for any wind instrument at any difficulty level. There are benefits and drawbacks to each approach. The more specific the target, the more focused the algorithm can be. This means more domain knowledge can be incorporated and more restrictive parameters can be set. It is likely that a specific approach would enable the quality of output to be improved. However, a specific approach would, by definition, also be limited in its utility. A general approach would need to consider many more factors. For example, for an algorithm to target multiple instruments it would need to model the differences between the technical requirements for those instruments. A more general algorithm is likely to have an increased utility. However, it also takes on the risk of attempting to cover too much scope, which would limit its ability to generate quality output.

The approach taken in this work is somewhere in between. It is not so general as to target many difficulty levels, but it also isn't restricted to a single type of exercise. Although the application of the algorithm presented in this work was generating musical sight reading exercises for the flute, its parameters are purely data driven—they are not instrument-specific. Instead, appropriate values can be extracted from models of expert-written examples, which may relate to any monophonic instrument. This is discussed further in **Section 4.4**.

Some avenues for future development can be found in the ruleset used to evaluate the fitness for purpose of solutions. For

example, the rule for 'Note placement' states that *Strong notes from the target key (i.e., I, III, V) should be placed on at least 50% of the strong beats in the melody*. This indicates that music sounds better when strong notes from the target key are placed on strong beats of the bar. Such note placements could be encouraged by the evolutionary algorithm. Doing so might reduce or even remove the need for this evaluation rule, but should also increase the esthetics of results by reinforcing the key signature.

A similar approach could be taken to reinforce the time signature of an exercise. This relates to the "Placement of long notes" rule, which states that *80% of notes longer than a crotchet should be placed on strong beats of the bar*, and the "Placement of rests" rule, which states that *80% of rests should be placed on weak beats of the bar*. By encouraging these optimal note and rest placements the music should 'feel' like it is written in the target time signature. This would increase the overall esthetics of the generated melodies and avoid some situations where phrases seem to end abruptly.

As an addition to modeling expert-written examples, the algorithm could incorporate alternative models of musical complexity. These models would be relative to a specific instrument. One possible model is the *musiplectics* system (Holder et al., 2015). In this work, Holder et al. (2015) defines a method for computationally measuring the complexity of a musical score for any instrument. Measuring the complexity of a score first requires the definition of several parameters for the chosen instrument. Currently, only the parameters for a B♭ clarinet are provided. Implementing support for more instruments represents a non-trivial quantity of work, most of which requires input from an expert in the instrument in question. However, doing so might result in a valuable addition to the capabilities of the evolutionary algorithm.

Another area of improvement for the algorithm would be to use chord progressions. Currently, the algorithm generates exercises within a particular key signature. It does not, however, create melodies which follow chord progressions. For example, a common 4 bar chord progression is *I, IV, V, I*. If the exercise was in C major, this would mean that the 4 bars would be rooted in C major, F major, G major, and C major, in that order. Chord progressions give a melody a sense of movement and interest. Although not considered in this work, the expert-written exercises do use chord progressions. As such, it would make sense for the algorithm to do so as well.

One way chord progressions could be implemented would be to use the progression map proposed by Stephen (2017). This map defines transitions between chords which will sound aesthetically pleasing. This implementation would not require any changes to be made to the tree structure used to represent melodies. Although each bar would be rooted in a different key, the melody overall would still be in the one key signature. That is, even if the second bar in a C major melody might be written in F major, it would still only use pitches from the C major scale. The ruleset for evaluating exercises, however, would need to be updated. This is because some of the rules explicitly reference the target key. For example, the "Note placement" rule states that *Strong notes from the target key (i.e., I, III, V) should be placed on at least 50% of the strong beats in the melody*. If the generated melodies were to follow chord progressions, the "target key" part of this rule would need to be

interpreted as referring to the chord of the bar, not of the overall melody.

Some of the expert-written exercises also change key completely as they progress, or contain accidentals outside of the key signature. Neither of these features is currently supported by the algorithm. Allowing additional accidentals is trivial, as it would simply require removing the restriction within the algorithm preventing it from selecting pitches outside the target key. However, this implementation would introduce significant complexity to the system, as the potential for selecting poor sounding pitch sequences would dramatically increase. A better implementation would allow non-key pitches to be selected, but restrict when that could occur.

Allowing for complete key changes is a more difficult task. Currently, the melody tree structure does not record the key signature of the melody it describes. As such, it also does not support the ability to record a change in key signature. This is not necessarily an issue, as information relating to the key signatures can be recorded elsewhere. The true difficulty lies in determining when a key change is appropriate, what the new key should be, and how to smoothly transition between the old and new keys. Such functionality would require significant changes to the algorithm. It would also require the key change to be noted in the output so that the melody can be interpreted correctly.

Another feature of the expert-written exercises not currently shown in the generated exercises is anacrusis. An anacrusis is where a single bar is split into two parts which are placed at the beginning and end of a melody. This type of structure is not supported by the melody tree or the algorithm, and adding support would require significant work.

4.3 Building Better Models of Expert Knowledge

Given that the algorithm is designed to emulate expert models of musical sight reading exercises, it stands to reason that developing better expert models would improve the quality of its output. Currently, the expert models are a combination of simple characteristics (i.e., key signature, time signature) and statistical measures (i.e., note/rest/interval proportions). As such, there is significant scope for further development in this area.

One area for potential development relates to the analysis of co-occurring features. Currently, the note, rest, and interval proportions are treated separately. However, it is possible that there are some dependencies between these features that are not currently being captured. For example, it might be that larger intervals are more likely to be placed on longer notes, and smaller intervals on shorter notes. Finding these types of co-occurrences should be a relatively simple task. More difficult would be determining which co-occurring features are important to emulate, and how they should be implemented.

A similar area is that of sequence or pattern identification. The expert-written exercises, and music in general, exhibit many clear patterns. For example, often a dotted quaver will be followed by a semiquaver, or four semiquavers will be used in sequence. These types of patterns generally serve to reinforce the beats within the music and create a sense of rhythmic stability. Identifying the use of these patterns within the expert-written exercises, and incorporating them into the algorithm would serve to both

better emulate the characteristics of the expert-written examples, and create more generally aesthetically pleasing results.

A complex area that has not yet been addressed either through the analysis of expert-written exercises or in the algorithm development is that many musical sight reading exercises are targeted to developing a specific skill. For example, some exercises for the flute contain a large proportion of long notes to encourage the development of breath control. Others might specifically use a series of arpeggios to reinforce scale structures.

Incorporating this type of information into the algorithm would be a significant undertaking. It would require extensive expert knowledge to identify the purposes of different musical sight reading exercises and describe how they have been written to address these purposes. Developing the ability to algorithmically generate similarly targeted exercises would be as, if not more, difficult. However, doing so would greatly increase the utility of the generated exercises, as they would be able to more specifically target the needs of different users.

4.4 Applying the Algorithm to Other Instruments

Although the use of the algorithm presented in this work was to generate musical sight reading exercises for the flute, the parameters are purely data driven. That is, they are instrument-agnostic. Given this, the algorithm can be applied to generate exercises for any monophonic instrument. This would involve curating a set of expert-written examples, and using those examples to determine appropriate parameters. Additional work would be required to support polyphonic instruments, particularly in the development of the tree structure used to represent melodies.

To validate the algorithm's abilities in generating exercises for other instruments, the evaluation ruleset would need to be revisited. While the rules themselves are instrument-agnostic, their exact interpretation is sometimes relative to the analysis of expert-written examples. If the algorithm were applied to another instrument, those rules would need to be revised with respect to a new set of expert-written examples.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

CP contributed the larger idea of the study, wrote the code, performed the bulk of analysis, and drafted the paper. TH, AB, and CJW revised the work critically for important intellectual content at all stages, and guided the overall direction of the work.

ACKNOWLEDGMENTS

This work was supported by an Australian Government Research Training Program (RTP) Scholarship.

REFERENCES

- Acevedo, A. (2004). "Fugue composition with counterpoint melody generation using genetic algorithms," in *International symposium on computer music modeling and retrieval*. Berlin, Heidelberg: Springer, 96–106.
- Aldwell, E., and Cadwallader, A. (2018). *Harmony and voice leading*. Boston, MA: Cengage Learning, 736.
- Associated Board of the Royal Schools of Music (1995). *Flute specimen sight reading*. London, United Kingdom: Associated Board of the Royal Schools of Music, 24.
- Australian Music Examinations Board (2018b). [Dataset] AMEB ask andrew: introduction to melody writing. Available at: <https://www.youtube.com/watch?v=dHsSD2GqDQM> (Accessed May 13, 2013).
- Australian Music Examinations Board (2018a). *2018 Manual of syllabuses*. Melbourne, Australia: Australian Music Examinations Board.
- Banton, L. (1995). The role of visual and auditory feedback during the sight-reading of music. *Psychol. Music* 23, 3–16. doi:10.1177/0305735695231001
- Biles, J. (2007). "Evolutionary computation for musical tasks," in *Evolutionary computer music*. London, United Kingdom: Springer, 28–51.
- Crozier, R. (2000). *The music teacher's companion: a practical guide*. London, United Kingdom: Associated Board of the Royal Schools of Music, 144.
- Dahlstedt, P. (2007). "Autonomous evolution of complete piano pieces and performances," in: Proceedings of musicAL workshop, in: advances in artificial life, 9th european conference, ECAL 2007, Lisbon, Portugal, September 10–14, 2007, (Springer), 4648.
- de León, P., Inesta, J., Calvo-Zaragoza, J., and Rizo, D. (2016). Data-based melody generation through multi-objective evolutionary computation. *J. Math. Music* 10, 173–192. doi:10.1080/17459737.2016.1188171
- Fonseca, C., and Fleming, P. (1995). "Multiobjective genetic algorithms made easy: selection sharing and mating restriction," in First international conference on genetic algorithms in engineering systems: innovations and applications, Sheffield, UK, September 12–14, 1995 (Sheffield, United Kingdom: IET).
- Galyen, S. D. (2005). Sight-reading ability in wind and percussion students: a review of recent literature. *Update Appl. Res. Music Educ.* 24, 57–70. doi:10.1177/87551233050240010107
- Goetschius, P. (2009). *Exercises in melody-writing*. Charleston, South Carolina: BiblioBazaar, LLC, 126.
- Gregory, T. B. (1972). The effect of rhythmic notation variables on sight-reading errors. *J. Res. Music Educ.* 20, 462–468. doi:10.2307/3343804
- Harris, P. (1994). *Improve your sight-reading!: flute*. London, United Kingdom: Faber Music, 32.
- Holder, E., Tilevich, E., and Gillick, A. (2015). "Musiplectics: computational assessment of the complexity of music scores," in 2015 ACM international Symposium on new ideas, new paradigms, and Reflections on Programming and software (onward!), October 2015 (New York, NY: ACM), 107–120.
- Horn, J., Nafpliotis, N., and Goldberg, D. (1994). "A niched pareto genetic algorithm for multiobjective optimization," in Proceedings of the first IEEE conference on evolutionary computation, Nagoya, Japan, May 20–22, 1996, (Nagoya, Japan: IEEE) vol. 1, 82–87.
- Ji, W. (2017). Effectiveness of targeted feedback toward rhythm sightreading in university/college-level music education contexts. Ph.D. thesis, Education: Faculty of Education.
- Johnson, M., Tauritz, D., and Wilkerson, R. (2004). "Evolutionary computation applied to melody generation," in Proceedings of the ANNIE, St. Louis, MO, November, 2004 (St. Louis, MO: American Society of Mechanical Engineers), Vol. 7, 73–78.
- Kopiez, R., and In Lee, J. (2008). Toward a general model of skills involved in sight reading music. *Music Edu. Res.* 10, 41–62. doi:10.1080/14613800701871363
- Kornicke, L. (1992). *An exploratory study of individual difference variables in piano sight-reading achievement*. Washington, D.C., United States: UMI/Bell & Howell, 574.
- Koza, J. R. (2018). [Dataset] Genetic programming. Available at: <http://geneticprogramming.com/Tutorial/>.
- Kwalwasser, J. (1955). *Exploring the musical mind*. New York, NY: Coleman-Ross Co., 189.
- Laitz, S. G. (2008). *The complete musician: an integrated approach to tonal theory, analysis, and listening*. New York, NY: Oxford University Press, Vol. 1, 960.
- Lehmann, A. C., and McArthur, V. (2002). Sight-reading. *Sci. Psychol. Music Perform.* 135–150. doi:10.1093/acprof:oso/9780195138108.001.0001
- McPherson, G., and Gabrielson, A. (2002). From sound to sign. *Science. Psychol. Music Perform.* 99–116. doi:10.1093/acprof:oso/9780195138108.003.0007
- Miller, M. (2005). *The complete idiot's guide to music theory*. London, United Kingdom: Penguin, 314.
- Perricone, J. (2000). *Melody in songwriting: tools and techniques for writing hit songs*. Wisconsin, United States: Hal Leonard Corporation, 192.
- Rae, J. (2007). *Sound at sight—sight reading pieces for flute*. Trinity College London, United Kingdom: Book, 1.
- Rizo, D., Inesta, J., and Moreno-Seco, F. (2003). "Tree-structured representation of musical information," in Iberian conference on pattern recognition and image analysis, Madrid, Spain, July 1–4, 2019 (Madrid, Spain: Springer), 838–846.
- Schoenberg, A. (1967). *Fundamentals of music composition*. New York City, NY: St. Martin's Press, 224.
- Schulz, D. (2016). Pianote: a sight-reading program that algorithmically generates music based on human performance. MS dissertation. California (US): University of California Polytechnic.
- Selleck, J. (2012). *Flute sight-reading (series 3)*. Melbourne, Australia: Australian Music Examinations Board (AMEB).
- Spillman, R. (1990). *Sightreading at the keyboard*. New York City, NY: Schirmer Books, 304.
- Stephen, M. (2017). [Dataset] A progression map for major keys. Available at: <http://mugglinworks.com/chordmaps/chartmaps.htm>.
- The Associated Board of the Royal Schools of Music (2018). *ABRSM syllabus*. London, United Kingdom: The Associated Board of the Royal Schools of Music.
- Tsangari, V. (2010). An interactive software program to develop pianists' sight-reading. MS dissertation. Iowa (US): University of Iowa.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2021 Pierce, Hendtlass, Bartel and Woodward. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



Listener Modeling and Context-Aware Music Recommendation Based on Country Archetypes

Markus Schedl^{1,2*}, Christine Bauer³, Wolfgang Reisinger¹, Dominik Kowald⁴ and Elisabeth Lex⁵

¹Johannes Kepler University Linz, Linz, Austria, ²Linz Institute of Technology, Linz, Austria, ³Utrecht University, Utrecht, Netherlands, ⁴Know-Center GmbH, Graz, Austria, ⁵Graz University of Technology, Graz, Austria

OPEN ACCESS

Edited by:

Roger B. Dannenberg,
Carnegie Mellon University,
United States

Reviewed by:

Anders Oland,
Carnegie Mellon University,
United States
Alexandra Bonnici,
University of Malta, Malta

*Correspondence:

Markus Schedl
markus.schedl@jku.at

Specialty section:

This article was submitted to
Machine Learning and
Artificial Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 30 October 2019

Accepted: 30 November 2020

Published: 02 February 2021

Citation:

Schedl M, Bauer C, Reisinger W,
Kowald D and Lex E (2021) Listener
Modeling and Context-Aware Music
Recommendation Based on
Country Archetypes.
Front. Artif. Intell. 3:508725.
doi: 10.3389/frai.2020.508725

Music preferences are strongly shaped by the cultural and socio-economic background of the listener, which is reflected, to a considerable extent, in country-specific music listening profiles. Previous work has already identified several country-specific differences in the popularity distribution of music artists listened to. In particular, what constitutes the “music mainstream” strongly varies between countries. To complement and extend these results, the article at hand delivers the following major contributions: First, using state-of-the-art unsupervised learning techniques, we identify and thoroughly investigate (1) country profiles of music preferences on the fine-grained level of music tracks (in contrast to earlier work that relied on music preferences on the artist level) and (2) country archetypes that subsume countries sharing similar patterns of listening preferences. Second, we formulate four user models that leverage the user’s country information on music preferences. Among others, we propose a user modeling approach to describe a music listener as a vector of similarities over the identified country clusters or archetypes. Third, we propose a context-aware music recommendation system that leverages implicit user feedback, where context is defined via the four user models. More precisely, it is a multi-layer generative model based on a variational autoencoder, in which contextual features can influence recommendations through a gating mechanism. Fourth, we thoroughly evaluate the proposed recommendation system and user models on a real-world corpus of more than one billion listening records of users around the world (out of which we use 369 million in our experiments) and show its merits vis-à-vis state-of-the-art algorithms that do not exploit this type of context information.

Keywords: music, recommender system, culture, country, clustering, context, user modeling, music preferences

1 INTRODUCTION

Recommendation systems (or recommender systems) have become an important means to help users find and discover various types of content and goods, including movies, videos, books, and food (Ricci et al., 2015). As such, they represent substantial business value. In the music industry, recommender systems—powered by machine learning and artificial intelligence—have radically changed the market; they have even become major drivers in this industry. Essentially, music recommender systems (MRS) shape today’s digital music distribution (Schedl et al., 2015) and have become vital tools for marketing music to a targeted audience, as evidenced by the success of

recommender-systems-featuring music streaming services such as Spotify, Deezer, or Apple Music. While MRS operate in a multi-stakeholder environment including platform providers, artists, record companies, and music consumers/listeners (Bauer and Zangerle, 2019), it is most commonly the music consumers/listeners, who are considered the users of an MRS. In the paper at hand, we also take this perspective.

Traditionally, content-based filtering and collaborative filtering (CF)—or hybrid combinations thereof—have been the most common algorithms to create recommender systems (Ricci et al., 2015). The former assumes that users will like items similar to the ones they liked in the past, and therefore selects items to recommend according to some notion or metric of similarity in terms of item content (e.g., music style, timbre, or rhythm) between the user's liked items and unseen items from the catalog. In contrast, CF assumes that a user will prefer items that are liked by other users with similar preferences. In this case, items to recommend are, for instance, found by comparing the target user's consumption or rating profile to that of the other users, identifying the most similar other users, and recommending what they liked (user-based CF). Alternatively, users and items can be directly matched via similarities computed in a joint low-dimensional representation of users and items (i.e., model-based CF).

Enhancing the classical approaches CF and content-based filtering, in recent years, researchers started to leverage additional information—beyond users, items, and their interactions—to improve recommendations. Recommender systems that consider user characteristics or information describing a situation are typically referred to as *context-aware recommendation systems* (Adomavicius and Tuzhilin, 2015). Next to considering time and location as contextual side information, taking information derived from the user's country into account has been demonstrated to improve recommendation quality; for instance, cultural and socio-economic characteristics of the user's country (Zangerle et al., 2018), or the user taste's proximity to their country-specific music mainstream ("mainstreaminess") (Bauer and Schedl, 2019).

Against this background, we approach the task of context-aware music recommendation based on country information; in contrast to most previous works, we consider user country in our approach without using any external information about the country, such as cultural, economic, or societal information. The reason is that respective data sources about countries (e.g., Hofstede's cultural dimensions,¹ the Quality of Government measures,² or the World Happiness Report³) provide information on the country level, which may not necessarily reflect the circumstances of individual users and, thus, can introduce problems in the recommendation process. For instance, cultural values or income may be very unequally distributed among a country's population.

To avoid this, instead of using external information derived from the user's country, we leverage purely the self-reported country information of the users as available in the system, and investigate how behavioral data about music listening can be used to (1) identify archetypal country clusters based on track listening preferences, (2) how users can be modeled using the results of (1), and (3) how the resulting user models can be integrated into a state-of-the-art deep learning-based music recommendation algorithm.

As in many other domains, nowadays, deep neural network architectures dominate research in music recommendation systems, due to their ability to automatically learn features from low-level audio signals and their superior performance (Schedl, 2019). This article is no exception. We propose a multi-layer generative model in which contextual features can influence recommendations through a gating mechanism.

In this context, we formulate the following research questions:

- RQ1:** To what extent can we identify and interpret groups of countries that constitute *music preference archetypes*, from behavioral traces of users' music listening records?
- RQ2:** Which are effective ways to model the users' geographic background as a contextual factor for music recommendation?
- RQ3:** How can we extend a state-of-the-art recommendation algorithm, based on variational autoencoders, to include user context information, in particular, the geo-aware user models developed to answer RQ2?

In the remainder of this article, we first explain the conceptual foundation of our work and discuss it in the context of related research (**Section 2**). Subsequently, we detail the methods we adopt to investigate the research questions; in particular, we specify the approaches used for data preparation, clustering, user modeling, and track recommendation (**Section 3**). The results of our experiments on uncovering geographic music listening archetypes and on music track recommendation, altogether with a detailed discussion thereof, are presented in **Section 4**. Finally, **Section 5** concludes the article with a brief summary of the major findings, a discussion of limitations, and pointers to future work.

2 CONCEPTUAL BACKGROUND AND RELATED WORK

A multitude of factors have been found to influence an individual's music preferences. There is a long history of research investigating the relationships between music preferences and, for instance, demographics (Colley, 2008; Bonneville-Roussy et al., 2013; Cheng et al., 2017), personality traits (Rentfrow and Gosling, 2003; Schäfer and Mehlhorn, 2017), and social influences (ter Bogt et al., 2011; Bonneville-Roussy and Rust, 2018).

In the middle of the nineteenth century emerged a cultural hierarchy in America (DiMaggio, 1982; Levine, 1988) where a high social status patronized the fine arts (referred to as "highbrow") while all other forms of popular culture were associated with a lower status (referred to as "middlebrow" or

¹<https://geerthofstede.com/research-and-vsm/dimension-data-matrix>

²<https://qog.pol.gu.se>

³<https://worldhappiness.report>

“lowbrow”). In the 1990s, a series of studies (Peterson and Simkus, 1992; Peterson and Kern, 1996) have defended the view that, for the elite, highbrow was being replaced by a consumption pattern termed “omnivorousness”. Cultural omnivorousness reflects that people’s taste includes both elite and popular genres. This was subsequently shown to hold for various countries (e.g., Holbrook et al., 2002; Coulangeon, 2003; Fisher and Preece, 2003). Also, the consumption practices of low status taste were reconceptualized: The earlier view that the lowbrow group would be willing to consume any entertainment on offer (Horkheimer and Adorno, 1972) was replaced by the finding that low status people tend to choose one form of entertainment and avoid others (Bryson, 1997). Thus, overall the view evolved from highbrow–lowbrow to omnivore–univore. Analyzing music consumption across eight European countries, Coulangeon and Roharik (2005) supported the “omnivore–univore” scheme rather than the former “highbrow–lowbrow” model. The omnivorous cultural taste was later found unstable over time (Rossman and Peterson, 2015), though. Katz-Gerro (2002) has shown that the dividing line of class distinctions varies across countries and also the genre associations to social classes deviate. She concludes that, while class matters, the main determinants of cultural preferences relate to gender, education, and age (Katz-Gerro, 1999). Coulangeon (2005) questions the earlier view on the reasons for the different tastes of higher- and lower-status classes: He challenges that it would be the upper class’ familiarity with the so-called “legitimate” culture and the little accessibility to that culture for the lower-status classes, that distinguished what the upper class and lower-status classes prefer. Instead, he attributes it to the diversity of the stated preferences of people of the upper class, whereas the preferences of members of lower-status classes appear more exclusive. Later work, studying music taste in the “modern age” (Nuccio et al., 2018), found little evidence that musical taste is indeed aligned with class position.

Although there is a multitude of factors that influence an individual’s music preferences that lead to a diversity of music created and listened to, there are (market) structures and other mechanisms that effect certain tendencies in what music is preferred within a particular community. For instance, the music recording industry is typically considered a globally oriented market (Dolata, 2013). Yet, studies have revealed the existence of national boundaries (Bauer and Schedl, 2018). There are various country-specific mechanisms that affect an individual’s music preferences and consumption behavior: Preferences are culturally shaped (Baek, 2015; Budzinski and Pannicke, 2017); music perceptions vary across cultures, for instance, with respect to mood (Morrison and Demorest, 2009; Stevens, 2012; Lee and Hu, 2014; Singhi and Brown, 2014); and countries have substantially different national market structures with respect to, for instance, available music repertoire due to copyright and licensing, advertising campaigns, local radio airplay, or quotas for national artists (Hracs et al., 2017; Gomez Herrera and Martens, 2018).

Knowledge about country-specific differences in music preferences can be explicitly used to improve music recommender systems, for instance, by leveraging information about the users’ geographic or cultural background. For instance,

Vigliensoni and Fujinaga (2016) use a factorization machine approach for matrix factorization and singular value decomposition to integrate—amongst others—a user’s country as context information. Bauer and Schedl (2019) use a contextual pre-filtering approach (Adomavicius and Tuzhilin, 2015), where the user base is first segmented by user country, and a target person is then compared to other people from the very same country (in contrast to a comparison with the entire user base). Sánchez-Moreno et al. (2016) use a k-nearest neighbor (k-NN) approach integrating, among others, the user’s country as attribute. Zangerle et al. (2018) leverage further country-specific data sources; for each country, they use the respective scores on the cultural dimensions by Hofstede et al. (2005) as well as the scores of the World Happiness Report (Helliwell et al., 2016) to tailor recommendations to the individual.

The work at hand differentiates from related work in several aspects.

- First, although music preferences vary across countries, several studies (e.g., Moore et al., 2014; Pichl et al., 2017; Schedl et al., 2017; Bauer and Schedl, 2019) have shown similarities in music preferences between countries, typically identified with clustering approaches. Yet, to the best of our knowledge, the work at hand is the first one to integrate information on country similarities into the music recommendation approach.
- Second, while other work, most notably, Zangerle et al. (2018), reaches out to include external data about countries (such as economic factors, happiness index, cultural dimensions), the approach at hand remains independent from any external data sources, enabling platform providers to build a self-sustaining recommendation system. Such a system can rely exclusively on data that is contained in the provider’s platform, including users’ self-disclosed country information.
- Third, most existing research on music preferences and recommender systems considers music preferences on a genre level (e.g., Skowron et al., 2017; Adiyansjah et al., 2019) or artist level (e.g., Sánchez-Moreno et al., 2016; Bauer and Schedl, 2019). Research on country-aware music recommendation systems that provide recommendations on the track level is rare (e.g., Zangerle et al., 2018). However, the genre and the artist level may be too coarse-grained to reflect users’ music preferences, for several reasons. Music genres are vaguely defined (Beer, 2013; Sonnett, 2016; Vlegels and Lievens, 2017) and users’ perceptions thereof differ tremendously (van Venrooij, 2009; Brisson and Bianchi, 2019). Artists frequently cover several music styles throughout their career, where some tracks may be more favored than others for reasons including lyrics quality, the influence of associated music videos, over-exposure, or associations with unpleasant personal experiences (Cunningham et al., 2005). Accordingly, the work at hand investigates music recommendations on the track level to reflect users’ preferences in a more fine-grained manner than genre labels attributed to an artist’s overall repertoire could do.
- Fourth, while deep learning approaches are increasingly used for recommender systems in general and for music

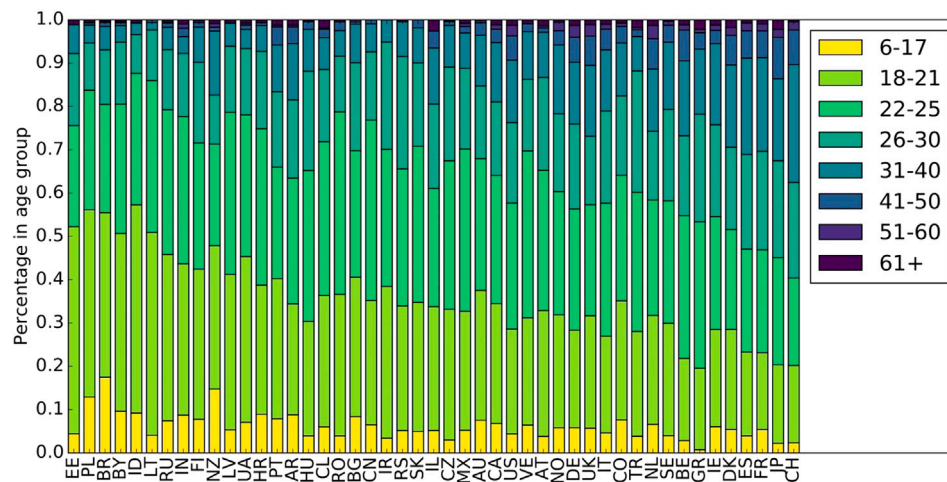


FIGURE 1 | Distribution of age over countries. Countries are sorted in decreasing order of number of users from left to right.

recommendation in particular, the integration of geographic aspects—especially user country—with deep learning for music recommendation is a particular asset of the work at hand. For instance, a recent survey on deep learning-based recommender systems (Batmaz et al., 2019) reports that extant research mainly uses textual information to capture context in approaches to context-aware recommender systems. The authors particularly consider context that is extracted from items (e.g., text documents) instead of users.

3 METHODS

In the following, we detail how we gather and process the dataset used in our study, which contains information about users' music listening behavior (Section 3.1). We then describe our approach to identify country clusters based on this dataset (Section 3.2). Finally, we elaborate on our approaches to create user models incorporating country information and we detail our neural network architecture that integrates these models (Section 3.3).

3.1 Data Acquisition and Processing

We base our investigations on the LFM-1b dataset (Schedl, 2016), which we filter according to our requirements as detailed below. The LFM-1b dataset⁴ contains music listening information for 120,322 Last.fm users, totaling to 1,088,161,692 individual listening events (LEs) generated between January 2005 and August 2014; the majority of LEs was created during years 2012–2014.⁵ Each LE is characterized as a quintuple of user-id, artist-id, album-id, track-id, and timestamp. The average number of

LEs per user in the dataset is 8,879 (std. 15,962). For some users, also demographic data (country, age, and gender) is available in LFM-1b. More precisely, 46% of the users do provide information about their country, the same percentage do provide information about their gender, and 62% about their age. The majority of users who provide their country are from the United States (18.5%), followed by Russia (9.1%), Germany (8.3%), the United Kingdom (8.3%), Poland (8.0%), Brazil (7.0%), and Finland (2.6%). The mean age of the users who reveal it is 25.4 years (std. 9.4); the median age is 23 years. The age distribution differs significantly between countries, though. In Figure 1, we show the age distribution for the countries with at least 100 users (47 countries), categorized into age groups. The youngest users are found in Estonia and Poland, while the oldest users are Swiss and Japanese. Among the users who indicate their gender, 72% are male and 28% are female. These percentages differ, however, considerably between countries. In Figure 2, we therefore depict the ratios between genders, again for the top 47 countries in terms of number of users. While the Baltic countries Lithuania and Latvia have an almost equal share of male and female users, India and Iran show a very unequal distribution (around 90% male users).

As reported above, about 46% of users in the LFM-1b dataset disclose their country. For our country-specific analysis, we therefore only consider users (and their LEs) for whom country information is available. This results in a dataset of 55,186 users, who have listened to a total of 26,021,362 unique tracks. The distribution of the number of LEs over tracks is visualized in Figure 3.

We subsequently reduce the data to decrease noise originating from the user-generated nature of the metadata in the LFM-1b dataset (in particular, misspellings and ambiguities), i.e., we filter out tracks and countries. This noise would otherwise likely cause distortions in future steps of our approach. First, we drop tracks that have been listened to less than 1,000 times, globally, resulting in a total of 122,442 tracks to consider further. Second, to minimize possible distortions caused by countries with a low number of LEs or a low number of unique users, we only consider

⁴<http://www.cp.jku.at/datasets/LFM-1b>

⁵The LFM-1b dataset used in our study is considered derivative work according to paragraph 4.1 of Last.fm's API Terms of Service (<https://www.last.fm/api/tos>). The Last.fm Terms of Service further grant us a license to use this data (according to paragraph 4).

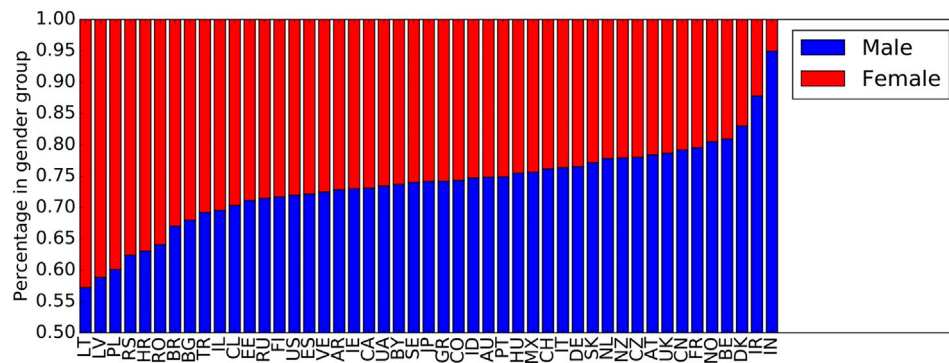


FIGURE 2 | Distribution of gender over countries. Countries are sorted in decreasing order of number of users from left to right.

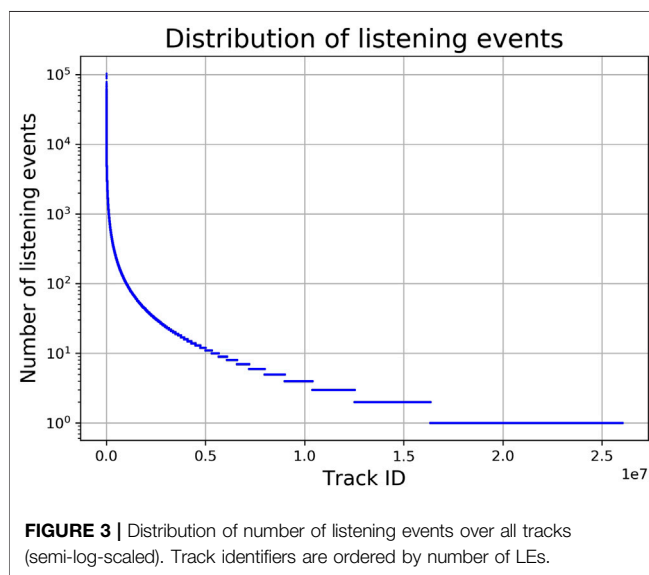


FIGURE 3 | Distribution of number of listening events over all tracks (semi-log-scaled). Track identifiers are ordered by number of LEs.

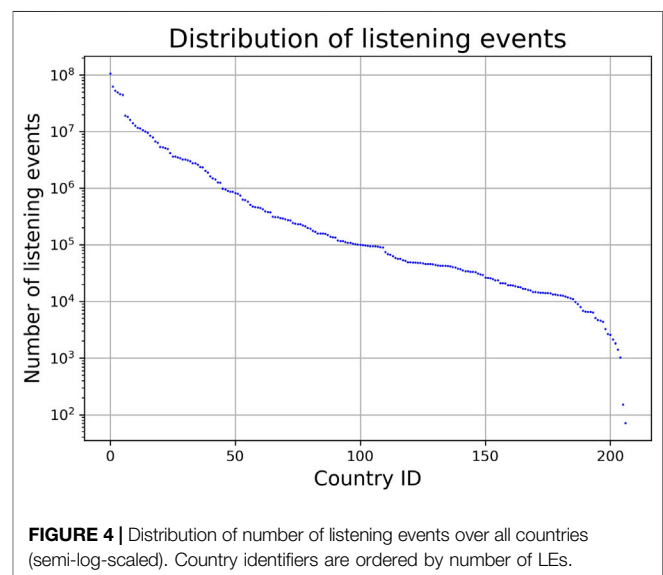


FIGURE 4 | Distribution of number of listening events over all countries (semi-log-scaled). Country identifiers are ordered by number of LEs.

countries with at least 80,000 LEs and at least 25 users. We chose these values as thresholds based on an empirical investigation of the distributions of LEs and of users over countries (cf. **Figures 4** and **5**, respectively). The former shows a flat characteristic around country-id 100, followed by a clear gap between country-id 110 and 111 (which corresponds to 80,000 LEs). The latter reveals a sudden drop at country-id 70 (which corresponds to 25 users). Applying this country filtering eventually results in 70 unique countries and a total of 369,290,491 LEs, which represents only a small drop of 1.5% (in comparison to 374,770,382 LEs created by users of all countries in the dataset). After these preprocessing steps, each country is represented as a 122,442-dimensional feature vector containing the LEs over all tracks.

3.2 Identifying Country Clusters and Archetypes

To cluster countries according to their citizens' listening behavior, it is important to first normalize the data of each

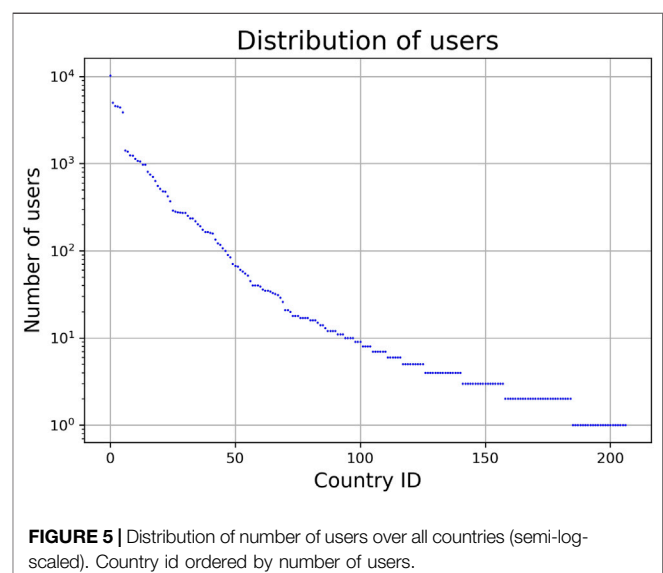


FIGURE 5 | Distribution of number of users over all countries (semi-log-scaled). Country id ordered by number of users.

country to avoid distortions caused by different country sizes. To this end, we normalize each country's feature vector to sum up to one.⁶ We next apply truncated SVD/PCA (Halko et al., 2001), reducing the dimensionality of the feature vectors to 100, while still preserving 99.8% of the variance in the data.⁷ Taking these 100-dimensional feature vectors as an input to a t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008) and subsequently using OPTICS (Ankerst et al., 1999) enables us to visualize the data and identify clusters of countries sharing similar music listening behaviors.

T-SNE is a visualization technique that embeds high-dimensional data in a low-dimensional (typically, two-dimensional) visualization space, paying particular attention to preserving the local structure of the original data. It is particularly useful to disentangle data points that lie on more than one manifold. T-SNE represents proximities or affinities between pairs of data items by estimating the probability that the first data item will choose the second one as its nearest neighbor, and vice versa. In the original data space, this probability is modeled by means of a Gaussian distribution centered around each data item in the high-dimensional space; in the visualization space by means of a t-student distribution centered around each data item in the low-dimensional space. Kullback–Leibler divergence of the joint distributions between pairs of data points in the original space and in the visualization space is then minimized via gradient descent.

Ordering Points To Identify the Clustering Structure (OPTICS) is a density-based clustering method that creates a linear ordering of data items based on their spatial proximity. For this purpose, OPTICS first identifies core data points that have at least a certain number of neighbors in their vicinity (the minimum cluster size) and assigns a core distance to them, describing how dense the area around each core point is. Furthermore, a reachability distance between each pair of data items o and p is established, which is the maximum of (1) the distance between o and p , and (2) the core distance of o , whichever is bigger. Data items assigned to the same cluster have a lower reachability distance to their nearest neighbors than items that belong to different clusters. OPTICS subsequently creates an ordering of data items in terms of their reachability distance and identifies sudden changes in reachability between neighboring items, assuming that these correspond to cluster borders. The number of clusters is controlled by a parameter ξ that defines the minimum steepness (relative change in distance) between neighboring data items to be considered a cluster boundary.⁸

⁶Please note that country-specific results may still be influenced by some users showing particularly high playcounts. Nevertheless, we decided against excluding or penalizing the listening information of such users just because users with a high playcount indicate a more pronounced inclination to listen to music. Our reasoning is that users who contribute only few listening events to Last.fm should be considered less important to model their country-specific listening behavior than users who heavily contribute. In addition, removing such “power listeners” would distort the original distribution of users’ playcounts in the sample.

⁷Reducing the dimensionality of the dataset to 50 dimensions preserves only 90.1% of the variance.

⁸In this work, we use Euclidean distance as distance metric and set $\xi = 0.05$.

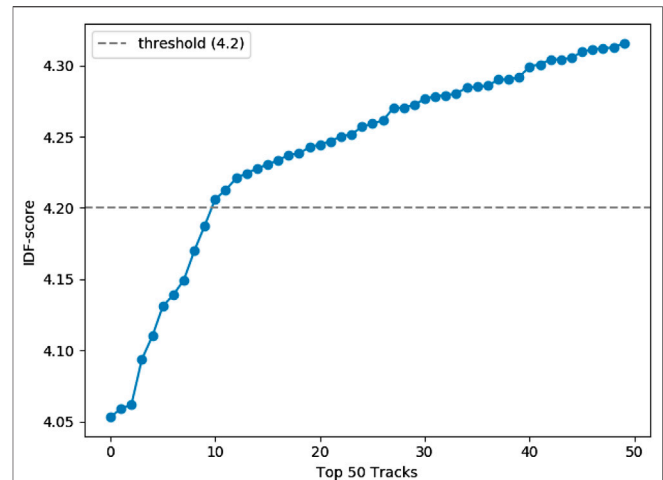


FIGURE 6 | Inverse document frequency (IDF) scores for the top 50 tracks.

As for parameter optimization, we adopt a grid search strategy to identify a well-suited perplexity for t-SNE (5) and a minimum size of clusters, i.e., minimum number of data items in each cluster, for OPTICS (3).⁹ Please note that we use ISO 3166 2-digit country codes to refer to countries in this article.¹⁰

For an analysis of the identified clusters in a way that enables the establishment of archetypes of music preferences, we adopt the following approach. As shown in **Figure 3**, we observe a long-tail distribution of listening events over tracks, which means that a few dominating tracks are listened to by a lot of users, while most tracks are only listened to by a few users. Thus, these dominating tracks will also be popular among the list of top-tracks per cluster, which makes it hard to distinguish between the clusters and to interpret their corresponding archetypes. To overcome this, we adapt a scoring function similar to the inverse document frequency (IDF) (Jones, 1972) metric from the field of information retrieval, which assigns high scores to rarely occurring tracks and low scores to frequently occurring tracks. Formally, we define IDF for each track t_i as $IDF(t_i) = \log \frac{N}{n_i}$, where N is the number of all listening events and n_i is the number of LEs for track t_i . The distribution of IDF values of the top 50 tracks, in terms of $IDF(t_i)$, is plotted in **Figure 6**. In an empirical analysis, we identify 10 overall dominating tracks using a threshold of 4.2 on the IDF values (see **Figure 6**). These tracks are Rolling in the Deep by Adele, Somebody That I Used to Know by Gotye, Islands and Intro by The xx, Blue Jeans by Lana Del Rey, Supermassive Black Hole by Muse, Skinny Love by Bon Iver as well as Use Somebody, Sex on Fire and Close by Kings of Leon. We remove these tracks from

⁹More precisely, we performed grid search on t-SNE perplexity in the range [1, 2, 3, 5, 10, 15, 20, 25, 30, 35, 40, 50] and on the minimum number of data points per cluster enforced by OPTICS in the range [2, 3, 4, 5], optimizing for average neighborhood preservation ratio (nearest neighbor consistency).

¹⁰<https://www.iso.org/iso-3166-country-codes.html>

further analyses when discussing archetypes as these are not suited to discriminate between clusters.

In our analysis of archetypes, we include genre annotations, which we obtain as follows. For all tracks in the dataset, we retrieve the top user-generated tags using the Last.fm API.¹¹ Subsequently, we filter the tags of each track using a comprehensive list of music genres and styles from Spotify, called Spotify microgenres (Johnston, 2018). This list contains 3,034 genre names (as of May 2019 when we extracted them), including umbrella genres such as pop and country, as well as smaller niches such as Thai hip-hop, German metal, and discofox (Johnston, 2018). The fine-grained reflection of subtle differences in microgenres provides a more particularized basis for describing the clusters, compared to the use of a more coarse-grained taxonomy of music genres. We note as a limitation that the microgenre categories are defined in a similarly vague manner as coarse-grained taxonomies of music genre (Beer, 2013; Sonnett, 2016; Vlegels and Lievens, 2017); and the semantics associated with (micro)genre names have evolved over time so that a precise definition appears difficult. Relying on a big corpus of data where microgenres are visualized and sonified (see The Every Noise project¹²), we nevertheless believe that using the concept of microgenres helps future research to build upon our work. Further note that we rely on the top user-generated tags from the Last.fm community for attributing microgenres to tracks; the microgenre-track associations, thus, reflect the Last.fm community's understanding of microgenres, which may not be congruent with the music experts' understanding. Additionally, synonyms may be present in the user-generated tags and, thus, two different tags could be used interchangeably to annotate the same tracks (e.g., "Rap" and "HipHop").

To allow interested readers to conduct further analyses of the identified clusters on a microgenre level, we release the full list of the top 20 tracks (and corresponding artists) per cluster, and we include—for each track and artist—all microgenre annotations.

3.3 User Modeling and Music Track Recommendation

We build our context-aware music recommendation approach on top of a variational autoencoder (VAE) model (Jordan et al., 1999). VAEs are a type of autoencoders (Kramer, 1991) that consist of an encoder, a decoder, and a loss function. In contrast to classic autoencoders, which learn encodings directly, VAEs learn the distribution of encodings using variational inference. Via sampling from the learned distribution, more representations of the same items can be generated given the same amount of training data. Thus, VAEs can learn more complex items than classic autoencoders.

We opted to extend the VAE architecture for collaborative filtering presented by Liang et al. (2018) because in a large-scale study conducted by Dacrema et al. (2019), the approach followed by Liang et al. (2018) was found the only deep neural network-

based approach that outperformed equally well tuned non-deep-learning approaches. In addition, Liang et al. (2018) evaluated their VAE architecture on the Million Song Dataset (Bertin-Mahieux et al., 2011), a common benchmark in the music domain. They showed substantially superior performance compared to several baselines, in particular, the linear model weighted matrix factorization and collaborative denoizing autoencoders.

As depicted in **Figure 7**, we extend the VAE architecture by integrating context information using a gating mechanism. The gate output modulates the latent code in a way to incorporate context-based (country and cluster) differences of users. The abstract concepts are weighted based on how important the models deem them for a specific user group.¹³ Specifically, we model users in form of a 122,442-dimensional listening vector (i.e., n_tracks), which represents their track listening history, together with context information. We investigate four different ways to define a user's context: (1) the user's country, (2) the cluster membership of the user's country, (3) the Euclidean distances between the user's listening vector and all identified cluster centroids, and (4) the Euclidean distances between the user's listening vector to all country centroids.

We derive context from the self-reported country of a user. For our VAE model with country context (i.e., model 1), a one-hot encoding of the 70 included countries is used, whereas for VAE with cluster context (i.e., model 2), context is determined by the user's country membership in a cluster (see **Table 1**), resulting in a one-hot encoding of length 9. For the context models 3 and 4, we first calculate the cluster centroids, i.e., each track's listening events of all users belonging to a cluster are summed and then normalized by the total amount of listening events across all tracks. Subsequently, for each user, the Euclidean distances between the respective user's normalized feature vector and all cluster centroids are determined and used as context features for the VAE with cluster distances (i.e., model 3). Country distances are calculated accordingly, where each country is considered as its own cluster (i.e., model 4). Taken together, $n_context$ is 70 in case of model 1 and model 4, and 9 in case of model 2 and model 3.

Our recommendation approach assumes that each user can be represented by a latent k -dimensional multivariate Gaussian, which is sampled, weighted by gates derived from context information, and transformed with a non-linear function to reconstruct the initial track listening history (cf. **Figure 7**). As mentioned before, our VAE model without contextual features is based on the work of Liang et al. (2018). To integrate context models, we extend the VAE by adding a gating mechanism to feed in contextual information according to the four ways detailed above. In a two-layer feed-forward neural network, the initial feature vector is encoded first into an intermediate representation $enc1$ and then into a latent k -dimensional multivariate Gaussian. The mean values μ and variance values σ are the outputs of the encoding network:

¹¹<https://www.last.fm/api/show/track.getTopTags>

¹²<http://everynoise.com>

¹³We also run experiments in which we simply concatenate track listening history and context information, but this did not show improvements over the VAE based on just the listening history.

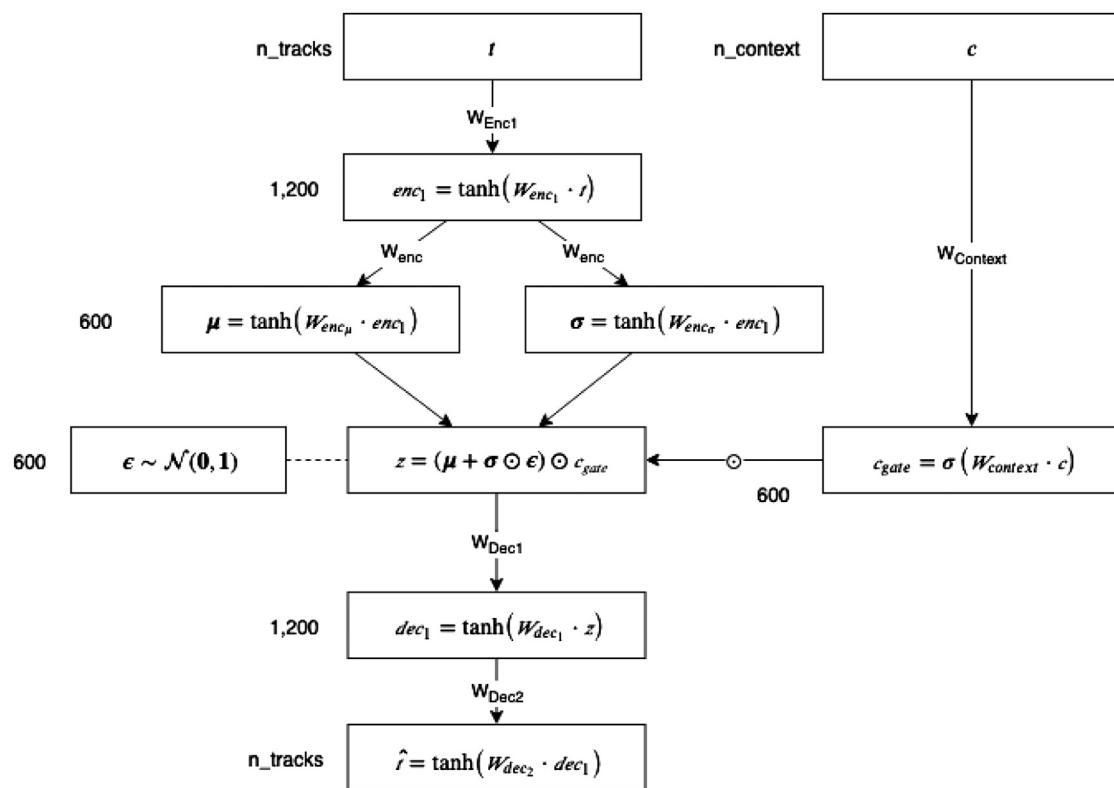


FIGURE 7 | Architecture of the variational autoencoder with gated context information.

TABLE 1 | Country clusters as determined by OPTICS with a minimum cluster size of 3, based on the output of a t-SNE visualization (perplexity of 5) on PCA-reduced country feature vectors (100 dimensions).

| Cluster | Countries |
|---------|--|
| 0 | ES, IT, IS, SI, PT |
| 1 | BE, NL, CH, SK, CZ, DE, AT, FI, PL |
| 2 | GB, EE, JP |
| 3 | AU, NZ, US, CA, PH |
| 4 | CL, CR, IL, UY |
| 5 | CO, MX, BG, GR |
| 6 | RO, EG, IR, TR, IN |
| 7 | BR, ID, VN, MY |
| 8 | LT, LV, UA, BY, RU, MD, KZ, GE |
| -1 | AQ, FR, NO, ZA, IE, MK, AR, HR, RS, BA, HU, TW, DK, HK, SG, CN, KR, PE, TH, SE, PR, VE, GT |

Countries identified as too noisy by OPTICS are represented as Cluster -1.

$$enc_1 = \tanh(W_{enc1} \cdot t) \quad (1)$$

$$\mu = \tanh(W_{enc\mu} \cdot enc_1) \quad (2)$$

$$\sigma = \tanh(W_{enc\sigma} \cdot enc_1) \quad (3)$$

We use \tanh as a nonlinearity for all layers in the autoencoder. Based on our experiments (see Section 4.2.1), we set the size of W_{enc1} to $n_tracks \times 1,200$ and both $W_{enc\mu}$ and $W_{enc\sigma}$ to $1,200 \times 600$. This results in a length of 1,200 for enc_1 and 600 for the latent representation z . The user context, given

by its input vector c is transformed by a dense layer with sigmoid nonlinearity into a context gate c_{gate} of the same length as latent z . Next, the gate is applied with component-wise multiplication to z :

$$c_{gate} = \sigma(W_{context} \cdot c) \quad (4)$$

$$\epsilon \sim \mathcal{N}(0, 1) \quad (5)$$

$$z = (\mu + \sigma \odot \epsilon) \odot c_{gate} \quad (6)$$

The weighted latent representation is then decoded back into the original space by a network with mirroring size but different learned parameters of the encoder:

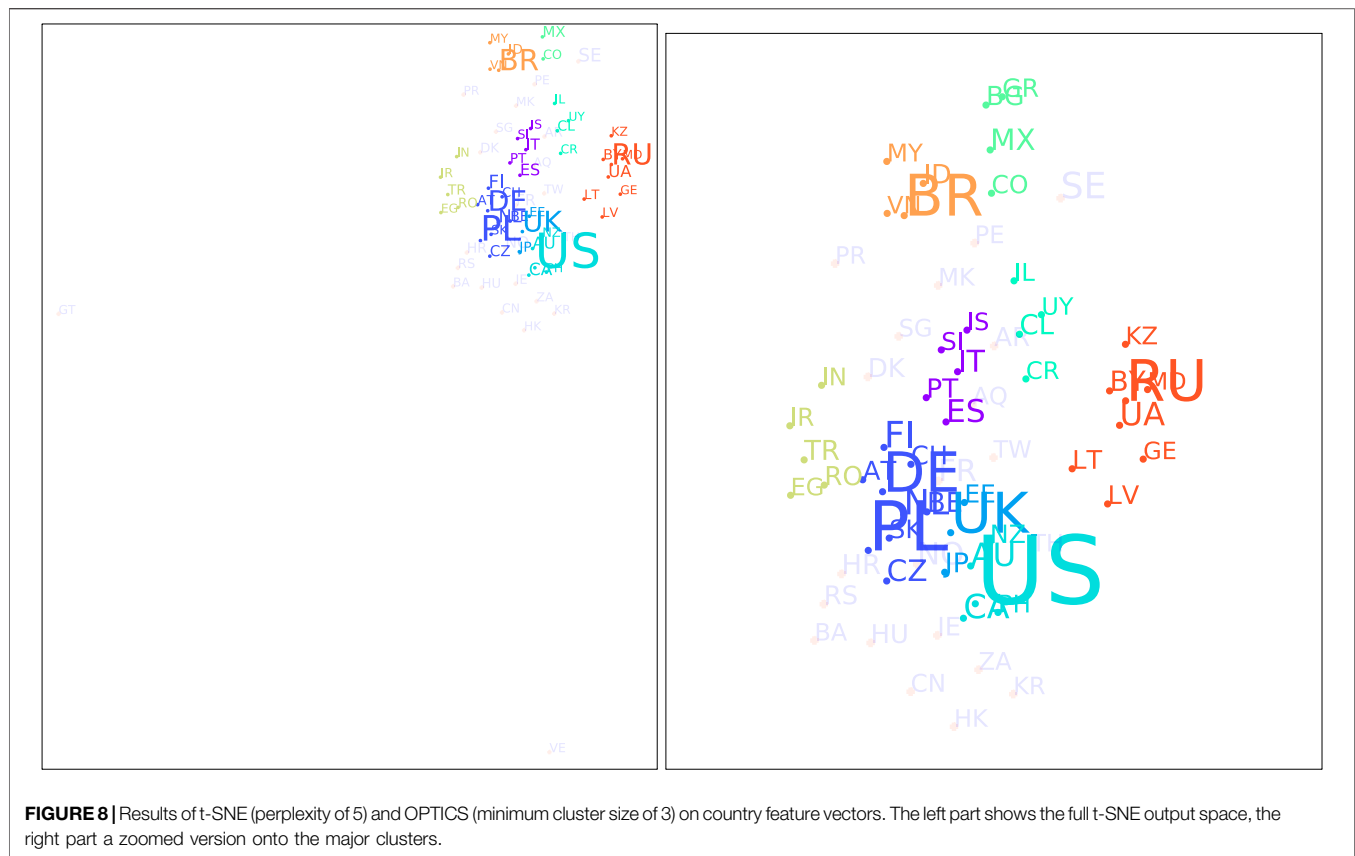
$$dec_1 = \tanh(W_{dec1} \cdot z) \quad (7)$$

$$\hat{t} = \tanh(W_{dec2} \cdot dec_1) \quad (8)$$

The detailed data flow and computation in each layer is visualized in Figure 7. Based on the known track history of a target user, the models generate a variational distribution \hat{t} . Top- k track recommendations are then retrieved by ranking the mean values of this distribution.

4 RESULTS AND DISCUSSION

In the following, we present and interpret the results of our approach to identify country clusters and archetypes of music



listening preferences (**Section 4.1**) and of the music track recommendation experiments (**Section 4.2**). We further connect the discussion to the initial research questions, which we answer in the context of the obtained results.

4.1 Clustering of Countries According to Music Listening Preferences

We present the identified clusters and discuss the relationship of the countries subsumed in each cluster beyond music preferences (**Section 4.1.1**), for instance, in terms of geographic proximity, linguistic similarities, and historical background. Furthermore, we discuss differences in user characteristics such as the users’ gender, age, and their listening patterns in terms of playcounts. In **Section 4.1.2**, we describe the characteristics of the clusters with respect to music preferences, i.e., we detail the track preferences that characterize the corresponding music archetypes.

4.1.1 Identified Country Clusters

Using the approach described in **Section 3.2**, we can identify nine country clusters, which are presented in **Table 1** and visualized in **Figure 8**. Cluster 0 contains Spain (ES), Portugal (PL), Italy (IT), Slovenia (SI), and Iceland (IS). Cluster 1 includes as many as nine countries: Belgium (BE), The Netherlands (NL), Austria (AT), Switzerland (CH), Germany (DE), Czech Republic (CZ), Slovakia (SK), Poland

(PL), and Finland (FI). Cluster 2 refers to the United Kingdom (GB), Estonia (EE), and Japan (JP). Cluster 3 includes Australia (AU), New Zealand (NZ), the United States (US), Canada (CA), and the Philippines (PH). Cluster 4 refers to Chile (CL), Costa Rica (CR), Uruguay (UY), and Israel (IL). Cluster 5 contains Colombia (CO), Mexico (MX), Bulgaria (BG), and Greece (GR). Cluster 6 the following countries: Romania (RO), Egypt (EG), Iran (IR), Turkey (TR), and India (IN). Cluster 7 is composed of Brazil (BR), Indonesia (ID), Vietnam (VN), and Malaysia (MY). Cluster 8 encompasses eight countries: Lithuania (LT), Latvia (LV), Ukraine (UA), Belarus (BY), Russia (RU), Moldova (MD), Kazakhstan (KZ), and Georgia (GE).

Four of the countries in Cluster 0 are geographically tied together, sharing national borders (i.e., Spain (ES), Portugal (PL), Italy (IT), and Slovenia (SI)). Only Iceland (IS) is geographically dislocated. Furthermore, Spain (ES), Portugal (PL), and Italy (IT) share their roots in Romance language. Moreover, there is a Slovene minority in Italy (IT), which may lead to partly similar music preferences in Slovenia (SI) and Italy (IT).

Cluster 1 contains nine countries. Belgium (BE) and the Netherlands (NL) are neighboring countries and share the official language spoken (note, Belgium (BE) has two official languages). Austria (AT), Switzerland (CH), and Germany (DE) share the German language (note, Switzerland (CH) has four official languages). Czech Republic (CZ) and Slovakia (SK) are not only



neighboring countries, but actually formed one joint country until 1992. The languages spoken in the Czech Republic (CZ), Slovakia (SK), and Poland (PL)—a neighboring country to the former two—show strong linguistic similarities. Altogether, we can see that Belgium (BE), the Netherlands (NL), Austria (AT), Switzerland (CH), Germany (DE), Czech Republic (CZ), Slovakia (SK), and Poland (PL) are geographically connected, sharing national borders (cf. **Figure 9**). Only Finland (FI) is geographically disconnected from the other countries in this cluster.

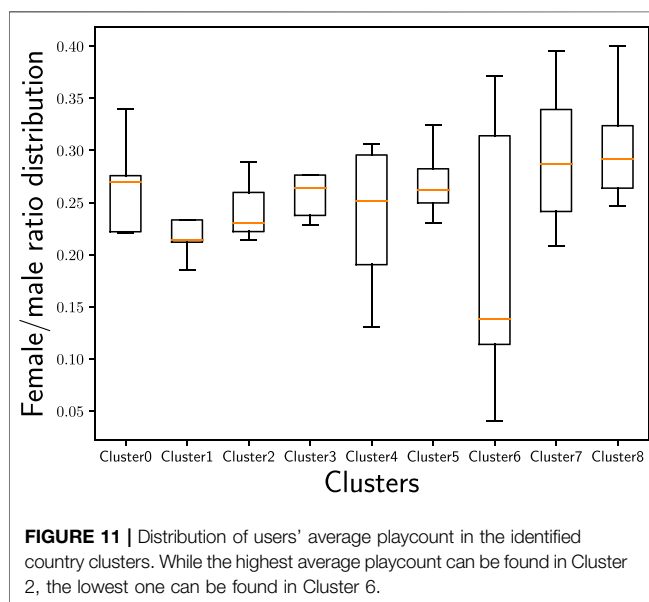
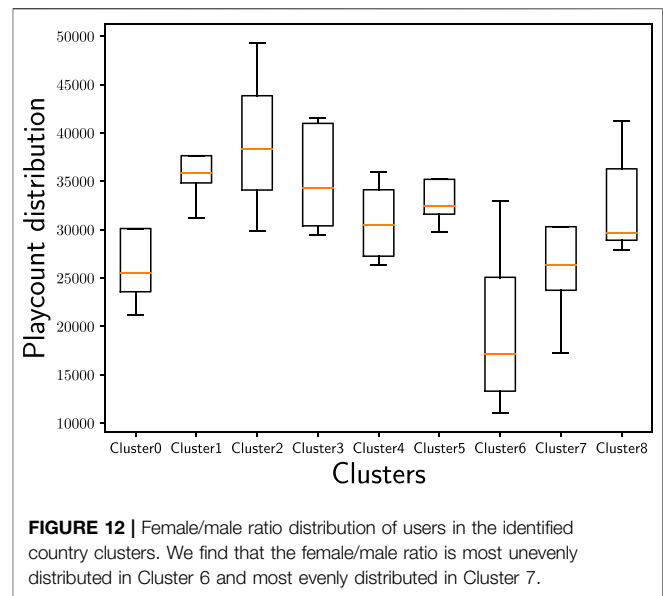
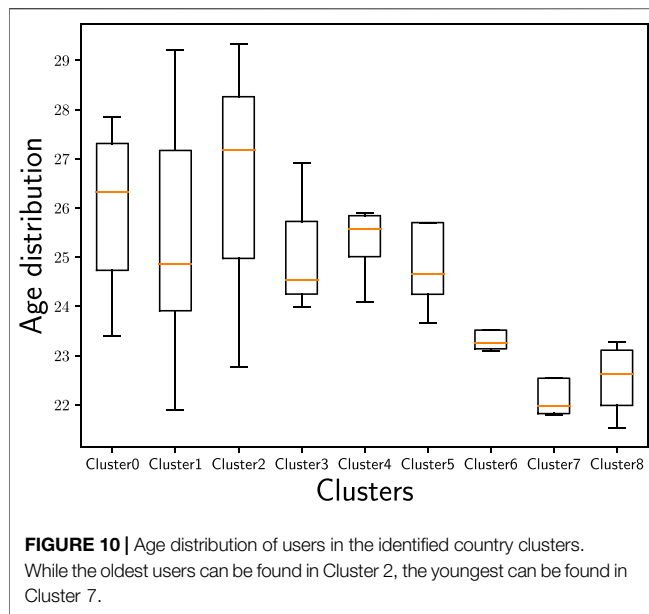
Cluster 2 delivers a highly surprising result because it contains three countries that are geographically far away from each other without any linguistic similarities or close historical connections: the United Kingdom (GB), Estonia (EE), and Japan (JP). The United Kingdom (GB) and Estonia (EE) are located at the Northwest and the Northeast of Europe—thus, at the opposite borders of Europe; Japan (JP) is even almost 8,000 km farther east of Estonia (EE). Although this cluster contains only three countries, with Japan (JP) and the United Kingdom (GB), it embraces two of the largest music markets worldwide (Statista Research Department, 2019). Interestingly, the United Kingdom (GB) is not part of Cluster 3 that includes most English-speaking countries. Considering the age distribution (**Figure 10**) in the

identified country clusters, we find that Cluster 2 shows the highest average age with a relatively large span.¹⁴ Furthermore, Cluster 2 shows by far the highest average playcount per user for the countries in this cluster (**Figure 11**). This indicates that users in this cluster are characterized as being ‘power listeners’. As the combination of countries in this cluster seems surprising, age and listening intensity may be the hidden—though determining—aspects for the emergence of this cluster.

The major connector of the countries in Cluster 3 is that they are all English-speaking countries: Australia (AU), New Zealand (NZ), United States (US), Canada (CA), and the Philippines (PH), where English is one of the two official languages in both Canada (CA) and the Philippines (PH).

Cluster 4 comprises the countries Chile (CL), Uruguay (UY), Costa Rica (CR), and Israel (IL). Both Chile (CL) and Uruguay (UY) are located in South America and are connected by their language: Spanish. The official language in Costa Rica (CR) is Spanish as well; located in Middle America, the geographic distance to Chile (CL) and Uruguay (UY) is not far. Israel (IL), in contrast, is a country in

¹⁴Please note that observations concerning age relate to our sample of Last.fm users.



the Middle East and is, thus, geographically disconnected from the other three countries in this cluster.

Cluster 5 contains two Latin-American countries as well as two countries in Southeastern Europe. The Latin-American countries, i.e., Mexico (MX) and Colombia (CO), are both Spanish-speaking countries. With Mexico (MX) located in the Southern part of North America and Colombia (CO) being part of South America, these are no neighboring countries, though. The two countries in Southeastern Europe, i.e., Bulgaria (BG) and Greece (GR), share a border. Thus, the cluster contains two country groups, which are geographically spread.

The countries in Cluster 6 are geographically connected, centered around countries being part of the Middle

East—Turkey (TR), Iran (IR), and Egypt (EG)—and flanked by Romania (RO), that has historical relations to the others due to the Ottoman Empire, and India (IN), that is adjacent to the Middle East and, thus, shows a geographical proximity to the other countries in this cluster. Furthermore, all the countries in Cluster 6 are very diverse when it comes to the various (minority) languages spoken, which may also be reflected in music preferences. Considering the female/male ratio of users (**Figure 12**) in the identified country clusters, we find that Cluster 6 shows the most unevenly distributed ratio across the countries in this cluster. Despite the wide span of female/male ratios in this cluster's countries, Cluster 6 is the cluster with the overall lowest female/male ratio compared to the other clusters. With respect to age (**Figure 10**), this cluster comprises rather young users in our sample of the Last.fm community (with the average age of users in the Clusters 7 and 8 being even younger, though). Overall, with respect to age and gender, Cluster 6 seems to have a differentiating profile compared to the other clusters. Furthermore, Cluster 6 shows by far the lowest average playcount per user (**Figure 11**). This low number could be the result of a listening pattern that is shaped by cultural aspects, but could, for instance, also be the consequence of limited access to the resources (e.g., broadband Internet connection, streaming platforms operating in the respective countries, licenses for music content). Considering those and similar aspects is a fruitful path for future research.

Cluster 7 covers three neighboring countries (with maritime borders) in the Southeast of Asia—Indonesia (ID), Vietnam (VN), and Malaysia (MY)—and Brazil (BR) in South America. The three countries in the Southeast of Asia have many similarities, including common frames of reference in history, culture, and religion; also their national languages are closely related. From a geographic perspective, Brazil (BR) appears being disconnected from the other countries in this cluster. The connection of Brazil



FIGURE 13 | Countries in Cluster 8 on a map.

(BR) with Indonesia (ID) and Malaysia (MY) is that all three countries have formerly been Portuguese colonies (Bada, 2018). Whether this historical connection is indeed also conclusive for similar music preferences is subject to further research. Referring back to **Figure 10**, where we plot the age distribution for the identified country clusters, and **Figure 12**, where we plot the female/male ratio, we see that Cluster 7 shows the lowest average age and is close to the highest female/male ratio. Furthermore, the female/male ratio is very evenly distributed in Cluster 7. We, thus, suspect that age and gender are the hidden factors construing this cluster or, at least, accentuating it.

As can be seen from **Figure 13**, Cluster 8 comprises nine countries that are in geographical proximity: the Baltic countries Lithuania (LT) and Latvia (LV), the Russian Federation (RU), Ukraine (UA), Belarus (BY), Moldova (MD), Kazakhstan (KZ), and Georgia (GA). Besides being characterized by the geographic proximity, these countries share a history of having been part of the Russian empire. Russian is a major (or influential) language in all of the countries in this cluster (Central Intelligence Agency, 2019).

Overall, we note that the country clusters show different characteristics with respect to age (**Figure 10**), gender (**Figure 12**), and average playcount per user (**Figure 11**). With respect to age, we find especially large differences between the Clusters 2 and 7: While the highest average age can be found in Cluster 2, the lowest average age can be found in Cluster 7. The female/male ratio is high in Cluster 7 and also evenly distributed. In contrast, the female/male ratio is most unevenly distributed in Cluster 6 with a high span of ratios across the countries in this cluster; and overall, the ratio is—in comparison to the other clusters—very low. With respect to the average playcount per user, it is also the Clusters 2 and 6 that show the largest differences: Among the users in Cluster 2 there seems to be a high ratio of ‘power listeners’, whereas the average playcount of users in Cluster 6 is low in comparison. Overall, it can, thus, not be rejected that those and similar aspects may be hidden factors that accentuate the differentiation between the clusters or may even be indicative for the emergence of those clusters.

4.1.2 Characteristics of the Identified Clusters and Music Preference Archetypes

To address the question what characterizes the various clusters in terms of music preferences, we use the approach described in **Section 3.2** to identify the most important tracks and genres for each cluster. **Table 2** provides a list of the 10 tracks with the highest playcounts per cluster (after the IDF-based filtering explained in **Section 3.2**) and their genre annotations;¹⁵ for genre annotations, we rely on the user-generated annotations retrieved from the Last.fm API and aligned with the Spotify microgenres, as described in **Section 3.2**. These most important tracks define the music preference archetypes corresponding to each cluster.

The most popular tracks in Cluster 0 are mainly attributed to the microgenres indie rock and alternative rock. Six tracks in the top 20 have indie rock as the most associated microgenre, three alternative rock. Eight of 20 tracks have both indie rock as well as alternative rock within their five most associated microgenres. All of the 19 tracks among the top 20 that have microgenres on track level (Si Te Quisieras Venir by the Los Planetas does not have microgenres assigned on a track level), are associated with indie rock or alternative rock; most of them with both. Only a few tracks in later positions (thus, not in the top 10) deviate from these genres (e.g., Set Fire to the Rain by Adele ranks on position 14 and is associated with the genres soul and pop, Hurt by Johnny Cash is on position 16 and is mainly associated with country and folk, or Get Lucky by Daft Punk feat. Pharrell Williams on the position 20 that is associated with electronic). With 5 of the 20 most frequently listened tracks in this cluster, the band Arctic Monkeys is particularly dominant in that cluster.

While indie rock and alternative rock are represented in the most frequently listened tracks in Cluster 0 as well as Cluster 1, the tracks in Cluster 1 differentiate insofar from those in Cluster 0 as there is a tendency that the tracks include pop or electronic elements (e.g., VCR by The xx associated with electronic and

¹⁵We released the full list of the top 20 tracks (and corresponding artists) per cluster and all microgenre annotations (for each track and artist).

TABLE 2 | The 10 most popular tracks per cluster. Playcount refers to the total number of listening events by the users in each cluster.

| Cluster no. | Track title | Artist | Playcount within cluster | Track genres |
|-------------|---------------------------------------|------------------------------|--------------------------|--|
| 0 | Mr. Brightside | The Killers | 4,248 | rock, indie rock, alternative rock |
| | Uprising | Muse | 3,955 | alternative rock, rock, progressive rock |
| | I Bet You look Good on the Dancefloor | Arctic Monkeys | 3,835 | indie rock, rock, alternative rock |
| | Fluorescent Adolescent | Arctic Monkeys | 3,772 | indie rock, rock, alternative rock |
| | VCR | The xx | 3,597 | electronic, indie rock, indie pop |
| | Reptilia | The Strokes | 3,394 | indie rock, rock, alternative rock |
| | Mardy Bum | Arctic Monkeys | 3,345 | indie rock, rock, alternative rock |
| | Hoppipolla | Sigur Rós | 3,336 | post-rock, ambient, post-rock |
| | There Is a light that Never Goes out | The Smiths | 3,289 | new wave, rock, brit-pop |
| | Teardrop | Massive Attack | 3,260 | triphop, electronic, downtempo |
| 1 | Set Fire to the Rain | Adele | 20,460 | soul, pop, singer/songwriter |
| | Little Lion Man | Mumford & Sons | 20,160 | folk, indie folk, banjo |
| | Otherside | Red hot Chili Peppers | 19,469 | rock, alternative rock, funk |
| | Radioactive | Imagine dragons | 19,338 | rock, indie rock, alternative rock |
| | VCR | The xx | 19,220 | electronic, indie rock, indie pop |
| | Heart Skipped a Beat | The xx | 19,004 | electronic, indie rock, rock |
| | Teardrop | Massive Attack | 18,810 | triphop, electronic, downtempo |
| | Sail | AWOLNATION | 18,728 | electronic, rock, indie rock |
| | The Pretender | Foo Fighters | 18,636 | rock, alternative rock, grunge |
| | Cosmic Love | Florence + the Machine | 18,486 | indie pop, rock, pop |
| 2 | There Is a Light That Never Goes Out | The Smiths | 7,479 | new wave, rock, brit-pop |
| | Mr. Brightside | The Killers | 7,128 | rock, indie rock, alternative rock |
| | Little Lion Man | Mumford & Sons | 6,979 | folk, indie folk, banjo |
| | R U Mine? | Arctic Monkeys | 6,408 | indie rock, rock, alternative rock |
| | I Bet You look Good on the Dancefloor | Arctic Monkeys | 6,302 | indie rock, rock, alternative rock |
| | I Miss You | Blink-182 | 6,295 | rock, punk, pop-punk |
| | Teardrop | Massive Attack | 6,187 | triphop, electronic, downtempo |
| | The Cave | Mumford & Sons | 6,150 | folk, indie folk, banjo |
| | VCR | The xx | 6,147 | electronic, indie rock, indie pop |
| | Harder Better Faster Stronger | Daft Punk | 6,083 | electronic, house, electronica |
| 3 | It Ain't Cool To Be CRAZY ABOUT YOU | George Strait | 19,048 | country, traditional country, |
| | Electric Feel | MGMT | 18,108 | electronic, electronica, indie pop |
| | Little Lion Man | Mumford & Sons | 17,089 | folk, indie folk, banjo |
| | Time to Pretend | MGMT | 16,802 | electronic, indietronica, electronica |
| | Flume | Bon Iver | 16,032 | folk, singer/songwriter, indie folk |
| | In the Aeroplane Over the Sea | Neutral Milk Hotel | 15,753 | indie rock, folk, lofi |
| | Midnight City | M83 | 15,635 | electronic, electro-pop, electro |
| | 1901 | Phoenix | 15,591 | indie pop, electronic, indie rock |
| | Such Great Heights | The Postal Service | 15,481 | electronic, indie pop, electronica |
| | The Cave | Mumford & Sons | 15,412 | folk, indie folk, banjo |
| 4 | Mephisto | Dead Can Dance | 2,468 | ambient, medieval, folk |
| | 3 Libras | A Perfect Circle | 1,284 | alternative rock, progressive rock, rock |
| | Ariane | Nova | 1,238 | – |
| | World's End | Hatsune Miku & Megurine Luka | 1,228 | – |
| | Mr. Brightside | The Killers | 1,109 | rock, indie rock, alternative rock |
| | Las Fuerzas | Dénver | 1,080 | – |
| | Jeremy | Pearl Jam | 1,069 | Grunge, rock, alternative rock |
| | Reckoner | Radiohead | 1,064 | Alternative rock, rock, experimental |
| | Them Bones | Alice in Chains | 1,057 | Grunge, rock, alternative rock |
| | Nude | Radiohead | 1,050 | Alternative rock, rock, electronic |
| 5 | Häaden Two | Robert Fripp | 11,616 | – |
| | The Smile | Phase | 7,898 | alternative rock, progressive rock, art rock |
| | Ibidem | Phase | 7,858 | alternative rock, art rock, rock |
| | Perdition | Phase | 7,752 | rock, psychedelic rock, progressive rock |
| | Transcendence | Phase | 7,690 | psychedelic rock, rock, alternative rock |
| | Hypoxia | Phase | 7,614 | psychedelic rock, rock, alternative rock |
| | Static | Phase | 6,988 | rock, progressive rock, space rock |
| | A Void | Phase | 6,913 | rock, alternative rock, indie rock |
| | Static (live) | Phase | 6,877 | progressive rock, psychedelic rock, rock |
| | Evening On My Dark Hillside | Phase | 6,793 | psychedelic rock, rock, alternative rock |

(Continued on following page)

TABLE 2 | (Continued) The 10 most popular tracks per cluster. Playcount refers to the total number of listening events by the users in each cluster.

| Cluster no. | Track title | Artist | Playcount within cluster | Track genres |
|-------------|----------------------------|--------------------|--------------------------|--|
| 6 | If I Could | Sophie Zelmani | 13,420 | singer/songwriter, pop, folk |
| | I Can't Change [New Song] | Sophie Zelmani | 13,409 | – |
| | Without God | Katatonia | 8,024 | doom metal, metal, death metal |
| | Day | Katatonia | 7,947 | doom metal, metal, progressive metal |
| | Lady of the Summer Night | Omega | 6,787 | Rock |
| | Sorrow | Pink Floyd | 6,485 | progressive rock, rock, classic rock |
| | Equinoxe Part 5 | Jean Michel Jarre | 6,457 | ambient, electronic rock, |
| | Gammapolis | Omega | 5,958 | classic rock, progressive rock, space rock |
| | To Know You | Sophie Zelmani | 4,783 | singer/songwriter, folk, pop |
| | To Know You (Alt. Version) | Sophie Zelmani | 4,641 | – |
| 7 | Set Fire to the Rain | Adele | 17,247 | soul, pop, singer/songwriter |
| | Fluorescent Adolescent | Arctic Monkeys | 13,007 | indie rock, rock, alternative rock |
| | Parade | Garbage | 11,770 | rock, alternative rock, pop |
| | National Anthem | Lana Del Rey | 11,602 | indie pop, pop, triphop |
| | Skyscraper | Demi Lovato | 11,451 | pop, pop-rock, disney |
| | Come & Get It | Selena Gomez | 11,387 | pop, electro-pop, dubstep |
| | Pumped Up Kicks | Foster the People | 11,171 | indie pop, pop, indie rock |
| | Dark Paradise | Lana Del Rey | 11,056 | pop, indie pop, chamber-pop |
| | Heart Attack | Demi lovato | 10,606 | pop, electro-pop, pop-rock |
| | You Only Live Once | The Strokes | 10,501 | indie rock, rock, alternative rock |
| 8 | Another Bottle Down | Asking Alexandria | 19,779 | post-hardcore, metal-core, screamo |
| | Only You | Savage | 17,657 | disco, pop, new wave |
| | ...Meltdown | Enter Shikari | 16,320 | post-hardcore, trance-core, electronic |
| | What You want | Evanescence | 12,345 | rock, alternative metal, Gothic rock |
| | Gandhi Mate Gandhi | Enter Shikari | 12,273 | post-hardcore, electronic, dubstep |
| | Dexter | Ricardo Villalobos | 11,889 | minimal, minimal techno, electronic |
| | Paradise Circus | Massive Attack | 9,922 | triphop, electronic, downtempo |
| | Teardrop | Massive Attack | 9,891 | triphop, electronic, downtempo |
| | Kill Mercy within | Korn | 9,484 | numetal, electronic, dubstep |
| | Seven Nation Army | The White Stripes | 9,380 | rock, alternative rock, indie rock |

indie rock or Cosmic Love by Florence + the Machine). Four tracks in the top 20 have indie pop as the most associated microgenre, 3 electronic. Ten tracks in the top 20 have indie rock as well as alternative rock as tagged microgenres. For all tracks except Hurt by Johnny Cash and Lonely Day by System of a Down, pop is one of the tagged microgenres. Electronic is associated with 9 of the 20 tracks.

In Cluster 2, two tracks that are most associated with folk are among the most popular tracks in the cluster (e.g., Little Lion Man or The Cave by Mumford & Sons). Among the top 20, there are 4 tracks associated mostly with folk. Tracks that are associated with electronic and pop (e.g., Judas by Lady Gaga) and tracks associated with triphop and electronic (e.g., Teardrop by Massive Attack) are also strongly represented. We recall **Figure 10** showing that Cluster 2 has the highest average age in our sample of Last.fm users. The high average age of users in Cluster 2 and the tendency to like folk music are in line with previous research that found that folk music is more established among older users compared to younger ones (Schedl and Ferwerda, 2017; Schedl and Bauer, 2017). Yet, the results in Schedl and Ferwerda (2017) suggest that the preference for folk music is more prevalent for female than for male users; this seems not to be fully in line with the characteristics of Cluster 7 at first sight because the female/male ratio in Cluster 2 is not particularly high (**Figure 12**). Delving deeper on the track characteristics, though, we notice that previous works considered a rather coarse-grained taxonomy of genres,

whereas the work at hand considers microgenres. **Table 2** shows that the 10 most popular tracks in Cluster 2 reflect indie rock (4 out of 10), alternative rock (3 out of 10), and (indie) folk (2 out of 10). In previous work (Schedl and Ferwerda, 2017), alternative (rock) was associated rather with male users (typically with younger users, though). So the indie and alternative element may suggest a rather male audience.

While the most listened song in Cluster 3 is associated with country (It Ain't Cool To Be Crazy About You by George Strait), this cluster shows a lot of tracks that are tagged with folk among the most popular ones for that cluster; 4 of the top 20 have it as their most associated microgenre. The folk tracks are either associated with folk and the singer/songwriter genre (e.g., Flume or Holocene by Bon Iver) or are attributed to indie folk (e.g., In the Aeroplane Over the Sea by Neutral Milk Hotel). Eleven tracks in the top 20 are associated with electronic or electronica within the track's five most tagged microgenres.

The most popular tracks in Cluster 4 are predominantly associated with progressive rock or alternative rock (e.g., 3 Libras by A Perfect Circle). Within the top 20 of this cluster, 10 tracks are associated with some form of progressive rock and 2 with progressive metal, 14 with alternative rock, and 9 with some form of metal (i.e., progressive metal, alternative metal, doom metal, or with the generic term metal). An interesting deviation from the dominance of the rock genre is the track World's End by Hatsune Miku & Megurine Luka, who is a vocaloid and j-pop artist. Indeed, all playcounts for that track are generated by a

single user 9 from Chile (CL); thus, this track is not representative for Cluster 4. A further deviation is constituted by Por la Ventana by Gepe associated with the genres folk and singer/songwriter, which is listened to by more than one user.

The most popular tracks in Cluster 5 are mostly associated with the psychedelic rock genre. Interestingly, 11 of the 20 most popular tracks are by the band Phase. An exception from the strong psychedelic rock representation in this cluster is the track Slow Me Down by Anneke van Giersbergen, a track that is associated with the singer/songwriter genre, while the artist is mainly associated with alternative rock and metal, but also pop-rock.

Cluster 6 is characterized by a dichotomy of genres among the most popular tracks. On the one hand, there are tracks associated with singer/songwriter and pop (e.g., If I Could and I Can't Change by Sophie Zelmani). On the other hand, there is a strong representation of doom metal with tracks such as Without God and Day by Katatonia. Interestingly, both Sophie Zelmani as well as Katatonia are present with several songs among the most popular tracks in this cluster. Recalling Figures 10–12 that visualize the user characteristics for the eight clusters, uneven distribution with respect to the female/male ratio and the generally low playcount per user (compared to the other clusters), and the young age of its users may be characterizing aspects for Cluster 6 that result in this heterogeneous picture with singer/songwriter and pop tracks, on the one hand, and the strong representation of doom metal, on the other. For instance, Schedl and Ferwerda (2017) found pop being more popular among female than male users, while it is the opposite for metal. Interestingly, the results of Schedl and Ferwerda (2017) (considering a global sample, also relying on data from Last.fm) suggest that the age group in which the users of Cluster 7 range, is the age group that likes pop least of all analyzed age groups, and for liking of meta this age group ranges in the middle field.

The only cluster that includes many popular tracks associated with the pop genre is Cluster 7. Tracks include Skyscraper by Demi Lovato, Come and Get It by Selena Gomez, and Dark Paradise by Lana Del Rey. Next to the generic tag pop (19 occurrences), the most mentioned microgenres among the top 20 in this cluster are poprock (16 occurrences) and indie pop (13 occurrences), followed by britpop (9), electro pop (6), dance pop (6), dream pop (4), synth pop (3), chamber pop (3), alternative pop (3), teen pop (2), art pop (2), power pop (1), jangle pop (1), and k-pop (1). The high ratio of female users (Figure 12) might be a cohesive characteristic in this cluster as already previous work has shown that female users are more inclined to listen to pop music than male users, in particular in the Last.fm community (Schedl and Bauer, 2017; Schedl and Ferwerda, 2017).

Cluster 8 is characterized by the post-hardcore genre. Seven tracks in the top 20 in this cluster are tagged with post-hardcore, five of those have it as their most tagged microgenre. Triphop (8 tracks), screamo (6 tracks), and hardcore (6 tracks) are also well represented among the top 20 in this cluster. Popular tracks include Another Bottle Down by Asking Alexandria, ... Meltdown by Enter Shikari, and Nineteen Fifty Eight by

A Day to Remember. An interesting deviation from this post-hardcore association are, for instance, Dexter by Ricardo Villalobos (minimal techno) and Cookie Thumper! by Die Antwoord (hip hop), which are also among the most popular tracks in this cluster.

Summarizing the answer to RQ1, which we addressed here (To what extent can we identify and interpret groups of countries that constitute *music preference archetypes*, from behavioral traces of users' music listening records?), we find nine clusters of countries, with each of the clusters representing a *music preference archetype* that reflects different nuances of music preferences in terms of the Spotify microgenres. While some *music preference archetypes* represent countries with geographical proximity (e.g., Cluster 6 and Cluster 8) and some archetypes share linguistic similarities (e.g., Cluster 3 and Cluster 8), others include interesting outliers (e.g., Iceland (IS) in Cluster 0, Israel (IL) for Cluster 4, or Brazil (BR) in Cluster 7).

4.2 Music Track Recommendation Using Country Context

In the following, we first detail the setup of the conducted evaluation experiments for the music track recommendation task, including evaluation protocol, baselines, and performance metrics (Section 4.2.1). Subsequently, we report and discuss the obtained results and answer the related research questions (Section 4.2.2).

4.2.1 Experimental Setup

After preselection and filtering (cf. Section 3.1), the dataset contains the listening histories of 54,337 Last.fm users. To carry out the recommendation experiments, we split the data into training, validation, and test sets. For each of validation and test set, 5,000 users are randomly sampled. The original VAE model (Liang et al., 2018) and our extended VAE architecture that integrates the user context models described in Section 3.3 are trained on the full listening events of the users in the training set. For users in the validation and test set, 80% of all listening events are randomly selected to act as an input for the model, and the remaining 20% are used for evaluation. The NDCG@100 metric (see below) on the validation set is used to select the hyperparameters of our models.

Baselines: In addition to comparing our extended context-aware model to the original VAE recommendation architecture (Liang et al., 2018), we also include two baselines in the experiments, i.e., variants of most popular (MP) and implicit matrix factorization (IMF). In the *most popular* (MP) models, a popularity measure is calculated for each track based on its sum of listening events across users in the training set. We implemented and evaluated three flavors of MP: *MP global* computes the most popular tracks on a global scale (independent of country); *MP country* considers only the top tracks in the country of the target user; *MP cluster* considers only the top tracks within the cluster the country of the target user belongs to. We then rank tracks accordingly and use the ranking to produce recommendations, which are evaluated on the 20% split of the test set (for each user). To make results between the baseline and our proposed model

comparable, we exclude tracks that are part of a user's known listening history, i.e., listening events from the remaining 80%. As a second baseline, we adopt a collaborative filtering approach using *implicit matrix factorization* (IMF) according to Koren et al. (2009). We use the implementation provided by Spotlight¹⁶ with random negative sampling (50:50), 128 latent dimensions, and a pointwise loss function.

Performance metrics: To quantify the accuracy of the recommendations, we use the following metrics (similar to Liang et al., 2018; Schedl et al., 2018; Aioli, 2013), which we report averaged over all users (in the test set). Thus, for each user in the test set, we generate recommendations using the data in the training set and compare the recommended tracks with the actually listened tracks of the user present in the test set in order to calculate the performance metrics. Note that we use the definitions common in recommender systems research, which are partly different from the ones traditionally used in information retrieval. *Precision@K* for user u :

$$P@K(u) = \frac{1}{K} \sum_{i=1}^K rel(i), \quad (9)$$

where K is the number of recommended items and $rel(i)$ is an indicator function signaling whether the recommended track at rank i is relevant to u or not. This means that $rel(i) = 1$ if the recommended track at rank i can be found in the test set; $rel(i) = 0$ if not. *Recall@K* for user u :

$$R@K(u) = \frac{1}{\min(K, N_u)} \sum_{i=1}^K rel(i) \quad (10)$$

where N_u is the number of items in the test set that are relevant to u , K is the size of the recommendation list, and $rel(i)$ is the same indicator function as used for *Precision@K*. When comparing *Precision@K* and *Recall@K*, *Precision@K* can be seen as a measure of the usefulness of recommendations and *Recall@K* as a measure of the completeness of recommendations. *Normalized discounted cumulative gain@K*:

$$NDCG@K(u) = \frac{DCG@K(u)}{IDCG@K(u)} \quad (11)$$

where $IDCG@K(u)$ is the ideal $DCG@K$ for user u , achieved when all items relevant to u are ranked at the top, and $DCG@K(u)$ is the *discounted cumulative gain* at position k for user u . It is given by:

$$DCG@K(u) = \sum_{i=1}^K \frac{rel(i)}{\log_2(i+1)} \quad (12)$$

where $rel(i)$ is the same indicator function as used for *Precision@K* and *Recall@K*. In contrast to those two performance metrics, *NDCG@K* is a ranking-based metric, which also takes the position of the recommended tracks into account since higher-ranked items are given more weight.

We compute and report all metrics for $K = 10$ and $K = 100$, simulating users who are just interested in a few top recommendations and users who inspect a large part of the recommendation list, respectively.

4.2.2 Results and Discussion

Table 3 shows the performance achieved on the test set, averaged over all users in the test set. As a general observation, we see that the VAE-based approaches outperform the baselines (MP and IMF) by a substantial extent. Of the baselines, IMF performs superior to MP global while the other two variants of MP (MP country and MP cluster) yield better results than IMF. The poor performance of MP global is somewhat surprising since several studies (e.g., Tiwari et al., 2018; Lai et al., 2019; Vall et al., 2019) have shown that recommendation approaches leveraging popularity information—e.g., always suggesting the items that are most frequently consumed—often achieve highly competitive accuracy values in offline experiments, despite the obvious fact that such recommendations will likely not be perceived very useful by the users. A likely reason is that we perform track recommendation while the earlier mentioned works commonly adopt an artist recommendation setup. In an artist recommendation scenario, it is very likely that a user has consumed every highly popular artist at least once. This leads to a high performance of a popularity-based approach. In the track recommendation scenario adopted in the work at hand, the granularity of items (tracks vs. artists) is higher and—in comparison to the artist recommendation scenario—it is not necessarily the case that the most popular tracks have been consumed by most users at least once. Overall, a popularity-based approach may work well for artist recommendation but less so for the more fine-grained track recommendation.

On the other hand, we also note that the other two variants (MP country and MP cluster) achieve much better results than MP global, even outperforming the IMF approach. This might be explained by the more narrow but better user-tailored consideration of the country-specific mainstream (cf. Bauer and Schedl, 2019), which is reflected in the computation of most popular tracks in the MP country and MP cluster models.

Comparing the proposed context-aware extensions of the VAE recommendation architecture to the original VAE (Liang et al., 2018), we observe a clear improvement of all metrics when integrating the user context models. This improvement is achieved irrespective of the actual user model we adopt (models 1–4). *Precision@10* increases by 3.4 percentage points (7.1%) from VAE to the best performing VAE context model (model 4) that leverages the distances between users and country centroids. Likewise, *Precision@100* increases by 1.7 percentage points (5.5%). *Recall@10* and *Recall@100* improve, respectively, by a maximum of 3.5 percentage points (7.2%), realized by model 4, and by 1.8 percentage points (4.9%), realized by model 2. In terms of *NDCG*, the largest gains are realized by VAE context model 2 that incorporates cluster ids. *NDCG@10* improves by 3.7 percentage points (7.4%) compared to VAE; *NDCG@100* increases by 2.1 percentage points (5.5%).

We investigate statistical significance of the results as follows. For all used metrics (i.e., $P@10$, $P@100$, $R@10$, $R@100$, $NDCG@10$,

¹⁶<https://maciejkula.github.io/spotlight/factorization/implicit.html>

TABLE 3 | Results with respect to Precision@K, Recall@K, and NDCG@K metrics.

| Model | P@10 | P@100 | R@10 | R@100 | NDCG@10 | NDCG@100 |
|-------------------------------------|-------|-------|-------|-------|---------|----------|
| MP global | 0.048 | 0.033 | 0.048 | 0.036 | 0.050 | 0.037 |
| MP country | 0.203 | 0.156 | 0.203 | 0.157 | 0.209 | 0.166 |
| MP cluster | 0.193 | 0.149 | 0.193 | 0.149 | 0.199 | 0.158 |
| IMF | 0.080 | 0.072 | 0.080 | 0.064 | 0.081 | 0.071 |
| VAE | 0.482 | 0.309 | 0.486 | 0.367 | 0.500 | 0.383 |
| VAE country id (model 1) | 0.513 | 0.325 | 0.517 | 0.384 | 0.532 | 0.402 |
| VAE cluster id (model 2) | 0.515 | 0.326 | 0.520 | 0.385 | 0.537 | 0.404 |
| VAE cluster dist (model 3) | 0.513 | 0.325 | 0.518 | 0.384 | 0.534 | 0.403 |
| VAE country dist (model 4) | 0.516 | 0.325 | 0.521 | 0.383 | 0.535 | 0.403 |
| VAE sampling | 0.224 | 0.099 | 0.239 | 0.255 | 0.252 | 0.223 |
| VAE sampling country id (model 1) | 0.230 | 0.102 | 0.245 | 0.259 | 0.259 | 0.227 |
| VAE sampling cluster id (model 2) | 0.231 | 0.101 | 0.246 | 0.259 | 0.261 | 0.227 |
| VAE sampling cluster dist (model 3) | 0.232 | 0.102 | 0.245 | 0.258 | 0.246 | 0.258 |
| VAE sampling country dist (model 4) | 0.225 | 0.100 | 0.239 | 0.255 | 0.255 | 0.223 |

For all metrics, pairwise comparison using a Wilcoxon signed-rank test shows significant improvements from MP global to IMF to VAE to all VAE models with context (models 1–4); there are no significant differences between the 4 VAE models that use context, though. The five rows at the bottom (“VAE sampling . . .”) show results for another set of experiments in which we randomly sampled (three times) exactly 122,442 tracks from about 1 million tracks instead of computing performance measures on the top 122,442 tracks of the whole collection as done in the main experiment.

NDCG@100), data is non-normally distributed (Kolmogorov-Smirnov test, $p \leq 0.001$). Accordingly, we use the Friedman test (Friedman, 1937) to test the models’ performances for differences. For each metric, the models differ at a significance level of $p \leq 0.001$. In pairwise comparisons using a Wilcoxon signed-rank test (Wilcoxon, 1945), for each metric, the tests indicate that VAE outperforms each of the baselines (i.e., MP and IMF) at a significance level of $p \leq 0.05$. Furthermore, we perform a pairwise comparison, again using Wilcoxon’s signed-rank test, for each metric and each pair of pure VAE and one of the models integrating context information (i.e., models 1–4). For each metric and each of the models 1–4, the models 1–4 outperform the pure VAE (without context integration) at a significance level of $p \leq 0.05$. Yet, the Friedman test did not indicate any significant differences of the models 1–4 for any of the metrics.

Returning to the original research questions, we answer RQ2 (Which are effective ways to model the users’ geographic background as a contextual factor for music recommendation?) by pointing to the fact that all four user models proposed are effective to significantly improve recommendation quality in terms of precision, recall, and NDCG measures. We note, however, that performance differences between the four user context models are largely negligible. In summary, leveraging country information for music track recommendation (either as country or cluster identifier, or as distances between the target user and each cluster’s centroid) is beneficial compared to not including any country information.

As for RQ3 (How can we extend a state-of-the-art recommendation algorithm to include user context information, in particular, our geo-aware user models?), we proposed an extension of a state-of-the-art recommender based on a VAE architecture (Liang et al., 2018), i.e., we devised a multi-layer generative model in which contextual features can influence recommendations through a gating mechanism.

To investigate the generalizability of our findings to a dataset with different characteristics, we perform an additional experiment as follows. We estimate performance on a more diverse dataset in terms of track popularity than the one that considers only the top 122,442 tracks. More precisely, we create a second dataset by first considering all tracks that have been listened to at least 100 (instead of 1,000) times, yielding 1,012,961 unique tracks. We then randomly sample, three times, exactly the same amount of tracks (122,442) as used in our main experiment, and we evaluate the VAE approaches on each randomly sampled subset, averaging performance measures across the three runs.¹⁷ Results can be found in the five last rows of Table 3 (models named “VAE sampling . . .”). While we observe an obvious decrease in performance when considering items further down the popularity scale, results are still in line with the findings obtained on the main dataset. In particular, our extended VAE models (models 1–4) still outperform the original VAE architecture, with respect to all performance metrics.

5 CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

In summary, we approached the task of identifying country clusters and corresponding archetypes of music consumption preferences based on behavioral data of music listening that originates from Last.fm users. Together with the users’ self-disclosed country information, we used the listening data (369 million listening events created by 54 thousand Last.fm users) as an input to unsupervised learning techniques (t-SNE and OPTICS), allowing us to *identify nine archetypal country clusters*. We discussed these clusters in detail with respect to

¹⁷Please note that computational limitations prevented us from running experiments on all 1,012,961 tracks, even more so on the entire LFM-1b dataset.

their corresponding users' music preferences on the track level and the linguistic, historical, and cultural backgrounds of the countries in each cluster. Additionally, we considered the distribution of age, gender, and average playcount per user as aspects in our analysis.

Furthermore, we proposed a context-aware music recommendation approach operating on the music track level, which integrates different user models that are based on the user's country or country cluster. To this end, we *extended a variational autoencoder (VAE) architecture by a gating mechanism to add contextual user features*. We considered *four user models*, either encoding the target user's country information (model 1) or cluster information (model 2) directly, or as a feature vector containing the distances between the target user and all cluster centroids (model 3) or all individual country centroids (model 4). In evaluation experiments, using precision, recall, and NDCG as performance metrics, we showed that all VAE architectures outperformed a popularity-based recommender and implicit matrix factorization, which served as our baselines. Results further revealed *superior performance of all VAE variants that include context information vis-à-vis VAE without context information*, regardless of how country information is encoded in the user model.

Yet, this work has potential **limitations** with respect to the underlying dataset, which we discuss in the following. There are social patterns that define how and why people access music (López-Sintas et al., 2014). A dataset containing logs of the interactions with an online platform can, thus, only capture those listening events of people using any form of online music platform. According to López-Sintas et al. (2014), music access patterns are structured by an individual's social position (indicated by education) and life stage (indicated by age). A bias with respect to the users' social background can therefore be expected for our dataset. For instance, the dataset has a strong *community bias* toward users in the United States (US), while other countries are less represented. Furthermore, *user information is self-reported* by the users, which may be prone to errors and may not necessarily reflect the truth. For instance, some users report as their country Antarctica (AQ) or a birth year of 1900, which both do not seem overly plausible—especially in combination (also see Figure 1 in Schedl, 2017). Moreover, some users show *very high playcounts for single tracks*, which are not popular among other users. This also affects six of the tracks presented in our discussion of the music preference archetypes. For instance, World's End by Hatsune Miku & Megurine Luka has a playcount of 1,228 generated by a single unique user. Similarly, One Thing' by Runrig and Resemmare by Valeriu Sterian both have exactly one unique listener, who generated a playcount of 4,000 and 3,591, respectively. The track Ariane by Nova has 3 unique users; I Can't Change [New Song] and To Know You (Alt. Version)—both by Sophie Zelmani—have 5 unique users each, whereof almost all playcounts were generated by only one single user. For both songs, this is the same user. Notably, also the preferences of the Last.fm users in our dataset toward certain genres differ from the genre preferences of the population at large. For instance, we found that rap and R&B as well as classical music is substantially underrepresented in Last.fm listening data (Schedl and Tkalcic,

2014), which we use in the present study. To some extent, these limitations related to the dataset could be alleviated in the future by performing further data cleansing and preprocessing steps, e.g., threshold-based filtering of exorbitant playcounts by a minority of listeners.

Another limitation of the work concerns a characteristic of t-SNE, which is that the *cost function t-SNE uses is non-convex*. This, in turn, may result in a different embedding of data points in the low-dimensional output space when the t-SNE algorithm is run on different software or hardware configurations.¹⁸ Please note that this does not only concern the present work, but potentially the entire (large) body of research that employs t-SNE for visualization. It is, however, an aspect that is barely discussed. We address this issue in the current work by providing exact details on our implementation and used software, and by releasing to the public the source code, parameter configurations, and dataset used in our experiments.

In this work, we used simple mechanisms to integrate country information as context factors into a VAE architecture. While they worked out well, i.e., outperformed a non-context-aware VAE, we expect even better performance with other user models, whose creation will be part of **future research**. For instance, we contemplate using *probabilistic models* to describe the likelihood of each user to belong to each cluster (or country), e.g., via Gaussian mixture models. Given the actual country of a user, we could then analyze in more detail users whose stated country is not the country with highest probability. Such a framework could also be used to *diversify recommendations* according to a user-selected country, fulfilling user intents such as “I want music of my preferred genre, but listened to by Brazilians”.

Furthermore, it would be worthwhile to *compare the clustering and recommendation results* we achieved here on the track level to results achieved when modeling music preferences on the artist level, keeping all other methodological details the same. In particular, since previous studies have predominantly shown that popularity-based music recommendation systems perform well when recommending artists, such a comparison could be enlightening.

Finally, we aim at delving into the possible *cultural, historical, or socio-economic reasons* that may underlie the differences in music preferences between the identified archetypes. To this end, we will consider theories and insights from cultural sciences, history, sociology, and economics, and connect our music preference archetypes to these theories. Another promising path for further analysis of the country clusters is to consider dimensions rooted in the music market or the music content itself, including considerations such as local demand, production of music styles, reception of music styles, diffusion, etc., as well as dimensions related to the users' listening habits.

¹⁸Note that our results are stable for a given machine, software configuration, and parameter setting since we fixed the seed of the random number generator. Running the code on other configurations, however, may result in a slightly different visualization and clustering.

DATA AVAILABILITY STATEMENT

To foster reproducibility, we release the code and data used in this work to the public. The code can be found on https://gitlab.cp.jku.at/markus/fiai2020_country_clusters; the dataset (Schedl et al., 2020) is available from <https://zenodo.org/record/3907362#.XvRq1CgzZPY>. In our experiments, we used the following setup: Windows 10 Home, Build 18,362; Python 3.7.0; T-SNE and OPTICS implementations of scikit-learn 0.21.3. The remaining, system-independent, configuration details can be found in the code.

AUTHOR CONTRIBUTIONS

Conceptualization (CB, EL, DK, and MS); Data curation (CB, MS, and WR); Methodology (CB, DK, EL, MS, and WR); Investigation (CB, DK, EL, MS, and WR); Software (DK, MS, and WR); Visualization (CB, MS, and WR); Writing (CB, DK, EL, MS, and WR).

REFERENCES

- Adiyansjah Gunawan, A. A. S., and Suhartono, D. (2019). Music recommender system based on genre using convolutional recurrent neural networks. The 4th international conference on computer science and computational intelligence (ICCCSI 2019): enabling collaboration to escalate impact of research results for society. *Proc. Comput. Sci.* 157, 99–109. doi:10.1016/j.procs.2019.08.146
- Adomavicius, G., and Tuzhilin, A. (2015). “Context-aware recommender systems,” in *Recommender systems handbook*. Editors F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. 2nd edn. (Berlin: Springer), 191–226.
- Aioli, F. (2013). “Efficient top-n recommendation for very large scale binary rated datasets RecSys ’13,” in Proceedings of the 7th ACM conference on recommender systems. (New York, NY, USA: ACM), 273–280. doi:10.1145/2507157.2507189
- Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). “Optics: ordering points to identify the clustering structure,” in Proceedings of the 1999 ACM SIGMOD international conference on management of data, ’99 SIGMOD. (New York, NY, USA: ACM), 49–60. doi:10.1145/304182.304187
- Bada, F. (2018). Former portuguese colonies, WorldAtlas. Available at: <https://worldatlas.com/articles/former-portuguese-colonies.html>
- Baek, Y. M. (2015). Relationship between cultural distance and cross-cultural music video consumption on youtube. *Soc. Sci. Comput. Rev.* 33, 730–748. doi:10.1177/0894439314562184
- Batmaz, Z., Yurekli, A., Bilge, A., and Kaleli, C. (2019). A review on deep learning for recommender systems: challenges and remedies. *Artif. Intell. Rev.* 52, 1–37. doi:10.1007/s10462-018-9654-y
- Bauer, C., and Schedl, M. (2019). Global and country-specific mainstreamness measures: definitions, analysis, and usage for improving personalized music recommendation systems. *PLoS One* 14, 1–36. doi:10.1371/journal.pone.0217389
- Bauer, C., and Schedl, M. (2018). “On the importance of considering country-specific aspects on the online-market: an example of music recommendation considering country-specific mainstream,” in 51st Hawaii international conference on system sciences. 3647–3656. Available at: <http://hdl.handle.net/10125/50349>
- Bauer, C., and Zangerle, E. (2019). “Leveraging multi-method evaluation for multi-stakeholder settings,” in 1st workshop on the impact of recommender systems, co-located with 13th ACM conference on recommender systems (ACM RecSys ’19). Editors O. S. Shalom, D. Jannach, I. Guy, and Ceur-ws.org. Available at: <http://ceur-ws.org/Vol-2462/short3.pdf>
- Beer, D. (2013). Genre, boundary drawing and the classificatory imagination. *Cult. Sociol.* 7, 145–160. doi:10.1177/1749975512473461

FUNDING

This research is supported by the Austrian Science Fund (FWF): V579 and the Know-Center GmbH (FFG COMET funding). This work was supported by the H2020 project TRUSTS (GA: 871481) and the Know-Center GmbH. The Know-Center GmbH is funded within the Austrian COMET (Competence Centers for Excellent Technologies) Program under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency (FFG).

ACKNOWLEDGMENTS

The authors would like to thank Peter Müllner from the Know-Center GmbH for providing the IDF calculations of the music tracks.

- Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., and Lamere, P. (2011). “The million song dataset,” in *Proceedings of the 12th International Society for music information retrieval conference, ISMIR 2011*. Editors A. Klapuri and C. Leider (Miami, Florida, USA: University of Miami), 591–596.
- Bonneville-Roussy, A., Rentfrow, P. J., Xu, M. K., and Potter, J. (2013). Music through the ages: trends in musical engagement and preferences from adolescence through middle adulthood. *J. Pers. Soc. Psychol.* 105, 703–717. doi:10.1037/a0033770
- Bonneville-Roussy, A., and Rust, J. (2018). Age trends in musical preferences in adulthood: 2. sources of social influences as determinants of preferences. *Music Sci.* 22, 175–195. doi:10.1177/1029864917704016
- Brisson, R., and Bianchi, R. (2019). On the relevance of music genre-based analysis in research on musical tastes. *Psychol. Music* 12, 33–39. doi:10.1177/0305735619828810
- Bryson, B. (1997). What about the univores? musical dislikes and group-based identity construction among americans with low levels of education. *Poetics* 25, 141–156.
- Budzinski, O., and Pannicke, J. (2017). Do preferences for pop music converge across countries?: Empirical evidence from the eurovision song contest. *Creativ. Ind. J.* 10, 168–187. doi:10.1080/17510694.2017.1332451
- Central Intelligence Agency (2019). Languages. The World Factbook. Available at: <https://www.cia.gov/library/publications/resources/the-world-factbook/fields/402.html>
- Cheng, Z., Shen, J., Nie, L., Chua, T.-S., and Kankanhalli, M. (2017). “Exploring user-specific information in music retrieval,” in 40th International ACM SIGIR conference on research and development in information retrieval, ’17. New York, NY, USA, 655–664. doi:10.1145/3077136.3080772
- Colley, A. (2008). Young people’s musical taste: relationship with gender and gender-related traits. *J. Appl. Soc. Psychol.* 38, 2039–2055. doi:10.1111/j.1559-1816.2008.00379.x
- Coulangeon, P. (2003). La stratification sociale des goûts musicaux. le modèle de la légitimité culturelle en question. *Rev. Fr. Sociol.* 44, 3–33. doi:10.3917/rfs.441.0003
- Coulangeon, P., and Roharik, I. (2005). “Testing the “omnivore/univore” hypothesis in a cross-national perspective. on the social meaning of eclecticism in musical tastes,” in The summer meeting of the ISA RC28. UCLA.
- Coulangeon, P. (2005). Social stratification of musical tastes: questioning the cultural legitimacy model. *Rev. Fr. Sociol.* 46, 123–154. doi:10.3917/rfs.465.0123
- Cunningham, S. J., Downie, J. S., and Bainbridge, D. (2005). ““The pain, the pain”: modeling music information behavior and the songs we hate,” in Proceedings of the 6th international conference on music information retrieval. (London, UK: ISMIR 2005), 474–477.
- Dacrema, M. F., Cremonesi, P., and Jannach, D. (2019). “Are we really making much progress? A worrying analysis of recent neural recommendation approaches,” in Proceedings of the 13th ACM conference on recommender

- systems. '19. (New York, NY, USA: Association for Computing Machinery), 101–109. doi:10.1145/3298689.3347058
- DiMaggio, P. (1982). Cultural entrepreneurship in nineteenth-century boston: the creation of an organizational base for high culture in america. *Media Cult. Soc.* 4, 33–50. doi:10.1177/016344378200400104
- Dolata, U. (2013). *The transformative capacity of new technologies: a theory of sociotechnical change* Of routledge *Advances in sociology*. London, United Kingdom: Routledge.
- Fisher, T. C., and Preece, S. B. (2003). Evolution, extinction, or status quo? canadian performing arts audiences in the 1990s. *Poetics* 31, 69–86. doi:10.1016/S0304-422X(03)00004-4
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.* 32, 675–701. doi:10.1080/01621459.1937.10503522
- Gomez Herrera, E., and Martens, B. (2018). Language, copyright and geographic segmentation in the eu digital, single market for music and film. *Rev. Econ. Res. Copyr. Issues.* 15, 20–37.
- Halko, N., Martinsson, P.-G., and Tropp, J. A. (2001). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* 53, 217–288. doi:10.1137/090771806
- Helliwell, J. F., Layard, P. R., and Sachs, J. (2016). *World happiness report 2016 update (sustainable development solutions network)*. Berlin: Springer.
- Hofstede, G., Hofstede, G. J., and Minkov, M. (2005). *Cultures and organizations: software of the mind*. New York, NY USA: McGraw-Hill.
- Holbrook, M. B., Weiss, M. J., and Habich, J. (2002). Disentangling effacement, omnivore, and distinction effects on the consumption of cultural activities: an illustration. *Market. Lett.* 13, 345–357. doi:10.1023/A:1020322600709
- Horkheimer, M., and Adorno, T. W. (1972). *Dialectic of enlightenment*. New York, NY, USA: Seabury Press.
- Hracs, B. J., Seman, M., and Virani, T. E. (2017). *The production and consumption of music in the digital age*. New York, NY, USA: Routledge.
- Johnston, M. (2018). *How spotify discovers the genres of tomorrow*. Available at <https://artists.spotify.com/blog/how-spotify-discovers-the-genres-of-tomorrow>.
- Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* 28, 11–21.
- Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Mach. Learn* 37, 183–233.
- Katz-Gerro, T. (1999). Cultural consumption and social stratification: leisure activities, musical tastes, and social location. *Socio. Perspect* 42, 627–646. doi:10.2307/1389577
- Katz-Gerro, T. (2002). Highbrow cultural consumption and class distinction in Italy, Israel, west Germany, Sweden, and the United States. *Soc. Forces.* 81, 207–229. doi:10.1353/sof.2002.0050
- Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer* 42, 30–37. doi:10.1109/MC.2009.263
- Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.* 37, 233–243.
- Lai, C., Lee, S., and Huang, H. (2019). A social recommendation method based on the integration of social relationship and product popularity. *Int. J. Hum. Comput. Stud.* 121, 42–57. doi:10.1016/j.ijhcs.2018.04.002
- Lee, J. H., and Hu, X. (2014). Cross-cultural similarities and differences in music mood perception. *Manag. Sci.* 44, 249–269. doi:10.9776/14081
- Levine, L. W. (1988). *Highbrow/lowbrow: the emergence of cultural hierarchy in America*. Cambridge, MA: Harvard University Press.
- Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. (2018). “Variational autoencoders for collaborative filtering.” in Proceedings of the 2018 world wide web conference (republic and canton of Geneva, Switzerland), 689–698. doi:10.1145/3178876.3186150
- López-Sintas, J., Cebollada, À., Filimon, N., and Gharhaman, A. (2014). Music access patterns: a social interpretation. *Poetics* 46, 56–74. doi:10.1016/j.poetic.2014.09.003
- Moore, J. L., Joachims, T., and Turnbull, D. (2014). “Taste space versus the World: an Embedding analysis of listening habits and geography,” in Proceedings of the 15th International Society for music information retrieval conference (ISMIR 2014), Taipei, Taiwan.
- Morrison, SJ, and Demorest, SM (2009). Cultural constraints on music perception and cognition. *Prog. Brain Res.* 178, 67–77. doi:10.1016/S0079-6123(09)17805-6
- Nuccio, M., Guerzoni, M., and Katz-Gerro, T. (2018). Beyond class stratification: the rise of the eclectic music consumer in the modern age. *Cultural Sociology* 12, 343–367. doi:10.1177/1749975518786039
- Peterson, R. A., and Kern, R. M. (1996). Changing highbrow taste: from snob to omnivore. *Am. Socio. Rev.* 61, 900–907.
- Peterson, R. A., and Simkus, A. (1992). *How musical tastes mark occupational status groups cultivating differences: symbolic boundaries and the making of inequality*, 152. Chicago, IL: University of Chicago Press. 152–186.
- Pichl, M., Zangerle, E., Specht, G., and Schedl, M. (2017). “Mining culture-specific music listening behavior from social media data,” in IEEE international symposium on multimedia (ISM), 208–215. doi:10.1109/ISM.2017.35
- Rentfrow, P. J., and Gosling, S. D. (2003). The do re mi’s of everyday life: the structure and personality correlates of music preferences. *J. Pers. Soc. Psychol.* 84, 1236–1256. doi:10.1037/0022-3514.84.6.1236
- F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor (2015). *Recommender systems handbook*. 2nd edn. (Berlin: Springer).
- Rossmann, G., and Peterson, R. A. (2015). The instability of omnivorous cultural taste over time. *Poetics* 52, 139–153. doi:10.1016/j.poetic.2015.05.004
- Sánchez-Moreno, D., González, A. B. G., Vicente, M. D. M., Batista, V. F. L., and García, M. N. M. (2016). A collaborative filtering method for music recommendation using playing coefficients for artists and users. *Expert Syst. Appl.* 66, 234–244. doi:10.1016/j.eswa.2016.09.019
- Schäfer, T., and Mehlhorn, C. (2017). Can personality traits predict musical style preferences? a meta-analysis. *Pers. Individ. Differ.* 116, 265–273. doi:10.1016/j.paid.2017.04.061
- Schedl, M., and Bauer, C. (2017). “Online music listening culture of kids and adolescents: listening analysis and music recommendation tailored to the young.” in 1st International workshop on children and recommender systems, in conjunction with 11th ACM conference on recommender systems (RecSys ’17). KidRec.
- Schedl, M., Bauer, C., Reisinger, W., Kowald, D., and Lex, E. (2020). The dataset used in the article “listener modeling and context-aware music recommendation based on country archetypes”. *Zenodo*. doi:10.5281/zenodo.3907362
- Schedl, M. (2019). Deep learning in music recommendation systems. *Front. Appl. Math. Stat.* 5, 44. doi:10.3389/fams.2019.00044
- Schedl, M., and Ferwerda, B. (2017). Large-scale analysis of group-specific music genre taste from collaborative tags. *IEEE Int. Symp. Multimed. ISM.* 17, 479–482. doi:10.1109/ISM.2017.95
- Schedl, M. (2017). Investigating country-specific music preferences and music recommendation algorithms with the lfm-1b dataset. *Int. J. Multimed. Inf. Retr.* 6, 71–84. doi:10.1007/s13735-017-0118-y
- Schedl, M., Knees, P., McFee, B., Bogdanov, D., and Kaminskas, M. (2015). “Music recommender systems,” in *Recommender Systems Handbook*. Editors F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. 2nd edn. (Berlin: Springer), 453–492.
- Schedl, M., Lemmerich, F., Ferwerda, B., Skowron, M., and Knees, P. (2017). “Indicators of country similarity in terms of music taste, cultural, and socio-economic factors,” *IEEE Int. Symp. Multimed. ISM.* 19, 308–311. doi:10.1109/ISM.2017.55
- Schedl, M. (2016). “The LFM-1B dataset for music retrieval and recommendation,” in Proceedings of the 2016 ACM on international conference on multimedia retrieval, ’16. (New York, NY: ACM), 103–110. doi:10.1145/2911996.2912004
- Schedl, M., and Tkalcic, M. (2014). “Genre-based analysis of social media data on music listening behavior: are fans of classical music really averse to social media?,” in Proceedings of the first International workshop on Internet-scale multimedia management, WISMM ’14. Editors R. Zimmermann and Y. Yu (Orlando, FL: ACM), 9–13. doi:10.1145/2661714.2661717
- Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., and Elahi, M. (2018). Current challenges and visions in music recommender systems research. *Int. J. Multimed. Inform. Retrieval* 7, 95–116. doi:10.1007/s13735-018-0154-2
- Singhi, A., and Brown, D. G. (2014). “On cultural, textual and experiential aspects of music mood,” in Proceedings of the 15th international society for music information retrieval conference (Taipei, Taiwan), ISMIR, 3–8.

- Skowron, M., Lemmerich, F., Ferwerda, B., and Schedl, M. (2017). "Predicting genre preferences from cultural and socio-economic factors for music retrieval," in *Advances in information retrieval*. Editors J. M. Jose, C. Hauff, I. S. Altungovde, D. Song, D. Albakour, S. Watt, et al. (Cham, Germany: Springer International Publishing), 561–567.
- Sonnett, J. (2016). Ambivalence, indifference, distinction: a comparative netfield analysis of implicit musical boundaries. *Poetics* 54, 38–53. doi:10.1016/j.poetic.2015.09.002
- Statista Research Department. (2019). Music industry in the United Kingdom—statistics & facts. Available at: <https://www.statista.com/topics/3152/music-industry-in-the-united-kingdom-uk/>.
- Stevens, C. J. (2012). Music perception and cognition: a review of recent cross-cultural research. *Top Cogn. Sci.* 4, 653–667. doi:10.1111/j.1756-8765.2012.01215.x
- ter Bogt, T., Delsing, M., van Zalk, M., Christenson, P., and Meeus, W. (2011). Intergenerational continuity of taste: parental and adolescent music preferences. *Soc. Forces* 90, 297–319.
- Tiwari, S., Pangtey, M. S., and Kumar, S. (2018). "Location aware personalized news recommender system based on Twitter popularity," in *Computational science and its applications—ICCSA 2018*. Editors O. Gervasi, B. Murgante, S. Misra, E. Stankova, C. M. Torre, A. M. A. Rocha, et al. (Cham: Springer), 650–658.
- Vall, A., Quadrana, M., Schedl, M., and Widmer, G. (2019). Order, context and popularity bias in next-song recommendations. *Int. J. Multimed. Inform. Retriev.* 8, 101–113. doi:10.1007/s13735-019-00169-8
- van der Maaten, L., and Hinton, G. (2008). Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, 2579–2605.
- van Venrooij, A. (2009). The aesthetic discourse space of popular music: 1985–86 and 2004–05. *Poetics* 37, 315–332. doi:10.1016/j.poetic.2009.06.005
- Vigliensoni, G., and Fujinaga, I. (2016). "Automatic music recommendation systems: do demographic, profiling, and contextual features improve their performance?" in 17th International society for music information retrieval conference. ISMIR'16, 94–100.
- Vlegels, J., and Lievens, J. (2017). Music classification, genres, and taste patterns: a ground-up network analysis on the clustering of artist preferences. *Poetics* 60, 76–89. doi:10.1016/j.poetic.2016.08.004
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bull.* 1, 80–83.
- Zangerle, E., Pichl, M., and Schedl, M. (2018). "Culture-aware music recommendation," in Proceedings of the 26th conference on user modeling, adaptation and personalization (UMAP 2018). Singapore, Singapore: ACM, 357–358. doi:10.1145/3209219.3209258

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2021 Schedl, Bauer, Reisinger, Kowald and Lex. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.



On the Adaptability of Recurrent Neural Networks for Real-Time Jazz Improvisation Accompaniment

Kosmas Kritsis^{1,2*}, Theatina Kylafi³, Maximos Kaliakatsos-Papakostas², Aggelos Pikrakis¹ and Vassilis Katsouras²

¹Department of Informatics, University of Piraeus, Piraeus, Greece, ²Institute for Language and Speech Processing, Athena Research Center, Athens, Greece, ³Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece

OPEN ACCESS

Edited by:

Steven Kemper,
The State University of New Jersey,
United States

Reviewed by:

Rocco Zaccagnino,
University of Salerno, Italy
Tetsuro Kitahara,
Nihon University, Japan

*Correspondence:

Kosmas Kritsis
kosmas.kritsis@athenarc.gr

Specialty section:

This article was submitted to
Machine Learning and
Artificial Intelligence,
a section of the journal
Frontiers in Artificial Intelligence

Received: 30 October 2019

Accepted: 08 December 2020

Published: 12 February 2021

Citation:

Kritsis K, Kylafi T, Kaliakatsos-Papakostas M, Pikrakis A and Katsouras V (2021) On the Adaptability of Recurrent Neural Networks for Real-Time Jazz Improvisation Accompaniment. *Front. Artif. Intell.* 3:508727. doi: 10.3389/frai.2020.508727

Jazz improvisation on a given lead sheet with chords is an interesting scenario for studying the behaviour of artificial agents when they collaborate with humans. Specifically in jazz improvisation, the role of the accompanist is crucial for reflecting the harmonic and metric characteristics of a jazz standard, while identifying in real-time the intentions of the soloist and adapt the accompanying performance parameters accordingly. This paper presents a study on a basic implementation of an artificial jazz accompanist, which provides accompanying chord voicings to a human soloist that is conditioned by the soloing input and the harmonic and metric information provided in a lead sheet chart. The model of the artificial agent includes a separate model for predicting the intentions of the human soloist, towards providing proper accompaniment to the human performer in real-time. Simple implementations of Recurrent Neural Networks are employed both for modeling the predictions of the artificial agent and for modeling the expectations of human intention. A publicly available dataset is modified with a probabilistic refinement process for including all the necessary information for the task at hand and test-case compositions on two jazz standards show the ability of the system to comply with the harmonic constraints within the chart. Furthermore, the system is indicated to be able to provide varying output with different soloing conditions, while there is no significant sacrifice of “musicality” in generated music, as shown in subjective evaluations. Some important limitations that need to be addressed for obtaining more informative results on the potential of the examined approach are also discussed.

Keywords: automatic accompaniment system, music generative system, real-time music interaction, music improvisation, machine learning, long short-term memory

INTRODUCTION

The use of automatic systems for generating music is a captivating vision and a multidisciplinary research problem studied for decades. The diversity of music generative systems relies on their different objectives and the musical content that they produce, such as chord progressions, melody generation, accompaniment arrangements and counterpoints (Briot et al., 2019). Already from the late 1950s and early 1960s, composers such as Lejaren A. Hiller (Hiller Jr and Isaacson, 1957) and Iannis Xenakis (Xenakis, 1963) explored stochastic models for algorithmic music generation.

With the recent advances in the computational capabilities of modern computers, there is an exploding tendency of generative system proposals, incorporating complex artificial neural network architectures as a technical foundation. Conditional generative models based on Generative Adversarial Networks (GANs) have been used to combine unpaired lead sheet and MIDI datasets for generating lead sheet arrangements. The lead sheet arrangement can be defined as the process that receives a lead sheet as input and outputs piano-rolls of a number of instruments to accompany the melody of a given lead sheet. Liu and Yang (2018) proposed an architecture that comprises three stages (lead sheet generation, feature extraction and arrangement generation) in order to generate eight-bar phrases of lead sheets and their arrangement. The feature extraction stage is responsible to compute symbolic-domain harmonic features from the given lead sheet in order to condition the generation of the arrangement. Wang and Xia (2018) developed a framework for generating both lead melody and piano accompaniment arrangements of pop music. Specifically, they consider a chord progression as input and propose three phases for generating a structured melody with layered piano accompaniments. First, the harmony alternation model receives a given chord progression in order to transform it to a different one that fits better with a specified music style based on Hidden Markov Models (HMMs). Then, the melody generation model generates the lead melody and the layered accompaniment voices through seasonal ARMA (Autoregressive Moving Average) processes. The final phase implements the melody integration model which is responsible for integrating the melody voices together as the final piano accompaniment.

On the other hand, Recurrent Neural Networks (RNNs) are often used to generate sequences of musical content in a stepwise manner, where the network input is the previous note and output is considered the predicted note to occur on the following time interval (Mozier, 1994). In a similar manner, RNNs with Long Short-Term Memory (LSTM) cells have been utilized for generating blues style melodies conditioned by a given chord progression (Eck and Schmidhuber, 2002). By definition, LSTM-based models have the ability to correlate and capture the temporal context of a sequence, thus simulating the human cognitive abilities for predicting sequential information. Also, RNNs have proven efficacy on modelling complex musical structures such as polyphonic chorales. For instance, the DeepBach system was trained to generate four-part chorales in the style of J. S. Bach (Hadjeres et al., 2017). Generative systems can be also constrained by music theory rules via a reinforcement learning mechanism as it is demonstrated by Jaques et al. (2017). In addition to the music theory rules, Boulanger-Lewandowski et al. (2012) employed probabilistic harmonic and rhythmic rules, based on distribution estimators conditioned by a RNN that is trained to discover temporal dependencies from polyphonic music scores of varying complexity.

Other approaches take into account the chord progressions for providing longer musical structures. For instance, in the work of Choi et al. (2016), a text-based LSTM network is employed for capturing the relationships within text documents that contain symbols of chord progressions. Another example based on chord

progressions is the JamBot system (Brunner et al., 2017) that generates music in two steps. The bottom network is a LSTM architecture that predicts a chord progression based on a chord “embedding,” while a second LSTM generates polyphonic music based on the predicted chord progression received from the bottom network. Nevertheless, this approach lacks the ability of modeling interactions within a polyphonic musical ensemble. In order to overcome this limitation, Chu et al. (2016) proposed a hierarchical architecture, where each level is a RNN that generates different accompaniments for the song. A monophonic melody is generated first, followed by the accompanying chords and drums.

In the scope of the Impro-Visor (Jazz Improvisation Advisor)¹ project, Johnson et al. (2017) proposed a neural network architecture consisting of two LSTM-based sub-networks that jointly learn to predict a probability distribution over future notes conditioned on past notes in the melody. Additionally, researchers from the same laboratory developed the JazzGAN system (Trieu and Keller, 2018) that utilizes RNN-based GANs to improvise monophonic jazz melodies over given chord progressions. Their results indicated that the proposed system was capable to address frequent and diverse key changes, as well as unconventional and off-beat rhythms, while providing flexibility with off-chord notes. Other proposals incorporate music theory grammar in combination with LSTM neural networks to generate jazz music. For instance, Wang et al. (2019) extracted the interval, duration and note category information from jazz MIDI files and trained a LSTM model to learn the transition probabilities between notes. Then they take advantage of the music grammar in order to arrange and output the generated sequence of notes.

LSTM networks have been also tested for generating jazz music compositions constrained by a given performer’s style. In particular, De Prisco et al. (2017) developed a three staged generative system, consisting of a One-Class Support Vector Machine (OCSVM) for learning the performing style of a specific jazz musician, an LSTM network to generate patterns relevant to the learned style and a splicing system to compose melodic lines in the given style. Splicing systems are formal models for generating languages (sets of words), inspired by a recombinant behavior of DNA (De Felice et al., 2015). A music splicing composer requires to define an alphabet, an initial set and a set of rules. Another example of a complex system that utilizes LSTM networks for learning statistical correlations between instruments within a jazz piano trio ensemble (piano, bass, drums) was proposed by Hori et al. (2017). They trained a LSTM architecture to learn the relationship between the musical features of the piano performance that is applied on top of a Hidden Markov Model (HMM), which is responsible to segment the bass and drums performance feature spaces. Overall the system is capable to generate coherent rhythmic patterns and bass melodies as accompaniments to a piano solo input. However the authors specify that their model can be further improved due to the lack of available jazz datasets. To this regard, Hung et al.

¹<https://www.cs.hmc.edu/~keller/jazz/improvisor/> – last accessed February 1st, 2020.

(2019) employed transfer learning techniques aiming to solve the problem of jazz data insufficiency. They proposed a Bidirectional Gated Recurrent Unit (BGRU) Variational Autoencoder (VAE) generative model trained on a dataset of unspecified genres as source and a Jazz-only dataset as target.

It is worth noting that only a few projects experiment with real-time creative scenarios where a human improviser is accompanied by an automatic agent without any musical constraints. To this end, Kaliakatsos-Papakostas et al. (2012) proposed an accompaniment system that employs Differential Evolution and Genetic Algorithms for producing the accompanying music. Another approach to real-time music generation for jazz improvisation that was proposed by Hutchings and McCormack (2017), implements a composite system with an LSTM-based melody agent, which was trained on chord progressions of jazz “standard” compositions and a rule-based harmony agent that manipulates precomposed melodies for improvising new themes and variations. The composition flow between the agents is controlled by a rating system that rewards harmonic consistency and melodic continuity.

Aim of this paper is to examine the characteristics of musical accompaniment that an artificial agent can provide in real-time to a human improviser, in a setting similar to typical forms of jazz improvisation, i.e. under the constraints of previously agreed upon harmonic sequence and metrical structure. Software tools and methods that are able to generate “static” accompaniment to human soloists, exist for a long time (Ferguson, 2005). The paradigm discussed in this paper includes “spontaneous” alterations in accompaniment responses of an artificial agent both in terms of rhythm and harmony, based on the improvisation of a human soloist. The algorithmic cornerstone of the examined approach relies on LSTM RNNs architectures. The motivation for pursuing and studying such an approach in modeling human-machine improvisation and the reasons for choosing to examine basic deep learning neural networks as an algorithmic backbone is analysed in the following section.

MOTIVATION, RESEARCH QUESTIONS AND CONTRIBUTION

In music, “masterful” violation of anticipation has been identified as key component for the emergence of emotion, meaning, concepts and overall interest (Huron, 2006). Furthermore, anticipation is shaped by the exposure to stimuli with common characteristics, a fact that induces relations between fundamental mechanisms of music understanding and statistical learning (Huron, 2006). The basic principles of jazz improvisation evolve around the violation of expectation, with improvising musicians constantly attempting to introduce meaningful novelty in the way they express themselves and communicate with other musicians in real-time. Therefore, jazz improvisation could be described as an exemplar for studying the core-mechanism of music cognition: interplay between anticipation and violation thereof.

Communication between improvising musicians is a key-point for achieving interesting and meaningful improvisations. In jazz improvisation, specifically, the role of each musician is manifold; the most prominent characteristics of the role of each musician, according to how they relate with the study at hand, can be summarised as follows:

1. *Preserve harmonic and rhythmic characteristics of a piece.* Typical jazz improvisation incorporates a standard jazz melody with a fixed harmonic description in a fixed metric structure. These components, however, are expected to be creatively altered by improvising musicians (usually not the metric structure though), towards creating meaningful violations of anticipation on the overall harmonic and rhythmic domain. For instance, chord substitutions are usual, either by introducing chords that include alternate voicings, extensions or even by including new chords altogether (e.g. tritone substitution).
2. *Express original ideas.* Violation of harmonic/rhythmic expectations is expected to come “with a reason.” A common approach for soloists to attempt to build new musical phrases when improvising, is by creatively modifying and combining “standard” jazz licks, a fact that helps towards building and violating anticipation. Jazz licks in the (muscle) memory of the soloist are products of statistical learning, built through practicing and listening multiple jazz pieces, excerpts and phrases.
3. *Communicate musically with the improvisation/accompaniment of other musicians.* In a broad sense, the role of the accompanist is to highlight musical choices of the soloist, or, even further, understand the intentions of the soloist and improvise accompaniments accordingly. Therefore, communication, on the side of the accompanist, includes predicting the intentions of the soloist and preparing the response in a timely manner, given that proper accompaniment needs to be provided concurrently with the solo. Jazz musicians, as musicians in any other field, develop a common perception that, in the examined case, can be described as the integration of a “similar” statistical model both in the soloist and the accompanist; this model allows the accompanist to roughly predict the imminent soloist choices during improvisation.

To this end, an artificial agent that is able to perform *basic* musical accompaniment in real-time under the aforementioned setting needs to have: 1) the ability to comply with harmonic and metrical constraints set by an input chart; 2) a model of anticipating for imminent actions of the human soloist; 3) a dictionary of accompanying voicings for given chords that is rich enough for producing diverse/interesting accompaniment; and 4) the ability to “adapt” its playing style (both in terms of voicings and rhythm) to the anticipated choices of the human soloist. Since the problem description incorporates statistical learning and given the fact that deep neural networks have exhibited impressive capabilities in capturing the prominent statistical behaviour in large amounts of training data, this study examines the incorporation of such machine learning tools for

the task at hand. Therefore, the research questions revolve around the suitability of deep learning methods for the described improvisation setting, under the methodological framework that is presented in *Materials and Methods*. These questions are formulated as follows:

- A. Is the presented framework able to capture “static” harmonic information of a given chart in a setting of dynamic constraints (changing playing style of the human soloist)?
- B. To what extent is the proposed system responding to dynamic components introduced by the human agent?
- C. Is the examined setup suitable for real-time performance, both in terms of robustness and computational feasibility?

Recent advances in deep learning include the development of systems that are able to generate music that adapts to pre-configured constraints. In general terms, such systems either compose music *sequentially* or *non-sequentially*. In sequential systems (e.g. as the one presented by Makris et al. (2019)), the decision for each note depends only on previous notes, with additional potential constraints. In non-sequential systems (e.g., as Deep Bach; Hadjeres et al. (2017)), new notes are inserted by sampling, forming “dynamic” constraints for notes that are inserted later on, regardless of time precedence – i.e. notes at the end of the piece could be inserted at an earlier stage than notes earlier in the piece, depending on randomly sampled priorities. In one sense, a system that is able to perform real-time accompaniment, as described in the presented study, needs to be able to both compose sequentially (since time moves forward while performing) and comply with constraints that change as the composition is constructed (since the human soloist is expected to violate the expectations reflected by the solo predictive model).

The main contribution of this paper is that it studies the characteristics of a complex, multi-layered neural network where both static and dynamic components are combined for preforming predictions. The real-time improvisation setup discussed herein offers a well-defined platform of experimentation with potential interest for real-world applicability and clearly defined research questions.

MATERIALS AND METHODS

The proposed system provides real-time accompaniment to a human musician, based on a given harmonic description of lead sheet chord symbols. The role of the system is to reflect harmonic information as given in the lead sheet and also interpret this information with variability, responding to the predicted implied harmonic variability of a human solo. To this end, data need to include information about: 1) metric structure, for letting the system become aware of measure changes; 2) lead sheet information, for learning to comply with given lead sheet chords; 3) a human solo channel, for learning to respond to what the human soloist is expected to play; and 4) an accompaniment channel, for learning to play proper accompaniment chords/voicings over the given lead sheet chords. Up to our knowledge, such a dataset containing all the

mentioned properties is missing from the research community. To this end, *Data Preparation* describes the processes for constructing a dataset by starting off with an initial dataset collected from online resources that covers most of the requirements. Afterwards, we present the proposed system that incorporates two layers of information processing: the first for predicting the imminent steps of the human performance and the second for integrating this prediction along with other static constraints (i.e. metric and lead sheet information) for making the final chord accompaniment prediction.

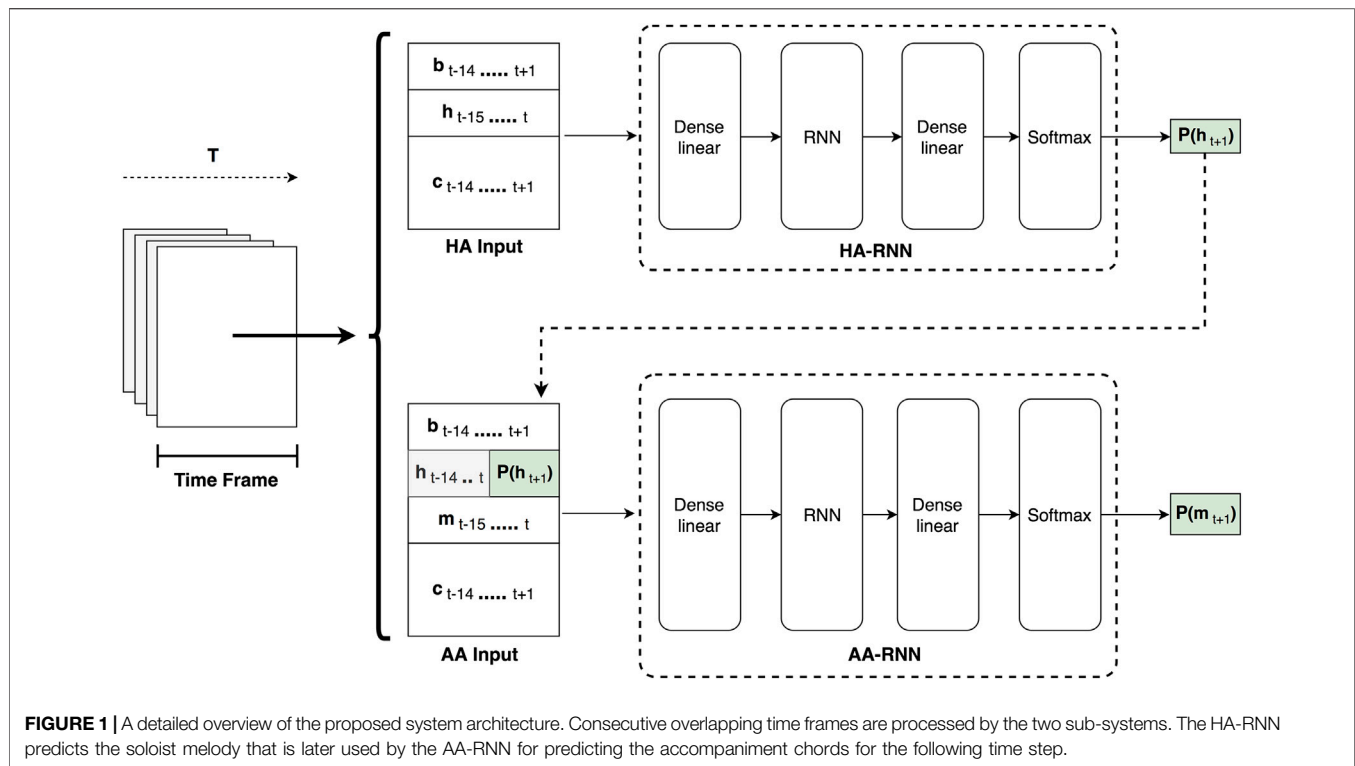
Data Preparation

The initial dataset² (Liu and Yang, 2018) contains all necessary information about the pieces, including tempo, beat, melody and the chords on a lead sheet. It should be noted that only lead sheet information is included in this dataset without actual notation of the accompaniment chords. In order to address this issue we performed a harmonic enrichment procedure that is described in detail later in this section. Furthermore, the beat information indicates the start of a measure. A single time step corresponds to the 1/24 of a quarter note, a time resolution which is fine enough to even represent rhythm values of sixty-fourth triples. The melody and the accompanying chords are represented as 128-key piano rolls with the aforementioned time resolution, where each active note at each time instance is annotated with the respective velocity value. With this representation however, the information about a note repetition is potentially obscured. For instance, there is no differentiation between a single note/chord of a quarter duration (24 time steps) and two successive notes/chords with a duration of an eighth per note (12 time steps). A time resolution reduction from 24 steps per beat (quarter) to two steps per beat was performed, such that each time step was represented by 1/2 of a quarter note, which is an eighth note. In other words, from each beat (24 time steps) we only kept the melodic information of the first and 13th time step, by splitting each quarter (24 time steps) in half. Thus keeping only the first of each of the two subsets of time instances (12 time steps).

In order to construct a suitable and compact representation of chord information in the form of a jazz standard lead sheet, we use the information extracted from the accompanying chords channel of the initial dataset. Specifically, instead of keeping the velocity values of the chord notes and their MIDI numbers, we only kept the pitch class of their root, as well as the type of those chords, by using ready-made functions from the MIT Music21 *Python* library³, which contains a set of functions for computer-aided musicology. Moreover, we chose to represent the jazz standard chord information as a binary vector of size 15, where the first 12 bits represent the root pitch class information, while the remaining 3 bits represent major/minor third, perfect/augmented/diminished fifth and major/minor seventh respectively. The reason for performing such an abstraction for representing chord information on the lead sheet is

²<https://github.com/wayne391/lead-sheet-dataset/> – last accessed February 1st, 2020.

³<https://web.mit.edu/music21> – last accessed February 1st, 2020.



motivated by the fact that jazz musicians need a fundamental description of harmony, which they can manipulate/alter in a creative manner. The employed scheme allows for basic chord types to be represented, e.g., major/minor triads, dominant seventh, major seventh, (half) diminished and augmented.

As mentioned earlier, the initially obtained dataset includes information only about lead sheet chords, without specific notation of actual accompanying chords. Hence we constructed the actual accompaniment chords algorithmically by applying a basic “harmonic enrichment” process, where the lead sheet chords are transcribed into actual accompanying chords with different inversions and diverse rhythmic patterns. The enrichment process begins by assigning accompaniment chords to positions of lead sheet chord symbols. After that, inverted chords are probabilistically inserted after the initially placed chords. The probability of chord insertion at a specific position on the score depends on the time passed without a chord event (the more the time, the higher the probability) and whether there is a melodic note event (melody notes increase the probability for chord insertion). Aim of this process is to introduce rudimentary variability in the accompaniment channel, based on the lead sheet chord symbols and the melodic rhythm.

Since the melody channel is monophonic, the 128-sized vector representation of each note in the melody channel is flattened to its single non-zero value (the actual MIDI number of that note). For the accompaniment channel, i.e. the actual notes that the system is intended to learn, a dictionary of all the unique chords in the training set is created and each chord is represented by its index in the dictionary. Practically, the

“flattened” values for both the melody and the accompaniment parts allow us to apply one-hot representation of the respective data streams. Before the harmonic enrichment process, the initial dictionary of the accompaniment chords incorporated 476 chord classes, while after the augmentation and before the transposition to all the possible 12 pitches we had 847 classes. Finally, after all the data preparation procedure, including the augmentation and transposition processes, we ended up having 2677 unique accompanying chord classes.

System Architecture and Real-time Considerations

As it is already mentioned, the generated accompaniment part should be related to the soloist’s intentions on the future melody notes to be played. To this regard, the proposed system architecture depicted in **Figure 1**, consists of two sub-systems, namely the Human Agent RNN (HA-RNN) and the Artificial Agent RNN (AA-RNN), that rely on the effectiveness of the LSTM recurrent neural network (RNN) for modeling sequential information.

The overall system receives as input successively overlapping windows comprising 16 time steps, representing events within a time resolution of eighth notes. The window slides one step/eighth note at each iteration, which occurs in every eighth successively. Information for each time step includes:

- The metric information (b_t).
- The soloist’s melodic/solo part (h_t).

TABLE 1 | System interpretations of chart chords for “All of Me” without solo (top) and with random (bottom) solo at epoch 59, shown as pitch class sets.

| No solo Chart chord | System interpretations | | | | |
|----------------------------|------------------------|---------------------|--------------------|----------------------|--------------------|
| [0, 4, 7, 11] | [0, 4, 7, 11] (80) | [0, 3, 8] (4) | [2, 7, 10] (12) | [0, 2, 4, 5, 7] (24) | |
| [0, 4, 7, 10] | [0, 5, 7, 10] (1) | [2, 6, 11] (1) | [2, 4, 6, 8] (4) | [2, 4, 8, 11] (1) | [0, 4, 7, 10] (25) |
| [2, 4, 8, 11] | [2, 6, 9, 11] (1) | [2, 4, 8, 11] (185) | [0, 4, 7] (4) | [0, 4, 7, 10] (18) | |
| [1, 4, 7, 9] | [1, 4, 7, 9] (168) | [2, 5, 9] (4) | [2, 7, 10] (4) | | |
| [2, 5, 9] | [2, 5, 9] (128) | | | | |
| [0, 4, 9] | [0, 4, 9] (64) | | | | |
| [0, 2, 6, 9] | [0, 2, 6, 9] (64) | | | | |
| [0, 2, 5, 9] | [1, 4, 7, 9] (8) | [0, 2, 5, 9] (72) | | | |
| [2, 5, 7, 11] | [2, 5, 7, 11] (79) | | | | |
| [0, 5, 9] | [0, 5, 9] (32) | | | | |
| [0, 5, 8] | [0, 5, 9] (4) | [0, 5, 8] (28) | | | |
| Random solo Chart chord | System interpretations | | | | |
| [0, 4, 7, 11] | [4, 6, 8, 10, 11] (1) | [4, 6, 8, 11] (3) | [2, 6, 11] (3) | [0, 4, 7, 11] (80) | |
| | | [0, 3, 8] (4) | [2, 7, 10] (12) | [0, 2, 4, 5, 7] (24) | |
| [0, 4, 7, 10] | [0, 4, 7, 10] (32) | | | | |
| [2, 4, 8, 11] | [0, 4, 7] (4) | [2, 4, 8, 11] (186) | [0, 4, 7, 10] (18) | | |
| [1, 4, 7, 9] | [1, 4, 7, 9] (168) | [2, 5, 9] (4) | [2, 7, 10] (4) | | |
| [2, 5, 9] | [2, 5, 9] (128) | | | | |
| [0, 4, 9] | [0, 4, 9] (64) | | | | |
| [0, 2, 6, 9] | [0, 2, 6, 9] (64) | | | | |
| [0, 2, 5, 9] | [1, 4, 7, 9] (8) | [0, 2, 5, 9] (72) | | | |
| [2, 5, 7, 11] | [2, 5, 7, 11] (79) | | | | |
| [0, 5, 9] | [0, 5, 9] (32) | | | | |
| [0, 5, 8] | [2, 5, 9] (4) | [0, 5, 8] (28) | | | |

Numbers in parentheses show the total time steps that a system-generated PC-set occurs under the respective chart PC-set.

- The accompaniment chords that are expected to be learned from the system (m_i).
- The chord information in the abstract lead sheet style described in *Data Preparation* (c_i).

Since the HA-RNN is responsible for predicting the solo melody of the following time step (h_{t+1}), it excludes the accompaniment channel from its input, while having the beat and chord information channels one eighth ahead from the current melody. On the other hand, the AA-RNN takes under account all the information channels, in addition to the predicted $P(h_{t+1})$ of the HA-RNN, in order to anticipate the accompaniment chord for the future eighth (m_{t+1}). Both agents at their core, implement a similar neural network architecture. Firstly, the input time frame is processed by the bottom “Dense linear” (fully connected) layer, where it gets embedded to a fixed size dimension through a linear transformation. Next, the embedded output is further encoded into a latent space through the LSTM RNN layer. Then, the top “Dense linear” layer receives the encoded LSTM output and applies a linear transform to a space with a dimensionality equal to the number of the target classes. Finally the output of the top fully connected layer passes through a softmax function, resulting to a probability distribution for the target classes ($P(h)$ and $P(m)$). The final prediction is the class with the highest probability.

As a proof-of-concept, we trained a basic system with batches of 128 samples. The embedding dimension of the bottom fully connected layer was equal to the size of the feature dimension of the input frame, whilst the RNN layer contained 64 LSTM cells. We used the Adam optimisation algorithm (Kingma and Ba, 2014) for

the minimization of the cross entropy cost function with a learning rate of 0.001. Both the HA-RNN and AA-RNN architectures were implemented using the TensorFlow 2.0 framework (Abadi et al., 2016) and trained for at least 1,200 epochs on a computer equipped with the NVIDIA Tesla K40c GPU, an Intel Core i7-5820K CPU at 3.30 GHz and 32 GB DDR4 RAM at 2133 Mhz. With the aforementioned experimental setup, we observed that the average time of the overall system to predict an accompaniment chord was around 0.66 ms (0.31 ms for the HA in addition to 0.35 ms for the AA). This fact indicates the feasibility of the proposed system to be adopted in real-time applications, however a thorough evaluation of the real-time capabilities of the presented method needs to be examined as future work. In this regard, we developed a prototype web application based on MIDI.js and Tensorflow.js javascript libraries for testing the adaptability of the proposed model to the user’s soloing input in a real-time setting. The model implementation and training code of the LSTM models, as well as the real-time web interface are hosted on a GitHub repository⁴. Since the project continuously evolves, the online repository will be updated with future developments and improvements.

RESULTS

The results are oriented towards answering the research questions given in *Motivation, Research Questions and Contribution*, i.e. whether and to what extent is the system

⁴<https://github.com/kosmasK/JazzICat>.

TABLE 2 | System interpretations of chart chords for “All of Me” without solo (top) and with random (bottom) solo, shown as pitch class sets.

| No solo | | System interpretations | | | |
|---------------|--|--|---|--|---|
| Chart chord | | | | | |
| [0, 4, 7, 11] | [3, 5, 8, 11] (2) [0, 3, 8] (1) | [0, 2, 9, 10] (1) [2, 5, 10] (1) | [2, 4, 9] (1) [0, 4, 7, 11] (103) | [1, 3, 10, 11] (1) [3, 7, 10] (13) | [1, 3, 6, 10] (1) [0, 5, 9] (3) |
| [0, 4, 7, 10] | [0, 3, 8] (2) | [2, 5, 10] (1) | [0, 4, 7, 10] (23) | [2, 5, 7, 11] (3) | [2, 7, 11] (3) |
| [2, 4, 8, 11] | [2, 4, 8, 11] (162) [1, 4, 9] (1) [0, 2, 5, 9] (3) | [4, 8, 11] (1) [1, 4, 7, 9] (12) [0, 2, 6, 9] (3) | [4, 7, 11] (4) [1, 4, 6, 10] (3) | [0, 3, 5, 9] (1) [2, 6, 8, 11] (3) | [3, 6, 8, 11] (1) [2, 5, 9] (6) |
| [1, 4, 7, 9] | [1, 4, 7, 9] (124) [2, 6, 8, 11] (4) | [2, 4, 7, 11] (9) [5, 8, 11] (3) | [1, 2, 6, 9] (11) [2, 8, 11] (3) | [2, 4, 8, 11] (7) | [1, 3, 10, 11] (4) |
| [2, 5, 9] | [2, 5, 7, 9] (15) [3, 7, 10] (6) | [2, 7, 10] (4) | [2, 5, 9] (94) | [1, 3, 10, 11] (3) | [1, 5, 8, 10] (3) |
| [0, 4, 9] | [0, 4, 9] (12) [2, 5, 9] (3) | [3, 6, 10] (1) | [1, 6, 9] (1) | [4, 8, 11] (1) | [0, 2, 4, 9] (45) |
| [0, 2, 6, 9] | [0, 2, 6, 9] (54) [2, 5, 10] (1) | [0, 4, 9] (5) | [0, 3, 6, 10] (1) | [0, 5, 8] (1) | [0, 3, 5, 8] (1) |
| [0, 2, 5, 9] | [2, 5, 10] (13) [0, 5, 8] (12) | [0, 2, 5, 9] (36) [2, 4, 7, 11] (4) | [0, 3, 7, 10] (2) [2, 5, 7, 10] (4) | [3, 7, 10] (1) | [1, 3, 4, 11] (8) |
| [2, 5, 7, 11] | [2, 5, 7, 11] (65) | [2, 3, 7, 10] (4) | [0, 3, 7] (4) | [0, 5, 8] (3) | [2, 5, 9, 10] (3) |
| [0, 5, 9] | [0, 4, 5, 9] (28) | [2, 5, 9] (4) | | | |
| [0, 5, 8] | [0, 5, 8] (28) | [3, 7, 10] (4) | | | |
| Random solo | | System interpretations | | | |
| Chart chord | | | | | |
| [0, 4, 7, 11] | [3, 5, 8, 11] (1) [2, 7, 10] (1) [0, 4, 7] (2) [0, 3, 6, 8] (1) | [1, 3, 6, 8] (2) [0, 5, 9] (5) [2, 4, 5, 9] (1) [1, 4, 9] (2) | [2, 4, 9] (1) [0, 4, 7, 11] (86) [1, 5, 8, 11] (1) [2, 5, 9, 10] (1) | [0, 4, 5, 9] (1) [3, 7, 10] (5) [2, 5, 10] (1) | [1, 3, 6, 10] (1) [0, 2, 4, 5, 7] (14) [0, 2, 6, 9] (1) |
| [0, 4, 7, 10] | [0, 4, 7, 10] (28) | [1, 5, 8] (1) | [0, 4, 5, 9] (1) | [2, 7, 11] (2) | |
| [2, 4, 8, 11] | [2, 4, 8, 11] (177) [1, 6, 9] (2) | [1, 4, 7, 9] (12) [2, 4, 5, 9] (1) | [4, 8, 11] (2) [3, 6, 9, 11] (1) | [4, 7, 11] (3) [2, 4, 7, 11] (1) | [1, 4, 8, 11] (2) |
| [1, 4, 7, 9] | [1, 4, 7, 9] (140) [0, 2, 6, 9] (6) | [1, 2, 6, 9] (5) [2, 4, 7, 11] (4) | [2, 4, 8, 11] (5) [2, 4, 5, 9] (1) | [1, 4, 6, 10] (4) | [1, 6, 8, 11] (5) |
| [2, 5, 9] | [2, 5, 9] (95) | [2, 5, 7, 9] (30) | [2, 7, 10] (2) | [0, 4, 7] (1) | |
| [0, 4, 9] | [0, 2, 4, 9] (30) [4, 8, 11] (1) | [2, 5, 9] (2) | [0, 4, 9] (25) | [1, 3, 10, 11] (2) | [2, 4, 6, 7] (2) |
| [0, 2, 6, 9] | [0, 2, 6, 9] (46) [2, 7, 11] (2) | [0, 4, 9] (8) | [1, 3, 10, 11] (2) | [2, 5, 7, 10] (2) | [0, 4, 7, 9] (2) |
| [0, 2, 5, 9] | [0, 2, 5, 9] (49) [2, 5, 7, 10] (6) [0, 5, 8] (1) | [1, 3, 10, 11] (3) [3, 7, 10] (3) [0, 2, 5, 7] (1) | [0, 2, 5, 8] (2) [2, 6, 9] (2) | [0, 4, 7] (6) [0, 3, 5, 8] (2) | [0, 1, 5, 8] (2) [2, 5, 10] (2) |
| [2, 5, 7, 11] | [2, 5, 7, 11] (62) [5, 8, 11] (1) | [2, 5, 9, 10] (2) [0, 4, 7, 10] (1) | [2, 5, 10] (8) [0, 2, 5, 9] (1) | [2, 3, 7, 10] (1) [0, 4, 7] (1) | [2, 7, 11] (1) |
| [0, 5, 9] | [0, 4, 5, 9] (28) | | | | |
| [0, 5, 8] | [0, 5, 8] (28) | [3, 7, 10] (4) | | | |

Numbers in parentheses show the total time steps that a system-generated PC-set occurs under the respective chart PC-set.

able to capture the harmonic lead sheet constraints, to what extent is the system influenced by different soloing styles and what are possible limitations for applying this approach in real-time settings with current technologies. To this end, two test jazz standards, “All of Me” and “Au Privave” are examined in different and diverse artificial improvisation settings, that simulate two extreme scenarios: where the human player 1) is not playing any note during the solo (consecutive occurrences of pause events) and 2) is playing random notes within two octaves (as a form of extremely complex improvisation). The responses of the system under these two settings for each piece are analysed for different epochs of training (randomly sampled across all training epochs), providing insights about how harmonic compliance is varied and how the existence of a solo

affects system responses (adaptability) at different stages of training. Since technical limitations led to building a system with limited computational power (incorporating solely a single LSTM layer with few neurons for the artificial agent) and keeping time resolution to eight notes, getting useful feedback from musicians through exhaustive real-time experiments was not possible. In this regard, a preliminary empirical evaluation based on listening tests was conducted by comparing generated and original accompaniments. We maintain, however, that the results presented herein indicate that employing more sophisticated architectures for (at least) the part of the artificial agent would lead to a system that both adapts to the playing style of the user and preserves harmonic consistency according to the given lead sheet.

Compliance with Lead Sheet Harmony

This section examines the ability of the system to play chords that correspond to the chord symbols on the lead sheet chart. This part of the study concerns the compliance with the basic harmonic guidelines provided by the chart and, therefore, comparison is presented on the level of pitch class sets (PC-sets). To this end, the lead sheet chart chords are translated to their corresponding pitch classes as well as the interpretations of the system. To obtain insight on how training epochs influence the harmonic compliance of the system, results are taken from an early and a late epoch of training (59 and 1,251). **Table 1** (epoch 59) and **Table 2** (epoch 1,251) show the chord symbols and the responses of the system in “All of me” when no solo (top) and random solo (bottom) was provided; similarly, **Tables 3, 4** show the responses of the system in “Au Privave” without and with random solo.

Regrading “All of Me”, **Table 1** shows that in most cases the exact harmonic description in the lead sheet chart is reflected by the system. Initially, it should be noted that harmonic deviations mostly concern the first few starting measures of each piece, where the system has not incorporated any memory in its decisions. The beginning chord of the chart, [0, 4, 7, 11], appears to have the most alterations, some of which are clearly erroneous (e.g. the [4, 6, 8, 10, 11] interpretation that was composed for random solo). **Figure 2A** shows the first eight measures and **Figure 2B** measures 33 to 40, composed by the system for “All of Me” in a real-time simulation setting with random solo (the solo part is not shown). The “erroneous” choices appear to be artefacts of the initial delay of the system to catch up with the constraints and start building up harmonic memory; **Figure 2A** shows that the first three chord shown in the lower part of **Table 1** are a result of this delay. Other harmonic deviations concern the delay of the system in complying with “unexpected” chord changes – given that most pieces in the dataset are pop songs. For instance, some misinterpretations of the E7 chord ([2, 4, 8, 11]) are a result of delay in “comprehending” the unexpected change; this is shown in the third bars of both (A) and (B) parts of **Figure 2**. System-generated chords for “Au Privave” follow a similar pattern in terms of harmonic compliance but with fewer erroneous harmonic deviations, as evident in **Table 3**.

Variability

The chords generated by the system in each improvisation setting for each piece are expected to be different, since different improvisations from the human soloist should trigger different responses. Those differences are examined by direct comparison of the system generated chords for the two improvisation modes, i.e. the chords generated by the system without human solo and with a random solo. A general figure that describes the differences between the system-generated chords in both examined pieces with (random) and without solo, is given by computing the percentage of chords that are different per time step for accompaniment sessions comprising four repetitions of the entire chart, with (random) and without solo. In “All of Me” only 2% of system-generated chords are different between random and no solo for epoch 59, which jumps to 60% for

TABLE 3 | System interpretations of chart chords for “Au Privave” without solo (top) and with random (bottom) solo at epoch 59, shown as pitch class sets.

| No solo | | System interpretations | | | |
|---------------|--------------------|------------------------|--------------------|-------------------|--|
| Chart chord | | | | | |
| [0, 5, 9] | [2, 5, 10] (6) | [0, 5, 9] (61) | [0, 5, 9, 10] (12) | | |
| [2, 5, 7, 10] | [2, 5, 7, 10] (60) | [0, 4, 7, 9] (36) | [2, 5, 9] (16) | | |
| [0, 4, 7, 10] | [0, 4, 7, 10] (35) | [2, 5, 8, 10] (4) | [1, 3, 7, 10] (4) | [0, 2, 6, 9] (4) | |
| [0, 3, 7, 10] | [0, 3, 7, 10] (16) | | | | |
| [1, 3, 5, 9] | [0, 3, 5, 9] (1) | [0, 3, 6, 8] (2) | [5, 8, 11] (9) | | |
| [2, 5, 8, 10] | [2, 5, 10] (1) | [2, 5, 8, 10] (28) | [3, 6, 10, 11] (3) | | |
| [1, 5, 8, 10] | [1, 5, 8, 10] (16) | | | | |
| [1, 3, 7, 10] | [2, 5, 8, 10] (16) | | | | |
| [0, 4, 7, 9] | [2, 5, 7, 10] (8) | [0, 2, 5, 9] (4) | [0, 4, 7, 9] (4) | | |
| [0, 2, 6, 9] | [0, 4, 5, 9] (12) | [2, 5, 9, 10] (4) | [0, 2, 5, 9] (4) | [0, 2, 6, 9] (12) | |
| Random solo | | System interpretations | | | |
| Chart chord | | | | | |
| [0, 5, 9] | [2, 5, 10] (3) | [0, 5, 9] (73) | [0, 5, 9, 10] (3) | | |
| [2, 5, 7, 10] | [0, 5, 9] (1) | [2, 5, 7, 10] (83) | [0, 4, 7, 9] (12) | [2, 5, 9] (16) | |
| [0, 4, 7, 10] | [0, 4, 7, 10] (46) | [0, 2, 6, 9] (1) | | | |
| [0, 3, 7, 10] | [0, 3, 7, 10] (16) | | | | |
| [1, 3, 5, 9] | [0, 3, 5, 9] (5) | [0, 3, 6, 8] (6) | [2, 6, 9, 11] (1) | [2, 5, 9, 11] (2) | |
| [2, 5, 8, 10] | [2, 5, 10] (1) | [2, 5, 8, 10] (31) | | | |
| [1, 5, 8, 10] | [1, 5, 8, 10] (16) | | | | |
| [1, 3, 7, 10] | [2, 5, 8, 10] (4) | [1, 3, 7, 10] (12) | | | |
| [0, 4, 7, 9] | [0, 4, 7, 9] (16) | | | | |
| [0, 2, 6, 9] | [0, 2, 5, 9] (2) | [0, 2, 6, 9] (30) | | | |

Numbers in parentheses show the total time steps that a system-generated PC-set occurs under the respective chart PC-set.

epoch 1,251, showing that the system decisions are affected slightly by the presence of a solo in early epochs, while the effect of solo is more evident as epochs progress. In “Au Privave” this percentage starts from 74% during epoch 59 and jumps to 84% at epoch 1,251, showing that system generations are more sensitive to the presence of a chord solo for this piece.

For observing the differences within each improvisation session, the system-generated chords in four repetitions of the entire chart are examined repetition-by-repetition – forming four quarters of the entire composition, referred to as “quartiles”. **Tables 5–8** show the quartile similarities for “All of Me” (epochs 59 and 1,251) and “Au Privave” (epochs 59 and 1,251) respectively, without (left) and with random solo (right). In “All of Me” and with an absence of solo, both in the early and the late epoch of training only the first repetition is different from the remaining three, as show in the first rows and columns of both matrices in **Tables 5, 7**. The insertion of the random solo does not influence the overall result in the early epoch (right matrix in **Table 5**), but for the late epoch the influence is evident (right matrix in **Table 7**). Therefore, the example of “All of Me” shows that training the system for more epochs allows some sense of responsiveness to human input, as evident by the variability that emerged from the random solo. In “Au Privave”, on the other hand, the incorporation of the random solo (**Table 6**) influences each repetition even from early training epochs, therefore creating different variations of the chart in each of the four iterations (except repetition three and four that differ only by 1%); variations for this test piece are even more evident in the more progressed training epoch (**Table 8**).

TABLE 4 | System interpretations of chart chords for “Au Privave” without solo (top) and with random (bottom) solo at epoch 1,251, shown as pitch class sets.

| No solo | | System interpretations | | | |
|---------------|--|--|--|--|---|
| Chart chord | | | | | |
| [0, 5, 9] | [1, 3, 10, 11] (5) [1, 3, 7, 10] (1) [5, 8, 11] (4) | [0, 3, 5, 9] (5) [0, 5, 8] (1) [2, 7, 10] (4) | [0, 3, 7, 8] (1) [0, 3, 8] (1) [0, 5, 7] (3) | [3, 6, 11] (1) [2, 7, 11] (1) | [2, 6, 8, 11] (1) [0, 5, 9] (51) |
| [2, 5, 7, 10] | [2, 5, 8, 10] (2) [1, 3, 10, 11] (4) [2, 5, 10] (4) | [2, 5, 7, 10] (59) [1, 4, 6, 9] (4) [0, 2, 5, 7] (12) | [0, 1, 5, 8] (1) [0, 4, 7, 10] (10) [0, 2, 5, 9] (3) | [2, 3, 7, 10] (1) [1, 4, 7, 9] (4) [0, 2, 7, 10] (3) | [0, 5, 8] (1) [0, 3, 7] (4) |
| [0, 4, 7, 10] | [0, 3, 7] (16) | [2, 5, 10] (1) | [1, 3, 6, 10] (1) | [0, 4, 7, 10] (24) | |
| [0, 3, 7, 10] | [0, 3, 7, 10] (16) | | | | |
| [1, 3, 5, 9] | [0, 3, 5, 9] (1) | [0, 1, 5, 8] (8) | [1, 3, 7, 10] (3) | | |
| [2, 5, 8, 10] | [3, 5, 8, 11] (4) [0, 5, 8] (4) | [0, 3, 7] (7) [2, 5, 9] (3) | [2, 7, 10] (3) | [2, 5, 8, 10] (4) | [2, 5, 10] (7) |
| [1, 5, 8, 10] | [3, 5, 7, 10] (10) | [1, 3, 7, 10] (6) | | | |
| [1, 3, 7, 10] | [1, 3, 7, 10] (2) | [0, 3, 8] (4) | [0, 5, 8] (4) | [3, 7, 10] (6) | |
| [0, 4, 7, 9] | [0, 4, 7, 9] (16) | | | | |
| [0, 2, 6, 9] | [0, 2, 6, 9] (12) | [0, 3, 7] (8) | [0, 5, 8] (4) | [0, 5, 7, 8] (4) | |
| Random solo | | System interpretations | | | |
| Chart chord | | | | | |
| [0, 5, 9] | [1, 3, 10, 11] (4) [1, 5, 8] (1) [2, 5, 8, 10] (5) [0, 3, 5, 9] (3) | [2, 5, 7, 10] (5) [3, 5, 7, 10] (2) [0, 3, 7] (2) [3, 5, 8, 11] (2) | [3, 5, 8, 10] (1) [2, 5, 10] (2) [0, 5, 9] (17) [0, 4, 7, 10] (2) | [1, 3, 7, 10] (1) [3, 6, 11] (6) [3, 7, 10] (4) [0, 5, 8] (3) | [1, 3, 5, 8] (1) [0, 4, 5, 9] (11) [4, 8, 11] (3) [0, 3, 8] (1) |
| [2, 5, 7, 10] | [1, 3, 7, 10] (2) [5, 6, 8, 11] (2) [1, 5, 8, 10] (1) [0, 5, 8, 10] (1) [0, 2, 3, 5, 10] (1) | [0, 3, 7] (1) [0, 3, 5, 9] (2) [2, 3, 7, 10] (3) [1, 4, 9] (1) [3, 6, 8, 11] (1) | [0, 3, 8, 10] (1) [2, 5, 9] (7) [1, 4, 9, 11] (2) [0, 5, 9] (1) [0, 2, 5, 9] (1) | [2, 6, 8, 11] (1) [1, 3, 10, 11] (4) [1, 4, 6, 10] (3) [2, 5, 10] (4) [0, 3, 7, 8] (2) | [2, 5, 7, 10] (57) [1, 4, 6, 9] (3) [0, 5, 7, 9, 10] (1) [5, 8, 11] (1) [0, 5, 8] (1) |
| [0, 4, 7, 10] | [2, 7, 11] (1) [0, 3, 7] (6) [2, 5, 10] (3) [0, 3, 7, 8] (2) | [3, 5, 7, 8] (1) [1, 4, 6, 9] (1) | [0, 5, 8] (2) [1, 3, 6, 10] (1) | [0, 4, 7, 10] (18) [3, 6, 8, 11] (1) | [0, 3, 8] (1) [2, 7, 10] (1) |
| [0, 3, 7, 10] | [0, 3, 7, 10] (11) | [1, 4, 9, 11] (1) | [1, 3, 10, 11] (1) | [3, 7, 10] (2) | [0, 1, 5, 8] (1) |
| [1, 3, 5, 9] | [0, 3, 5, 9] (5) [0, 5, 8] (1) | [1, 4, 6, 10] (1) [0, 3, 7] (3) | [2, 6, 8, 11] (1) | [0, 3, 5, 8] (1) | [1, 3, 7, 10] (1) |
| [2, 5, 8, 10] | [2, 7, 10] (1) [2, 5, 10] (3) | [2, 5, 8, 10] (19) [2, 5, 7, 10] (2) | [0, 5, 8] (5) | [1, 3, 7, 10] (1) | [3, 7, 10] (1) |
| [1, 5, 8, 10] | [1, 3, 6, 10] (1) [2, 5, 8, 10] (1) | [1, 3, 5, 10] (3) | [1, 3, 7, 10] (2) | [3, 5, 7, 10] (3) | [1, 5, 8, 10] (6) |
| [1, 3, 7, 10] | [1, 3, 7, 10] (8) | [0, 5, 8] (1) | [3, 6, 10] (1) | [3, 7, 10] (4) | [0, 3, 8] (1) |
| [0, 4, 7, 9] | [0, 4, 7, 9] (12) | [0, 2, 3, 5, 10] (1) | [2, 5, 7, 10] (1) | [1, 3, 5, 6, 8] (1) | |
| [0, 2, 6, 9] | [0, 2, 6, 9] (10) [2, 5, 7, 11] (1) | [0, 3, 5, 9] (2) [0, 1, 3, 8] (1) | [0, 3, 7] (5) | [0, 5, 8] (3) | [0, 2, 5, 9] (3) |

Numbers in parentheses show the total time steps that a system-generated PC-set occurs under the respective chart PC-set.

A final examination of variability in the generated chords is performed by measuring the number of different voicings per chord symbol on the chart. This is a more detailed examination of how the PC-sets presented in **Tables 1–4** are further split down in voicing layouts, i.e. what is the variability in terms of inversions and note doublings in the chords generated by the system. **Figure 3** shows the average number of different voicings composed by the system for each chord label in the chart, in form of errorbars for some random epochs sampled accross all training epochs. In “All of Me” (left image), each chord symbol in the chart is materialised with approximately 2.5 different voicing implementations in epoch 59, almost regardless of the presence of solo (red “x” indicates presence of random melody and blue circle absence thereof). The system presents increased voicing variability dependence on human solo input for this piece as the epochs increase. In

the case of “Au Privave,” the tendency of the system to become more dependent on human input becomes more evident as epochs increase. The error value of the objective function in a validation set during training is shown in **Figure 4**. The typical decrease that is observed indicates that there is a relation between error loss and system adaptability to human input, i.e., better training leads to further variability.

Listening Tests

The dataset used to train the artificial agent, contains a broad variety of popular western music melodies with simulated accompaniments derived from an augmentation process. Furthermore, the quantitative metrics presented in the previous subsections are not capable to completely capture the perceptual quality and originality of the chord accompaniments generated by the proposed system. To this end, we carried out a

Figure 2 displays two musical staves, A and B, representing the first eight measures and measures 33–40 of a system-generated accompaniment for the song "All of Me". The notation includes a random solo part (treble clef) and a random solo part (bass clef). The chords are indicated below the bass line: Cmaj7, C7, E7, E7, A7, A7, Dm, Dm.

FIGURE 2 | First eight measures (A) and measures 33–40 (B) of system-generated chords over the respective lead sheet chords for "All of Me" with random solo part (omitted in the depiction).

TABLE 5 | "Quartile" similarity in system-generated chords in "All of Me" without (left) and with random solo (right) at epoch 59.

| No solo | | | | |
|---------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.09 | 0.09 | 0.09 |
| 2nd qrt | 0.09 | 0.00 | 0.00 | 0.00 |
| 3rd qrt | 0.09 | 0.00 | 0.00 | 0.00 |
| 4th qrt | 0.09 | 0.00 | 0.00 | 0.00 |

| Random solo | | | | |
|-------------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.05 | 0.05 | 0.06 |
| 2nd qrt | 0.05 | 0.00 | 0.00 | 0.00 |
| 3rd qrt | 0.05 | 0.00 | 0.00 | 0.00 |
| 4th qrt | 0.06 | 0.00 | 0.00 | 0.00 |

TABLE 6 | "Quartile" similarity in system-generated chords in "Au Privave" without (left) and with random solo (right) at epoch 59.

| No solo | | | | |
|---------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.34 | 0.34 | 0.35 |
| 2nd qrt | 0.34 | 0.00 | 0.00 | 0.01 |
| 3rd qrt | 0.34 | 0.00 | 0.00 | 0.01 |
| 4th qrt | 0.35 | 0.01 | 0.01 | 0.00 |

| Random solo | | | | |
|-------------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 1.00 | 0.99 | 0.99 |
| 2nd qrt | 1.00 | 0.00 | 0.33 | 0.34 |
| 3rd qrt | 0.99 | 0.33 | 0.00 | 0.01 |
| 4th qrt | 0.99 | 0.34 | 0.01 | 0.00 |

subjective evaluation based on listening tests, aiming to study whether the generated accompaniments are comparable to the original chords existing in the dataset.

TABLE 7 | "Quartile" similarity in system-generated chords in "All of Me" without (left) and with random solo (right) at epoch 1,251.

| No solo | | | | |
|---------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.53 | 0.53 | 0.54 |
| 2nd qrt | 0.53 | 0.00 | 0.00 | 0.00 |
| 3rd qrt | 0.53 | 0.00 | 0.00 | 0.00 |
| 4th qrt | 0.54 | 0.00 | 0.00 | 0.00 |

| Random solo | | | | |
|-------------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.51 | 0.72 | 0.19 |
| 2nd qrt | 0.51 | 0.00 | 0.30 | 0.55 |
| 3rd qrt | 0.72 | 0.30 | 0.00 | 0.72 |
| 4th qrt | 0.19 | 0.55 | 0.72 | 0.00 |

TABLE 8 | "Quartile" similarity in system-generated chords in "Au Privave" without (left) and with random solo (right) at epoch 1,251.

| No solo | | | | |
|---------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.40 | 0.40 | 0.41 |
| 2nd qrt | 0.40 | 0.00 | 0.00 | 0.01 |
| 3rd qrt | 0.40 | 0.00 | 0.00 | 0.01 |
| 4th qrt | 0.41 | 0.01 | 0.01 | 0.00 |

| Random solo | | | | |
|-------------|---------|---------|---------|---------|
| | 1st qrt | 2nd qrt | 3rd qrt | 4th qrt |
| 1st qrt | 0.00 | 0.65 | 0.96 | 0.70 |
| 2nd qrt | 0.65 | 0.00 | 0.91 | 0.73 |
| 3rd qrt | 0.96 | 0.91 | 0.00 | 0.83 |
| 4th qrt | 0.70 | 0.73 | 0.83 | 0.00 |

For preparing the listening tests we randomly selected 10 solo melodies along with their original accompaniments from the validation set. Then we used the 10 selected melodic parts to

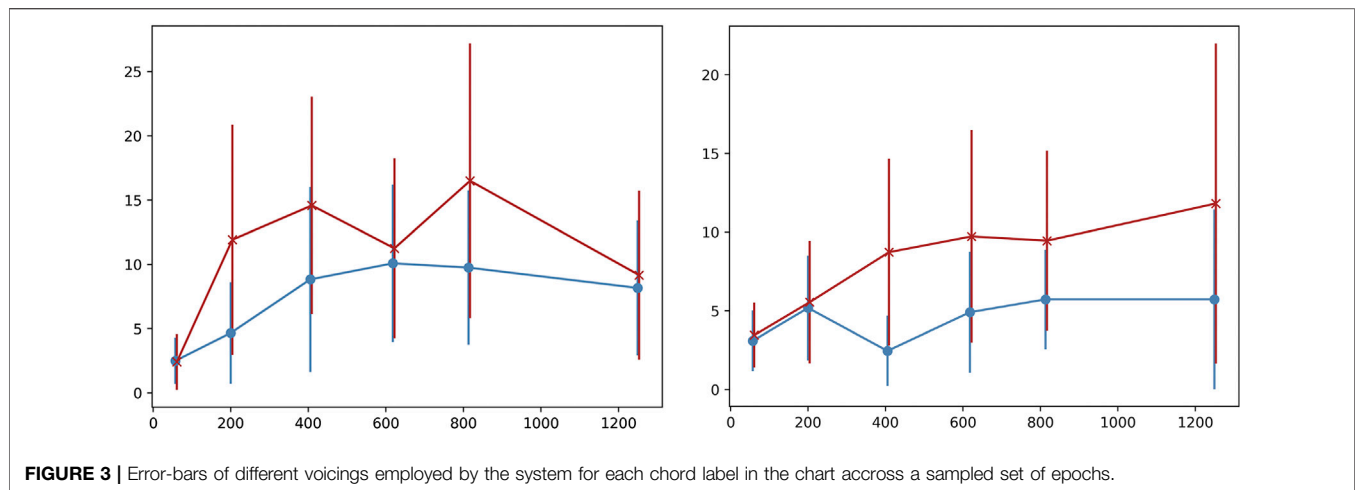


FIGURE 3 | Error-bars of different voicings employed by the system for each chord label in the chart across a sampled set of epochs.

generate their corresponding chord accompaniments with the proposed artificial agent, thus ensuring that the system receives novel input, not wielded during training. Accordingly, each participant was presented with 10 tests; each test included three audio clips, starting only with the melodic part and followed by its combinations with the two accompaniments (original and generated), which are introduced in a random order so as to avoid any possible biases. The actual audio excerpts had a duration of around 30 s and looped for six times to reach 3 min. Then, the participants had to answer the following three questions for each accompaniment (six questions per test) in a Likert scale from 1 (low) to 5 (high):

- Q1: Evaluate the overall high-level structure of the accompaniment with respect to the introduced melody.
- Q2: Evaluate the harmonic compliance of the accompaniment with reference to popular western music.
- Q2: Evaluate the rhythmical compliance of the accompaniment with reference to popular western music.

In our study 21 participants were involved, 15 male and six female, with the majority being 20–40 years old. All of the

participants were musicians with different levels of expertise, having at least intermediate knowledge of music theory. Consequently, we collected a total of 1,260 answers and the results are presented in **Table 9**. By inspecting only the mean values we can observe that the participants evaluated slightly better the original accompaniments in most questions. However in order to determine whether this preference is statistically important, we performed a Wilcoxon rank sum test, having as null hypothesis that there is no difference between the two accompaniments. The calculated p -values demonstrated that there is statistically significant difference between the original and the generated accompaniments in examples 5, 6, 7, 9 and 10 (highlighted with bold fonts in **Table 9**), while we cannot reject the null hypothesis for the remaining examples. In other words, in 50% of the examples, we cannot be certain about whether the generated music is inferior to the original, as far as the examined qualities can define.

Overall, we can say that the accompaniments generated by the proposed artificial agent had better rhythmical compliance rather than harmonic, which might be due to the metric information that is included in the system input. Also, the poor performance in some examples indicates that the computational capabilities of a single LSTM layer are limited, thus suggesting more sophisticated architectures to be tested. We strongly encourage the reader to visit the online repository and listen to the audio files of the listening tests.

CONCLUSION

The paper at hand presented a study on how deep neural network architectures can be employed for simulating a jazz improvisation setting between a human soloist and an artificial accompanist, based on a common chord chart. A basic implementation incorporating deep neural networks was presented and publicly available data were transformed in a way that all necessary information for the task at hand became available, i.e. information about metric structure, lead sheet chords, human-generated solo/melody and system-generated accompaniment responses. The motivation of this work is based on modeling the interplay between expectation and its violation by two improvising musicians (one human and one artificial) with implicit

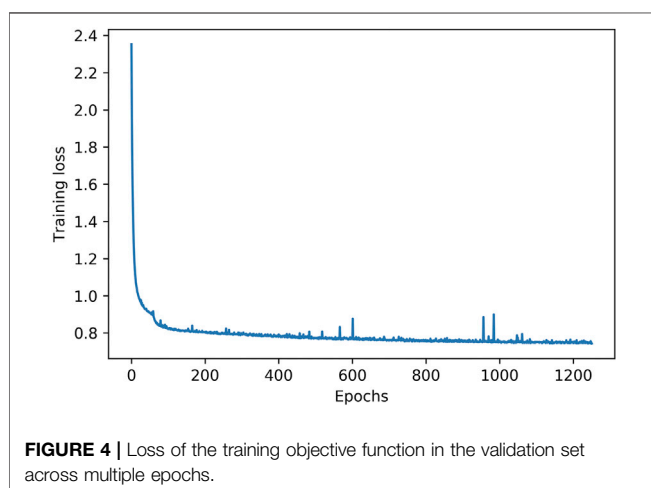


FIGURE 4 | Loss of the training objective function in the validation set across multiple epochs.

TABLE 9 | Results of our listening tests.

| | Question | p-value | Accompaniment | Median | Mean | | Question | p-value | Accompaniment | Median | Mean |
|-----------|----------|----------------|---------------|--------|------|------------|----------|----------------|---------------|--------|------|
| Example 1 | Q1 | 0.48121 | Original | 4.0 | 3.57 | Example 6 | Q1 | 0.04689 | Original | 4.0 | 3.71 |
| | | | Generated | 3.0 | 3.33 | | | | Generated | 3.0 | 2.9 |
| | Q2 | 0.35198 | Original | 3.0 | 3.43 | | Q2 | 0.00025 | Original | 4.0 | 3.95 |
| | | | Generated | 4.0 | 3.71 | | | | Generated | 2.0 | 2.48 |
| Example 2 | Q3 | 0.6966 | Original | 4.0 | 3.95 | Example 7 | Q3 | 0.00826 | Original | 4.0 | 3.86 |
| | | | Generated | 4.0 | 3.86 | | | | Generated | 3.0 | 2.86 |
| | Q1 | 0.30236 | Original | 4.0 | 3.57 | | Q1 | 0.00007 | Original | 4.0 | 4.19 |
| | | | Generated | 3.0 | 3.24 | | | | Generated | 2.0 | 2.43 |
| Example 3 | Q2 | 0.44293 | Original | 4.0 | 3.52 | Example 8 | Q2 | 0.0 | Original | 4.0 | 4.19 |
| | | | Generated | 3.0 | 3.24 | | | | Generated | 1.0 | 1.62 |
| | Q3 | 0.66891 | Original | 4.0 | 3.52 | | Q3 | 0.00005 | Original | 5.0 | 4.33 |
| | | | Generated | 4.0 | 3.33 | | | | Generated | 2.0 | 2.43 |
| Example 4 | Q1 | 0.26296 | Original | 4.0 | 3.67 | Example 9 | Q1 | 0.48907 | Original | 4.0 | 3.95 |
| | | | Generated | 3.0 | 3.29 | | | | Generated | 4.0 | 3.86 |
| | Q2 | 0.08951 | Original | 4.0 | 3.86 | | Q2 | 0.1159 | Original | 4.0 | 4.24 |
| | | | Generated | 3.0 | 3.24 | | | | Generated | 4.0 | 3.76 |
| Example 5 | Q3 | 0.95987 | Original | 4.0 | 3.71 | Example 10 | Q3 | 0.88003 | Original | 4.0 | 3.9 |
| | | | Generated | 4.0 | 3.71 | | | | Generated | 4.0 | 4.0 |
| | Q1 | 0.32656 | Original | 4.0 | 3.86 | | Q1 | 0.03353 | Original | 4.0 | 3.76 |
| | | | Generated | 4.0 | 3.52 | | | | Generated | 3.0 | 3.0 |
| Example 6 | Q2 | 0.15523 | Original | 4.0 | 3.86 | Example 11 | Q2 | 0.00001 | Original | 5.0 | 4.24 |
| | | | Generated | 3.0 | 3.38 | | | | Generated | 2.0 | 2.24 |
| | Q3 | 0.41361 | Original | 4.0 | 3.86 | | Q3 | 0.00037 | Original | 4.0 | 4.1 |
| | | | Generated | 4.0 | 3.52 | | | | Generated | 2.0 | 2.57 |
| Example 7 | Q1 | 0.0543 | Original | 4.0 | 3.71 | Example 12 | Q1 | 0.00153 | Original | 4.0 | 3.95 |
| | | | Generated | 3.0 | 2.9 | | | | Generated | 2.0 | 2.76 |
| | Q2 | 0.00022 | Original | 4.0 | 3.76 | | Q2 | 0.0014 | Original | 4.0 | 3.71 |
| | | | Generated | 2.0 | 2.19 | | | | Generated | 2.0 | 2.29 |
| Example 8 | Q3 | 0.00766 | Original | 4.0 | 4.0 | Example 13 | Q3 | 0.02863 | Original | 4.0 | 3.52 |
| | | | Generated | 3.0 | 2.86 | | | | Generated | 2.0 | 2.62 |

The bold fonts indicate the statistically significant differences provided by a Wilcoxon rank sum test between the original and the generated accompaniments.

machine learning approaches (deep neural networks) and the methodology included the development of “a model within a model”, that allows the artificial agent to have its own model of expectation for the human improviser. Additional challenges included the adaptation of large amounts of data to the desired form, leading to the development of a data enrichment process that generated variability in the accompaniment parts of the collected pieces.

Results were obtained by testing the system in two real-time simulation settings: without any assumed human solo and with the inclusion of a random solo. The responses of the system under these two settings in two well-known jazz standards (“All of me” and “Au Privave”) indicated that harmonic compliance with the chart chords was mainly achieved, except mainly from the beginning of each accompaniment session where the system needs to “collect memory” for starting performing better; this is possibly due to the random initialisation of the states in the LSTM networks that are in the core of the presented basic implementation. Even though it was expected for the system to be influenced by the incorporation of a human solo, this was not the case in both examined pieces. Specifically, in “All of Me” the inclusion of a random solo did not appear to affect the output of the system, while the system-generated chords exhibited self-repetition in accompaniment sessions incorporating four iterations of the chart. Conversely, in “Au Privave” the inclusion of the random solo affected the system output both by decreasing self-repetition in four iterations and by increasing the number of chord voicings employed by the system for given chart chords. In order to

evaluate the perceptual quality of the generated chords, we also performed a subjective evaluation based on listening tests, where participants had to compare original and generated accompaniments given their corresponding melodies, by ranking their harmonic and rhythmical compliance in a liker scale. A Wilcoxon rank sum test on the responses showed that 50% of the examples were not significantly inferior to the original accompaniments.

Future research is necessary for a more thorough examination of such system for real-time accompaniment. The results presented herein indicate that it is possible to model expectation and violation thereof for real-time jazz accompaniment with deep neural networks, however, severe limitations have to be acknowledged for performing further studies:

1. There is no proper data available with all the necessary information (lead sheet chords, metric information, solo and accompaniment). A crucial part of the data, i.e. the accompaniment, was actually constructed algorithmically while the solo part included melodies (rather than solos) with restricted expressional variability. The data enrichment method that was developed to construct artificial variability in the data was based on a rudimentary probabilistic implementation which is not enough for creating consistent connections that could be learned from the system.
2. The execution time of predictions might be marginally acceptable for scalable real-time systems. For the presented

study, time resolution was significantly reduced for making the system safely compatible with real-time conditions, however, this fact reduced the expressional capabilities of the system. This includes not only restricted capabilities for the system responses, but also restricted capabilities for the system to identify expressional characteristics of the human soloist.

3. The prominent style found in the dataset was pop, which comprises smaller harmonic variability in comparison to jazz. Therefore, the resulting accompaniment had to be creatively adjusted for more reflecting complex jazz lead sheet progressions. A consistent dataset of jazz standard accompaniment sessions is necessary for studying this problem more deeply.

DATA AVAILABILITY STATEMENT

The datasets [generated/analyzed] for this study can be found in the Zenodo open-access repository on the link <https://zenodo.org/record/3523222>. The training code of the LSTM models, the real-time web interface and the audio files of the listening tests can be found on GitHub following the link <https://github.com/kosmasK/JazzICat>.

REFERENCES

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). "Tensorflow: a system for large-scale machine learning." in Proceedings of the 12th Symposium on Operating Systems Design and Implementation (OSDI 2016), Savannah, GA, USA, November 2–4, 2016. 265–283.
- Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. *arXiv [Preprint]*. Available at: <https://arxiv.org/abs/1206.6392>.
- Briot, J.-P., Hadjeres, G., and Pachet, F. (2019). *Deep learning techniques for music generation*. New York, NY: Springer International Publishing.
- Brunner, G., Wang, Y., Wattenhofer, R., and Wiesendanger, J. (2017). "Jambot: music theory aware chord based generation of polyphonic music with lstms." in IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI 2017), Boston, MA, USA, Jul 15, 2017. 519–526.
- Choi, K., Fazekas, G., and Sandler, M. (2016). Text-based lstm networks for automatic music composition. *arXiv [Preprint]*. Available at: <https://arxiv.org/abs/1604.05358>.
- Chu, H., Urtasun, R., and Fidler, S. (2016). Song from pi: a musically plausible network for pop music generation. *arXiv [Preprint]*. Available at: <https://arxiv.org/abs/1611.03477>.
- De Felice, C., De Prisco, R., Malandrino, D., Zaccagnino, G., Zaccagnino, R., and Zizza, R. (2015). "Chorale music splicing system: an algorithmic music composer inspired by molecular splicing." in *Evolutionary and biologically inspired music, sound, art and design*. Editors C. Johnson, A. Carballal, and J. Correia, (New York, NY: Springer International Publishing, 50–61.
- De Prisco, R., Malandrino, D., Zaccagnino, G., Zaccagnino, R., and Zizza, R. (2017). "A kind of bio-inspired learning of music style." *Computational intelligence in music, sound, art and design*. Editors J. Correia, V. Ciesielski, and A. Liapis (New York, NY: Springer International Publishing), 97–113.
- Eck, D., and Schmidhuber, J. (2002). "Finding temporal structure in music: blues improvisation with lstm recurrent networks." in Proceedings of the 12th IEEE workshop on neural networks for signal processing (IEEE), Martigny, Switzerland, September 6, 2002, 747–756.

AUTHOR CONTRIBUTIONS

KK, MK-P, and TK prepared the dataset, designed and implemented the system, performed the experiments, analysed the results and wrote the paper of this study. AP and VK supervised the study at hand.

FUNDING

This work was supported by the project "Computational Sciences and Technologies for Data, Content and Interaction" (MIS 5002437), implemented under the Action "Reinforcement of the Research and Innovation Infrastructure," funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014–2020) and co-financed by Greece and EU (ERDF).

ACKNOWLEDGMENTS

We would like to thank all of our colleagues at the Athena Research Centre, University of Piraeus and National and Kapodistrian University of Athens, for their valuable support.

- Ferguson, L. (2005). Band-in-a-box for the general music classroom. *Gen. Music Today* 18, 7–13.
- Hadjeres, G., Pachet, F., and Nielsen, F. (2017). "Deepbach: a steerable model for bach chorales generation." in 34th international conference on machine learning, Sydney, Australia, August 6–11, 2017, Vol 70, 1362–1371.
- Hiller, L. A., Jr, and Isaacson, L. M. (1957). "Musical composition with a high speed digital computer." in *Audio engineering society convention*, (New York, NY: Audio Engineering Society), 9, 154–160.
- Hori, T., Nakamura, K., and Sagayama, S. (2017). "Jazz piano trio synthesizing system based on hmm and dnn." in Proceedings of the 14th sound and music computing conference 2017. Banff, Canada, October 5–8, 2017, 153.
- Hung, H.-T., Wang, C.-Y., Yang, Y.-H., and Wang, H.-M. (2019). Improving automatic jazz melody generation by transfer learning techniques. *arXiv [Preprint]*. Available at: <https://arxiv.org/abs/1908.09484>.
- Huron, D. B. (2006). *Sweet anticipation: music and the psychology of expectation*, Cambridge, MA: MIT press.
- Hutchings, P., and McCormack, J. (2017). "Using autonomous agents to improvise music compositions in real-time," in *Computational intelligence in music, sound, art and design*. Editors J. Correia, V. Ciesielski, and A. Liapis (New York, NY: Springer International Publishing), 114–127.
- Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. (2017). "Sequence tutor: conservative fine-tuning of sequence generation models with kl-control." in 34th international conference on machine learning, Sydney, Australia, August 6–11, 2017, 70, 1645–1654.
- Johnson, D. D., Keller, R. M., and Weintraub, N. (2017). "Learning to create jazz melodies using a product of experts" in 8th international conference on computational creativity, Atlanta, GA, ICCO, 151–158.
- Kaliakatos-Papakostas, M. A., Floros, A., and Vrahatis, M. N. (2012). "Intelligent real-time music accompaniment for constraint-free improvisation" in IEEE 24th international conference on tools with artificial, November 7–9, 2012, Athens, Greece, 1, 444–451.
- Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv [Preprint]*. Available at: <https://arxiv.org/abs/1412.6980>.
- Liu, H.-M., and Yang, Y.-H. (2018). "Lead sheet generation and arrangement by conditional generative adversarial network." in 17th IEEE international conference on machine learning and applications, December 17–20, 2018, Orlando, FL, 722–727.

- Makris, D., Kaliakatsos-Papakostas, M., Karydis, I., and Kermanidis, K. L. (2019). Conditional neural sequence learners for generating drums' rhythms. *Neural Comput. Appl.* 31, 1793–1804. doi:10.1007/s00521-018-3708-6
- Mozier, M. C. (1994). Neural network music composition by prediction: exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connect. Sci.* 6, 247–280. doi:10.1080/09540099408915726
- Trieu, N., and Keller, R. M. (2018). Jazzgan: improvising with generative adversarial networks."in 6th international workshop on musical metacreation (MUME 2018), June 25–26, 2018, Salamanca, Spain, 1–8.
- Wang, J., Wang, X., and Cai, J. (2019). "Jazz music generation based on grammar and lstm."in 11th international conference on intelligent human-machine systems and cybernetics, August 24–25, 2019, Hangzhou, China, 115–120.
- Wang, Z., and Xia, G. (2018). "A framework for automated pop-song melody generation with piano accompaniment arrangement." in 6th conference on sound and music technology, Xiamen, China, 1–9.
- Xenakis, I. (1963). *Formalized music: thought and mathematics in composition*. Bloomington, IN, Indiana University Press.
- Conflict of Interest:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2021 Kritsis, Kylaifi, Kaliakatsos-Papakostas, Pikrakis and Katsouros. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Advantages of publishing in Frontiers



OPEN ACCESS

Articles are free to read
for greatest visibility
and readership



FAST PUBLICATION

Around 90 days
from submission
to decision



HIGH QUALITY PEER-REVIEW

Rigorous, collaborative,
and constructive
peer-review



TRANSPARENT PEER-REVIEW

Editors and reviewers
acknowledged by name
on published articles

Frontiers

Avenue du Tribunal-Fédéral 34
1005 Lausanne | Switzerland

Visit us: www.frontiersin.org

Contact us: frontiersin.org/about/contact



REPRODUCIBILITY OF RESEARCH

Support open data
and methods to enhance
research reproducibility



DIGITAL PUBLISHING

Articles designed
for optimal readership
across devices



FOLLOW US

@frontiersin



IMPACT METRICS

Advanced article metrics
track visibility across
digital media



EXTENSIVE PROMOTION

Marketing
and promotion
of impactful research



LOOP RESEARCH NETWORK

Our network
increases your
article's readership