NEUROMORPHIC ENGINEERING EDITORS' PICK 2021

EDITED BY: André van Schaik and Bernabe Linares-Barranco
PUBLISHED IN: Frontiers in Neuroscience

frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-88971-161-1
DOI 10.3389/978-2-88971-161-1

NEUROMORPHIC ENGINEERING EDITORS' PICK 2021

Topic Editors:
André van Schaik, Western Sydney University, Australia
Bernabe Linares-Barranco, Instituto de Microelectrónica de Sevilla, Spain

Citation: van Schaik, A., Linares-Barranco, B., eds. (2021). Neuromorphic Engineering Editors' Pick 2021. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-88971-161-1

Table of Contents

ORIGINAL RESEARCH
published: 05 February 2019
doi: 10.3389/fnins.2019.00031

Information-Theoretic Intrinsic Plasticity for Online Unsupervised Learning in Spiking Neural Networks

*Wenrui Zhang and Peng Li**

Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, United States

As a self-adaptive mechanism, intrinsic plasticity (IP) plays an essential role in maintaining homeostasis and shaping the dynamics of neural circuits. From a computational point of view, IP has the potential to enable promising non-Hebbian learning in artificial neural networks. While IP based learning has been attempted for spiking neuron models, the existing IP rules are *ad hoc* in nature, and the practical success of their application has not been demonstrated particularly toward enabling real-life learning tasks. This work aims to address the theoretical and practical limitations of the existing works by proposing a new IP rule named SpiKL-IP. SpiKL-IP is developed based on a rigorous information-theoretic approach where the target of IP tuning is to maximize the entropy of the output firing rate distribution of each spiking neuron. This goal is achieved by tuning the output firing rate distribution toward a targeted optimal exponential distribution. Operating on a proposed firing-rate transfer function, SpiKL-IP adapts the intrinsic parameters of a spiking neuron while minimizing the KL-divergence from the targeted exponential distribution to the actual output firing rate distribution. SpiKL-IP can robustly operate in an online manner under complex inputs and network settings. Simulation studies demonstrate that the application of SpiKL-IP to individual neurons in isolation or as part of a larger spiking neural network robustly produces the desired exponential distribution. The evaluation of SpiKL-IP under real-world speech and image classification tasks shows that SpiKL-IP noticeably outperforms two existing IP rules and can significantly boost recognition accuracy by up to more than 16%.

Keywords: intrinsic plasticity, spiking neural networks, unsupervised learning, liquid state machine, speech recognition, image classification

OPEN ACCESS

Edited by:
André van Schaik,
Western Sydney University, Australia

Reviewed by:
Paul Miller,
Brandeis University, United States
Priyadarshini Panda,
Purdue University, United States

****Correspondence:***
Peng Li
pli@tamu.edu

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *20 August 2018*
Accepted: *14 January 2019*
Published: *05 February 2019*

Citation:
Zhang W and Li P (2019) Information-Theoretic Intrinsic Plasticity for Online Unsupervised Learning in Spiking Neural Networks. Front. Neurosci. 13:31. doi: 10.3389/fnins.2019.00031

1. INTRODUCTION

Neural plasticity, the brain's ability to adapt in response to stimuli from the environment, has received increasing interest from both a biological and a computational perspective. As one such main self-adaptive mechanism, intrinsic plasticity (IP) plays an important role in temporal coding and maintenance of neuronal homeostasis. Behaviors of IP have been discovered in brain areas of many species, and IP has been shown to be crucial in shaping the dynamics of neural circuits (Marder et al., 1996). In particular, Baddeley et al. (1997) observed the exponentially distributed neuron responses in visual cortical neurons. Such responses may aim at allowing neurons to transmit the maximum amount of information, e.g., measured by the highest entropy, to their outputs with a constrained level of firing activity. Discovered in individual biological neurons, IP changes the excitability of neurons through modification of voltage-gated channels (Desai et al., 1999).

From a computational point of view, one of the early biological IP models was explored on the Hodgkin-Huxley type neurons where a number of voltage-gated conductances were considered (Stemmler and Koch, 1999). Since then, much IP mechanism research has been conducted for different kinds of artificial neurons. On the one hand, Triesch (2005) first proposed a mathematical approach to derive an IP rule based on the sigmoid neuron model. This work used the Kullback Leibler (KL) divergence from an exponential distribution to the actual output firing rate distribution to derive an adaptation rule for the neuron to generate responses following the exponential distribution. Based on the same principle, an IP rule for hyperbolic tangent neurons was also proposed (Schrauwen et al., 2008). On the other hand, IP can control average firing activity and aid synapses to undergo Hebbian modification via STDP depending upon their history of use (Watt and Desai, 2010). Furthermore, it was shown that an improvement in performance could be obtained when the reservoir of an echo state network (ESN) is adapted using IP such that the neurons in the network can autonomously tune themselves to the desired output distribution (Schrauwen et al., 2008).

As the third generation of artificial neural networks, it has been shown that spiking neural networks (SNN) are more computationally powerful than previous generations of neural networks (Maass, 1997). However, developing effective intrinsic plasticity (IP) mechanisms for SNNs is a challenging problem. Several empirical IP rules were proposed for SNNs, however, without a rigorous theoretical foundation. Lazar et al. (2007) presented an IP rule by which a spiking neuron's firing threshold voltage changes by a fixed value per update based on whether the neuron fired or not. However, this method cannot precisely determine when and how much the firing threshold voltage should be changed in different situations, and there is no clear understanding of the optimality of the resulting IP behavior. Li and Li (2013) presented an approach in which the parameters of the IP rule derived for sigmoid neurons in Li (2011) were empirically mapped to ones for spiking neurons. Since this rule is derived based on the sigmoid neuron model which is significantly different from the spiking neuron model, the property of this IP rule remains elusive when it is applied to adapt the output firing activity of spiking neurons. Recently, Li et al. (2018) proposed an IP rule according to the inter-spike-interval (ISI). However, similar to Lazar et al. (2007), this method only constraints the ISI into a certain range but does not have a rigorous target for adapting the output response. Moreover, Panda and Roy (2017) proposed another homeostasis mechanism called Non-Hebbian Plasticity which decays synaptic weights based on the activity of postsynaptic neurons. It can control the reservoir neurons responses to match the firing rate profile of the input and also avoid weight crowding caused by STDP. This Non-Hebbian Plasticity is based on synaptic plasticity which is different from IP, the intrinsic neuronal plasticity. As discussed in Watt and Desai (2010), both of them are homeostatic plasticity mechanisms and observed in biological neurons. They can work together for homeostatic regulation.

From an information theoretical perspective, it may hypothesize that a nervous cell maximizes the mutual information between its input and output. Neglecting the intrinsic uncertainty of the output, i.e., the output uncertainty after the input is known, the above target is equivalent to maximizing the output entropy. To this end, it is instrumental to note that the exponential distribution of the output firing rate attains the maximum entropy under the constraint of a fixed mean firing rate (Bell and Sejnowski, 1995). Thus, inspired by the IP rule for sigmoid neurons of Triesch (2005), we aim to derive an IP rule for spiking neurons while minimizing the difference between the output firing rate distribution and the targeted exponential distribution. However, there are several significant challenges in deriving such a rule. Unlike artificial neurons whose output is in the form of firing rate, spiking neurons generate responses in the form of discrete spikes. As a result, firing rate information, as well as its dependency on the input, must be appropriately characterized from discrete spike times, which has not been established before under the context of intrinsic plasticity. Besides, it is not clear how a proper expression of the entropy of the output firing rate distribution (or its difference from the targeted exponential distribution) can be derived and robustly maximized (minimized) in an online fashion.

In this article, we approach the above challenges as follows. First, we derive a differentiable transfer function bridging the input current strength and output firing rate when the input level is fixed based on the leaky integrate-and-fire(LIF) model. This transfer function is referred to as the firing-rate transfer function (FR-TF). It shall be noted that FR-TF can correlate the dynamic evolution of the output firing activity measured as averaged firing rate as a function of a received input over a sufficiently long timescale. Next, with this transfer function, we derive an information-theoretical intrinsic plasticity rule for spiking neurons, dubbed *SpiKL-IP*, to minimize the KL-divergence from the exponential distribution to the output firing rate distribution. We further present an online version of the SpiKL-IP rule for minimizing our KL-divergence based loss function in a way analogous to the stochastic gradient descent (SGD) method, which is widely adopted for training deep learning neural networks. Finally, we address two practical issues to ensure the proper operation and robustness of the proposed online IP rule. Among the two issues, it is desirable to apply the proposed IP tuning using the instantaneous input current and the measured output firing rate, allowing seamless consideration of the potentially dynamically changing current input. However, this creates a mismatch to the underlying FR-TF transfer function, which is addressed by making the online IP rule dependent only on the output firing rate such that the LIF model parameters are tuned based on the input/output activities of long timescales. Under various settings, the outputs of targeted spiking neurons converge robustly to the desirable exponential distribution under the proposed SpiKL-IP rule.

We evaluate the learning performance of the proposed IP rule for real-world classification tasks under the context of the liquid state machine (LSM). When applied to the reservoir neurons of LSM networks, our rule produces significant performance boosts. Based on the TI46 Speech Corpus (Liberman et al., 1991), the SpiKL-IP rule boosts the recognition accuracy by 6% for single-speaker English letter recognition and by up to more than 16% for the challenging task of multiple-speaker English letter recognition. For image classification using the CityScape dataset (Cordts et al., 2016), our proposed method can improve the accuracy by more than 2%.

The rest of this article is organized as follows. Section 2 first introduces previous intrinsic plasticity working on spiking neurons. Then, it presents the derivation of the proposed firing-rate transfer function (FR-TF) and the complete online IP rule. Section 3 demonstrates the application of the proposed IP under various simulation settings. Finally, section 4 concludes this work.

2. MATERIALS AND METHODS

2.1. Previous IP Rules for Spiking Neurons

Unlike other types of artificial neurons, instead of producing continuous-valued firing rates, spiking neurons generate spike trains, which are not differentiable at the times of spikes. Thus, the relationship among the input, parameters of the neuron model, and the output firing rate become obscure. This is perhaps partially why intrinsic plasticity has not been deeply investigated for spiking neurons. A few empirical IP rules were proposed for spiking neuron model, which unfortunately lack rigor.

Lazar et al. (2007) proposed an IP rule to adjust the firing threshold voltage as follows

$$V_{th,i}(t+1) = V_{th,i}(t) + \eta\left(x_i(t) - \frac{k}{N}\right), \tag{1}$$

$V_{th,i}$ is the threshold of the neuron i, η the learning rate which is chosen to be small, $x_i(t)$ the sum of Dirac delta functions and it is 1 if the neuron fires an output spike at time t and 0 otherwise, k and N some chosen constants. This rule drives a neuron to spike on average k out of N times. It only targets setting the mean firing rate to a chosen value by adapting the firing threshold but does not attempt to generate the optimal output response, i.e., the optimal firing rate distribution.

Li (2011) derived an IP rule that tunes sigmoid neurons to follow the Weibull distribution in the same way as in Triesch (2005). Li and Li (2013) adopted this rule for spiking neurons by merely substituting the tuning parameters of the sigmoid neuron model to the parameters for spiking neurons, namely rR and rC, which are the reciprocals of the leaky resistance and membrane capacitance, respectively. As analyzed by the authors, this rule can make the firing activity of a spiking neuron at a "low but not too low" level. However, since this rule results from a simple mapping from the sigmoid neuron IP rule, it may not produce the optimal firing rate distribution for spiking neurons.

Li et al. (2018) proposed an approach based on the Izhikevich model (Izhikevich, 2003) to adjust the output firing activity such that the inter-spike-interval (ISI) is set between some limits specified by T_{min} and T_{max}. This basic idea is the same as the one in Lazar et al. (2007) but using a different neuron model. Again, this rule aims at helping the neuron to generate responses at a desired firing rate level without optimizing the output distribution to maximize the information content.

As discussed above, the existing IP rules for spiking neurons are empirical in nature and are not derived with a rigorous optimization objective in mind. Furthermore, no success in real-world learning tasks has been demonstrated. We address these limitations by rigorously deriving an IP rule that robustly produces the targeted optimal exponential firing rate distribution and leads to significant performance improvements by realistic speech and image classification tasks.

2.2. Firing-Rate Transfer Function

The leaky integrated-and-fire (LIF) model is one of the most prevalent choices for describing dynamics of spiking neurons. This model is given by the following differential equation (Gerstner and Kistler, 2002)

$$\tau_m \frac{dV}{dt} = -V + Rx \tag{2}$$

where V is the membrane potential, x the input current, τ_m the time constant of membrane potential with value $\tau_m = RC$, where R and C are the effective leaky resistance and effective membrane capacitance. Once the membrane potential V exceeds the firing threshold V_{th}, the neuron generates a spike, and the membrane potential is reset to the resting potential, which is $0\,mV$ in our case. A refractory period of duration t_r is also considered after a spike is generated during which V is maintained at $0\,mV$.

Before presenting the proposed SpiKL-IP rule for spiking neurons, we shall first establish the relationship between the input current and the resulting output firing rate. This relationship is not evident since the response is in the form of spikes and it depends on the cumulative effects of all the past input. As a result, it is difficult to evaluate the output firing rate of spiking neurons at each time point under a varying input. We deal with this difficulty by deriving the proposed firing-rate transfer function (FR-TF) where the input is assumed to be constant. In other words, FR-TF correlates the dynamic evolution of the output firing activity measured as averaged firing rate as a function of a received input over a sufficiently long timescale.

Assuming that the input current x_0 is constant and integrating (2) with the initial condition that $V\left(t^{(1)}\right) = 0$ gives the interspike interval $T_{isi} = t^{(2)} - t^{(1)}$ (Gerstner and Kistler, 2002)

$$T_{isi} = t_r + \tau_m ln \frac{Rx_0}{Rx_0 - V_{th}}, \qquad Rx_0 > V_{th}. \tag{3}$$

where the constraint of $Rx_0 > V_{th}$ comes from the fact that only when the constant input current is sufficiently large, the neuron can generate spikes. Since both the input x_0 and T_{isi} are constant, the mean output firing rate of the spiking neuron is given by

$$y = \frac{1}{T_{isi}} = \frac{1}{t_r + \tau_m ln \frac{Rx_0}{Rx_0 - V_{th}}}, \qquad Rx_0 > V_{th}. \tag{4}$$

In this way, we obtain the transfer function of spiking neurons under the condition that it has constant input so that this relation between input and output can be used in the deriving process. Since this function can only represent spiking neurons with a fixed input, to distinguish the spiking neurons and this transfer function, when referring to firing-rate model neurons, it means the neurons with this firing-rate transfer function (4).

Figure 1 shows two tuning curves of the firing-rate transfer function where the input current level is swept while either the leaky resistance R or the membrane time constant τ_m is held at a specific value. As shown in **Figure 1A**, changing R while fixing τ_m modifies both the bias and curvature of the tuning curve. **Figure 1B** illustrates that τ_m controls the curvature of the tuning curve when R is fixed. In the following part, the proposed SpiKL-IP Rule is based on tuning R and τ_m. Note that separately adjusting R and τ_m requires a neuron to vary its capacitance in response to its activity while changing capacitance is not observed in biological neurons to date.

2.3. Proposed SpiKL-IP Rule

Based on the presented firing-rate transfer function (4), we now take an information-theoretical approach to derive the SpiKL-IP rule to minimize the KL-divergence from the exponential distribution to the output firing rate distribution. We will show how the SpiKL-IP rule can be cast into an online form to adapt R and τ_m, and then address one practical issue to ensure the proper operation and robustness of the proposed online IP rule.

2.3.1. The Basic SpiKL-IP Rule

We consider the information processing of a given spiking neuron as it receives stimuli from external inputs or other neurons in the same network over a dataset, mimicking part of the lifespan of the biological counterpart. We define the input and output firing rate probability distributions for each spiking neuron in the following way. As shown in **Figure 2**, the input current level X varies across different time points, it forms an input probability distribution over the course of the entire process denoted by $f_x(x)$. Accordingly, the output firing rate Y varies over time and forms an output probability distribution denoted by $f_y(y)$.

The goal of the SpiKL-IP rule is to obtain an approximately exponential distribution of the output firing rate at a fixed level of metabolic costs. In a biological perspective, exponential distributions of the output firing rate have been observed in mammalian visual cortical neurons responding to natural scenes and allow the neuron to transmit the maximum amount of information given a fixed level of metabolic costs (Baddeley et al., 1997).

From an information-theoretic point of view, Bell and Sejnowski (1995) argued that a neuron might self-adapt to maximize the mutual information of the input X and the output Y, a measure for the amount of information about the input obtained from the output, or vice versa

$$I(Y, X) = H(Y) - H(Y|X), \tag{5}$$

where $H(Y)$ is the entropy of the output while $H(Y|X)$ represents the amount of entropy (uncertainty) of the output which does not come from the input. Under the assumption that the output noise N is additive and there is no input noise, the conditional entropy can be simplified to $H(Y|X) = H(N)$ (Nadal and Parga, 1994; Bell and Sejnowski, 1995) which does not depend on the neural parameters. Thus, maximizing $I(Y, X)$ is equivalent to maximizing $H(Y)$ (Bell and Sejnowski, 1995). To this end, it is instrumental to note when the mean of the distribution is kept constant, the exponential distribution corresponds to the largest entropy among all probability distributions of a non-negative random variable. This leads to the conclusion that the exponential distribution with a targeted mean shall be the optimal distribution for the output firing rate, where the mean specifies the practical constraint on energy expenditure. In addition, in this work, all neurons are implemented using the LIF model which is noiseless and no noise is added explicitly to the neuronal dynamics, which means that $H(N) = 0$ (Gerstner and Kistler, 2002). The exponential distribution is given by

$$f(x) = \mu exp(-\mu x), \qquad x >= 0, \tag{6}$$

where μ is the mean of the distribution.

Inspired by the IP rule for sigmoid neurons in Triesch (2005), we derive the SpiKL-IP rule for spiking neurons while minimizing the KL-divergence from a targeted exponential distribution to the actual output firing rate distribution, where Kullback Leibler divergence (KL-divergence) is used as a difference measure as follows

$$\begin{aligned} D &= d\left(f_y(y)||f_{exp}\right) \\ &= \int f_y(y) log\left(\frac{f_y(y)}{\frac{1}{\mu}exp(\frac{-y}{\mu})}\right) dy \\ &= \int f_y(y) log(f_y(y)) dy + \int f_y(y)\left(\frac{y}{\mu}\right) dy \\ &\quad + \int f_y(y) log\mu dy, \end{aligned} \tag{7}$$

where y and $f_y(y)$ denote the output, and the output firing rate distribution, respectively, and μ is the mean value of the targeted exponential distribution. The smaller the KL-divergence D is, the closer the exponential distribution is to the output distribution. In (7), since $\int f_y(y) dy = 1$ the third integral evaluates to a fixed value of $log\mu$. Minimizing KL-Divergence D by adapting R and τ_m reduces to minimize the first two integrals, giving rise to the following loss function

$$\begin{aligned} L &= \int f_y(y) log(f_y(y)) dy + \int f_y(y)\left(\frac{y}{\mu}\right) dy \\ &= E\left[log(f_y(Y)) + \frac{Y}{\mu}\right]. \end{aligned} \tag{8}$$

Note that (8) is in terms of an expectation over the entire output distribution. Now, we convert (8) into an online form that is analogous to the stochastic gradient descent method with a batch size of one. To make SpiKL-IP amenable for online training, using a proper stepsize we discretize the entire training process into multiple small time intervals each in between two

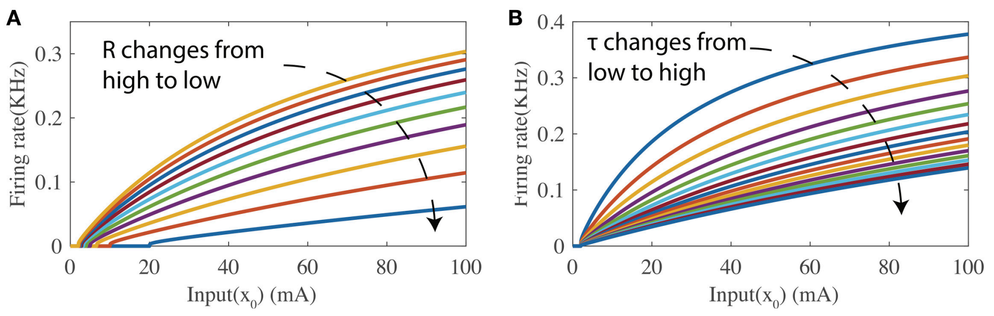


FIGURE 1 | The firing-rate transfer function (FR-TF). **(A)** As a function of the leaky resistance *R*, and **(B)** as a function of the membrane time constant τ_m.

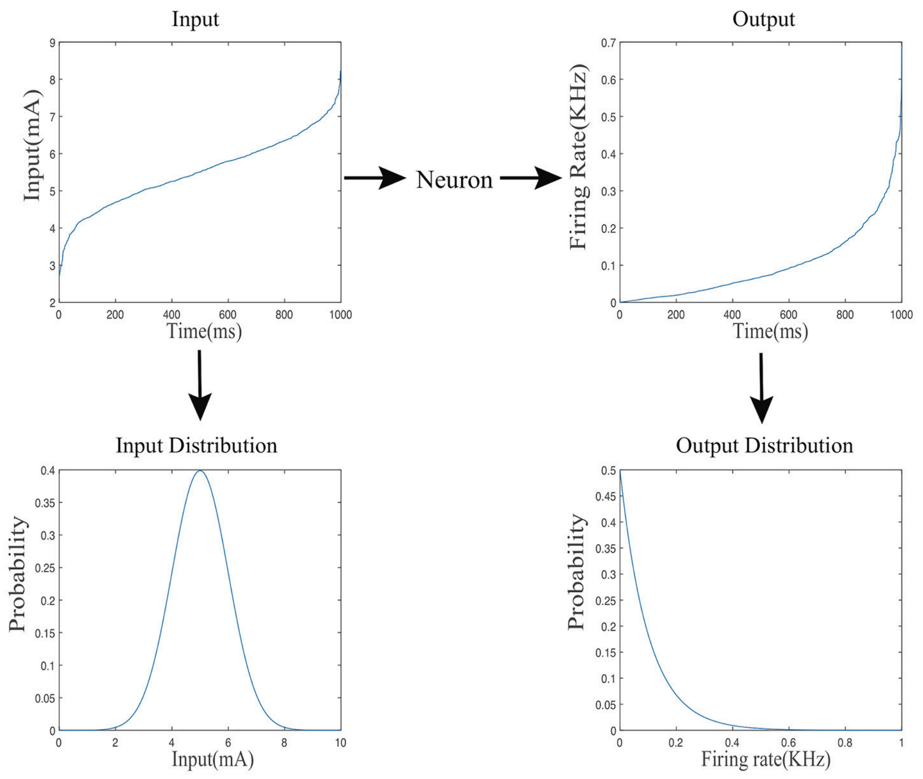


FIGURE 2 | The mapping from the input current distribution to the output firing rate distributing of a neuron.

adjacent time points as shown in **Figure 3**. The input level to the spiking neuron at each time point is considered as an individual observation or training example. In this way, the adjustment of the tunable parameters is not delayed until the output firing rate distribution is collected after the entire dataset is applied to the neuron (or neural network). Instead, these parameters are adjusted as the neuron experiences a given input example at each time point in an online manner. To do this, the following loss function that corresponds to the received input example is minimized at each time point *t*

$$L(t) = log(f_y(y(t))) + \frac{y(t)}{\mu}, \tag{9}$$

where $y(t)$ denotes the output firing rate Y observed at time t. From now on, we drop the explicit dependency of $y(t)$ and $x(t)$ (observed input level at t) on t for notational simplicity. Recognizing that the output probability distribution relates to the input counterpart by Papoulis and Pillai (2002)

$$f_y(y) = \frac{f_x(x)}{\frac{\partial y}{\partial x}} \tag{10}$$

and substituting it into (9) leads to

$$L(t) = log(f_x(x)) - log\left(\frac{\partial y}{\partial x}\right) + \frac{y}{\mu}, \tag{11}$$

which can be further simplified to

$$\hat{L}(t) = -log\left(\frac{\partial y}{\partial x}\right) + \frac{y}{\mu}, \tag{12}$$

as $log(f_x(x))$ is a function of the input probability distribution and does not depend on R and τ_m.

The online SpiKL-PI rule is based upon the partial derivatives of (9) with respect to x, R and τ_m. We first shall compute the derivatives of the output firing rate $y(t)$ with respect to x, R, τ_m. We make use of the firing rate transfer function (4) whose application at each time point t is justified if the input $x(t)$ changes slowly with respect to the chosen stepsize and the averaged output firing rate measure is used, and obtain

$$\frac{\partial y}{\partial x} = \frac{y^2 \tau_m V_{th}}{x(Rx - V_{th})} \tag{13}$$

$$\frac{\partial y}{\partial R} = \frac{y^2 \tau_m V_{th}}{R(Rx - V_{th})} \tag{14}$$

$$\frac{\partial y}{\partial \tau_m} = \frac{t_r y^2 - y}{\tau_m}. \tag{15}$$

Taking (13) into account, the partial derivatives of the loss function (9) with respect to R and τ_m are found to be

$$\begin{aligned}\frac{\partial L}{\partial R} &= \frac{\partial}{\partial R}\left(-log\left(\frac{\partial y}{\partial x}\right) + \frac{y}{\mu}\right) \\ &= \frac{\partial}{\partial R}\left(-(2log(y) - log(Rx - V_{th})) + \frac{y}{\mu}\right) \\ &= \frac{\left(\frac{y^2}{\mu} - 2y\right)\tau_m V_{th} + Rx}{R(Rx - V_{th})}\end{aligned} \tag{16}$$

and

$$\begin{aligned}\frac{\partial L}{\partial \tau_m} &= \frac{\partial}{\partial \tau_m}\left(-log\left(\frac{\partial y}{\partial x}\right) + \frac{y}{\mu}\right) \\ &= \frac{\partial}{\partial \tau_m}\left(-(2log(y) + log\tau_m) + \frac{y}{\mu}\right) \\ &= \frac{1 + \frac{1}{\mu}(t_r y^2 - y) - 2t_r y}{\tau_m},\end{aligned} \tag{17}$$

respectively, which gives the following online IP rule

$$\begin{aligned}R &= R - \eta_1 \frac{\partial L}{\partial R} \\ &= R + \eta_1 \frac{\left(2y - \frac{y^2}{\mu}\right)\tau_m V_{th} - Rx}{R(Rx - V_{th})}, \qquad Rx > V_{th} \\ \tau_m &= \tau_m - \eta_2 \frac{\partial L}{\partial \tau_m} \\ &= \tau_m + \eta_2 \frac{2t_r y - 1 - \frac{1}{\mu}(t_r y^2 - y)}{\tau_m}, \qquad Rx > V_{th}.\end{aligned} \tag{18}$$

where η_1 and η_2 are learning rates, μ the constant value depending on the desired mean of the output firing rate. The condition that $Rx > V_{th}$ comes from the transfer] function (4).

2.3.2. Practical Considerations

While (18) has the critical elements of the proposed online IP rule, its direct implementation, however, has been experimentally shown to be unsuccessful, i.e., it can neither train spiking neurons to generate output firing rates following the exponential distribution nor improve SNN learning performance for real-world classification tasks. The problem has to do with the fact that one underlying assumption behind the firing rate transfer function (FR-TF) (4) and hence the IP rule (18) is that the input current is constant or changes over a sufficiently slow timescale. However, in a practical setting, the total postsynaptic input received by a spiking neuron does vary in time, and the rate of change depends on the frequency of firing activities of its presynaptic neurons. With the internal dynamics, the output firing level of a spiking neuron cannot immediately follow the instantaneous current input, e.g., it is possible that the output firing rate is still low while the input current has increased to a high level. As a result, the assumption on the input current is somewhat constraining, and its violation leads to the ineffectiveness of IP tuning.

On the other hand, it is worth noting that the FR-TF captures the correlation between the average input current and the output firing rate over a long timescale. In the meantime, the proposed IP rule aims to adapt spiking neurons to produce a desired probability distribution of the output firing rate. In other words, the objective is not to tune each instance of the output firing rate. Instead, it is to achieve a desirable collective characteristic of the output firing rate measured by an exponential distribution. In some sense, the FR-TF correlates the input and output correspondence in a way that is meaningful for the objective of online IP tuning.

To find a solution to the above difficulty, we remove the dependency on the instantaneous input current from the IP rule of (18) by substituting the input x using the output firing rate y using the transfer function (4). More specifically, a new variable W is defined by $W = Rx - V_{th}$, which can be expressed using y based on (4) as

$$W = \frac{V_{th}}{e^{\left(\frac{1}{\tau_m}\left(\frac{1}{y} - t_r\right)\right)} - 1}. \tag{19}$$

Making use of (19), (18) is converted to a form which only depends on y

$$\begin{aligned}R &= R + \eta_1 \frac{2y\tau_m V_{th} - W - V_{th} - \frac{1}{\mu}\tau_m V_{th} y^2}{RW}, \qquad y > 0. \\ \tau_m &= \tau_m + \eta_2 \frac{2t_r y - 1 - \frac{1}{\mu}(t_r y^2 - y)}{\tau_m}.\end{aligned} \tag{20}$$

As can be seen, the rule in (20) adjusts the two parameters only based on the output firing rate y. Substituting the instantaneous value of x by the firing rate y based on the firing rate transfer function effectively operates the IP rule based on the averaged input/output characteristics over a longer timescale.

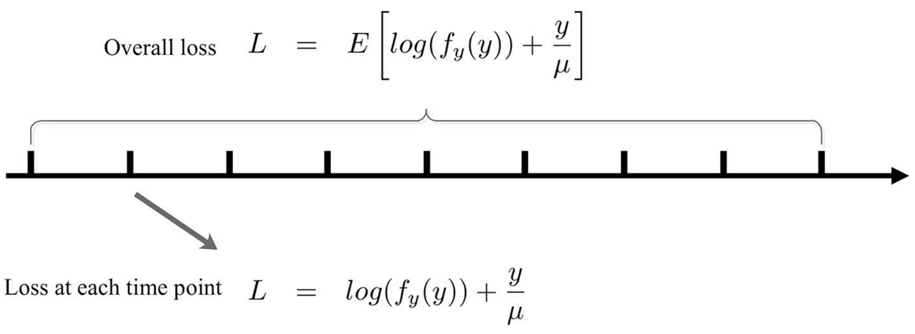

Overall loss $L = E\left[log(f_y(y)) + \frac{y}{\mu}\right]$
Loss at each time point $L = log(f_y(y)) + \frac{y}{\mu}$

FIGURE 3 | Online SpiKL-IP learning: minimization of the KL divergence at each time point during the training process.

Note that the condition that $Rx > V_{th}$ in (18) is changed to an equivalent form of $y > 0$ in (20). A closer examination of **Figure 1** shows that the firing rate transfer functions are not differentiable around $y = 0$ ($Rx = V_{th}$). Interpreting differently, the proposed IP tuning can operate only when the output firing rate is nonzero. To further improve the robustness of the proposed IP rule, the tuning in (20) is only activated when $y > \delta$ with δ being small such as 1 Hz. When $y \leq \delta$, R and τ_m are increased and decreased respectively to bring up the output firing activity.

Putting everything together, the final SpiKL-IP rule is

$$R = \{ \begin{matrix} R + \eta_1 \frac{2y\tau_m V_{th} - W - V_{th} - \frac{1}{\mu}\tau_m V_{th} y^2}{RW}, & y > \delta \\ R + \eta_1 \alpha_1, & y \leq \delta \end{matrix}$$
$$\tau_m = \{ \begin{matrix} \tau_m + \eta_2 \frac{2t_r y - 1 - \frac{1}{\mu}(t_r y^2 - y)}{\tau_m}, & y > \delta \\ \tau_m - \eta_2 \alpha_2, & y \leq \delta \end{matrix} \quad (21)$$

where α_1 and α_2 are chosen to be small.

To provide an intuitive understanding of the proposed SpiKL-IP rule, **Figure 4** shows how R and τ_m are altered by one-time application of SpiKL-IP at different output firing rate levels starting from a chosen combination of R and τ_m values.

3. RESULTS

To demonstrate the mechanisms and performances of the proposed SpiKL-IP rule, we conduct three types of experiments by applying SpiKL-IP to single neuron as well as a group of spiking neurons as part of a neural network. First, we show that when applied to a single neuron whose behavior is governed by the firing-rate transfer function (4) the proposed rule can tune the neuron to produce the targeted exponential distribution of the output firing rate even under a time-varying input. Then, we apply SpiKL-IP to a single spiking neuron as well as a group of spiking neurons to demonstrate that our rule can robustly produce the desired output firing distribution in all tested situations even although it is derived from the FR-TF which is based on the assumption that the input is constant. Finally, we demonstrate the significant performance boosts achieved by SpiKL-IP when applied to real-world speech and image classification tasks. Furthermore, we compare SpiKL-IP with two existing IP rules for spiking neurons (Lazar et al., 2007; Li and Li, 2013). In this article, we name the IP rule in Lazar et al. (2007) as the Voltage-Threshold IP rule and one in Li and Li (2013) as the RC IP rule.

The following simulation setups are adopted in each experiment. We simulate the continuous-time LIF model in section 2.2 using a fixed discretization time step of $1ms$ according to which all neuronal activities are evaluated in lockstep. To measure the firing rate of each spiking neuron as a continuous-valued quantity over time under a constant of varying input, we use the intracellular calcium concentration $C_{cal}(t)$ as a good indicator of the averaged firing activity over a chosen timescale

$$\frac{dC_{cal}(t)}{dt} = -\frac{C_{cal}(t)}{\tau_{cal}} + \sum_i \delta(t - t_i), \quad (22)$$

where τ_{cal} is the time constant, and the output firing spikes are presented by a series of Dirac delta functions. According to (22), the calcium concentration increases by one unit when an output spike is generated and decays with a time constant τ_{cal} (Dayan and Abbott, 2001). The time-varying output firing rate is measured using the normalized calcium concentration

$$y(t) = \frac{C_{cal}(t)}{\tau_{cal}}. \quad (23)$$

3.1. Single Neurons Modeled by FR-TF

We apply the proposed SpiKL-IP rule to a single neuron modeled based on the firing-rate transfer function (4). The parameters of the neuron and SpiKL-IP are set as follows: $V_{th} = 20\,mV$, $t_r = 2\,ms$, and $\mu = 0.2\,KHz$. In addition, the tuning ranges for R and τ_m are set to $[1\,\Omega, 1024\,\Omega]$ and $[1\,ms, 1,024\,ms]$ with R and τ_m initialized to $64\,\Omega$ and $64\,ms$, respectively. The input current level at each time point is randomly generated according to a Gaussian distribution with the mean of $7mA$ and variance of $1mA$ as well as a uniform distribution between $[0.5\,mA, 5.5\,mA]$ in a way that is similar to the setups in Triesch (2005); Li and Li (2013). For both cases, a total of $10,000$ time points are considered.

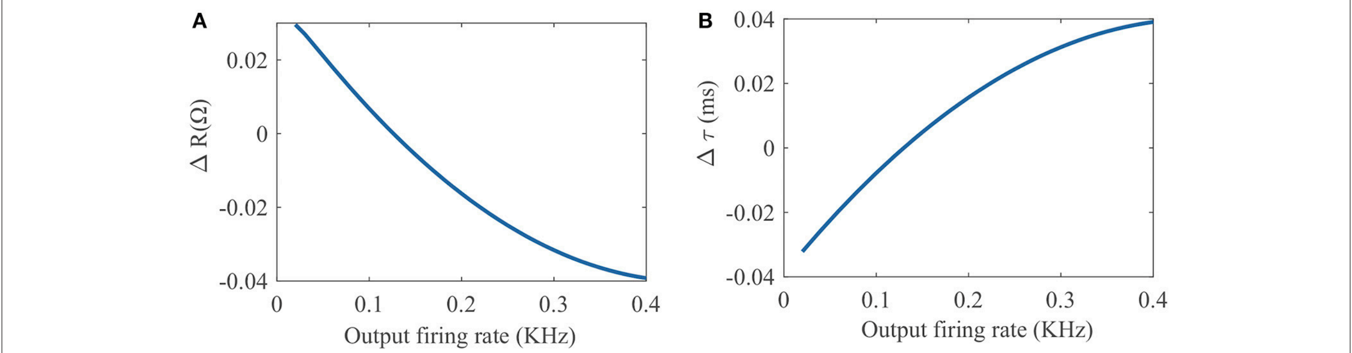


FIGURE 4 | Tuning characteristics of one-time application of SpiKL-IP at different output firing rate levels starting from a chosen combination of R and τ_m values R and τ_m. **(A)** Tuning of the leaky resistance R, and **(B)** tuning of the membrane time constant τ_m.

In **Figure 5**, we compare the recorded output firing rate distribution when no IP tuning is used with the one that is produced by the proposed SpiKL-IP rule under two random input distributions. In each plot of **Figure 5**, we fit the actual firing histogram with to a closest exponential distribution (red curve). It is evident from **Figures 5A,C** that without IP tuning the output firing distribution is far from the targeted optimal exponential distribution with the maximum entropy. With the application of SpiKL-IP, however, the output distribution can be trained to almost converge to the desirable exponential distribution under two dramatically different input distributions. Note that since the simulation time stepsize is $1\,ms$, the output firing rate is bound by $1\,KHz$. This creates a subtle difference between the actual and the exponential distribution at the tails of the two distributions, which is negligible in practice. These results indicate that the proposed IP rule can robustly maximize the information contained in the output firing rate distribution by tuning it toward the exponential distribution regardless of the input distribution.

3.2. Leaky Integrate-and-Fire Spiking Neurons

Since SpiKL-IP is based on the firing-rate transfer function which only characterizes the behavior of LIF neurons over a large timescale, it is interesting to test SpiKL-IP using LIF neurons. The parameters for the spiking neurons and SpiKL-IP are set as follow: $V_{th} = 20\,mV$, $t_r = 2\,ms$, $\mu = 0.2\,KHz$, $\tau_c = 64\,ms$ with R and τ_m initialized to $64\,\Omega$ and $64\,ms$, respectively. The tuning ranges for R and τ_m are again set to $[1\,\Omega, 1,024\,\Omega]$ and $[1\,ms, 1,024\,ms]$, respectively.

First, we apply SpiKL-IP to a single LIF neuron whose input is a spike (Dirac delta) train randomly generated according to a Poisson process with a mean firing rate of 160 Hz for a duration of 1,000 ms. The details of input generation are described in Legenstein and Maass (2007). The output firing rate is evaluated by the normalized intracellular calcium concentration in (23). **Figure 6** compares the output firing distributions generated with no IP and with the three IP rules. Clearly, the proposed rule produces an output distribution close to the desired exponential distribution while without IP tuning the neuron is unable to generate an exponentially distributed output. As shown in **Figure 6C**, the Voltage Threshold IP rule (Lazar et al., 2007) can only alter the average output firing rate rather than tuning the shape of the output firing rate distribution toward that of an exponential distribution. **Figure 6D** shows that it is also tricky for the RC IP rule (Li and Li, 2013) to train the neuron to generate an output whose distribution is close to the exponential distribution.

Next, more interestingly, we examine the behavior of IP tuning in a spiking neural network. In this case, we set up a fully connected recurrent network of 100 LIF neurons. There are 30 external inputs with each being a Poisson spike train with a mean rate of 80 Hz and a duration of $1,000ms$ as shown in **Figure 7**. Each input is connected to 30 neurons through synaptic whose weights are set to -8 or 8 with equal probability. The synaptic weights between the reservoir neurons in the network are uniformly distributed between -1 and 1. This neural network is similar to the reservoir network used in Schrauwen et al. (2008).

We randomly choose one neuron and record its output firing rate for a demonstration. As can be seen in **Figure 8A**, without IP tuning the output distribution is quite different from any exponential distributions. As shown in **Figures 8C,D**, neither the Voltage Threshold IP rule nor the RC IP rule can produce an output distribution that is reasonably close to an exponential distribution. In contrast, **Figure 8B** shows that the proposed SpiKL-IP rule leads to excellent results, generating an output distribution that is very close to an exponential distribution. These experiments demonstrate that SpiKL-IP maintains its effectiveness in the more complex network setting where spiking neurons interact with each other while receiving external spike inputs.

3.3. Real World Classification Tasks For LSM

Although intrinsic plasticity has been studied for a very long time with many different IP rules proposed, rarely any rule is tested on real-world learning tasks. As a result, it is not clear whether IP tuning is capable of improving the performance for these more meaningful tasks. In this paper, we realize several spiking neural networks based on the bio-inspired Liquid State Machine (LSM)

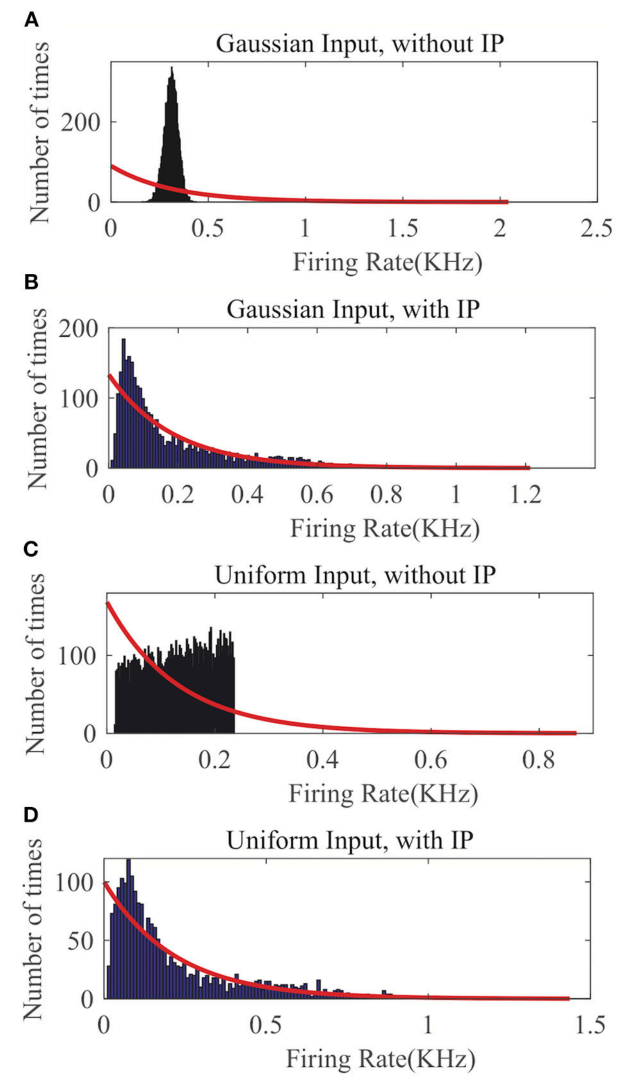


FIGURE 5 | The output firing-rate distributions of a single neuron characterized using the firing-rate transfer function and driven by randomly generated current input following a Gaussian or Uniform distribution. **(A)** Gaussian input without IP tuning, **(B)** Gaussian input with the SpiKL-IP rule, **(C)** uniform input without IP tuning, and **(D)** uniform input with the SpiKL-IP rule. The red curve in each plot represents the exponential distribution that best fits the actual output firing rate data.

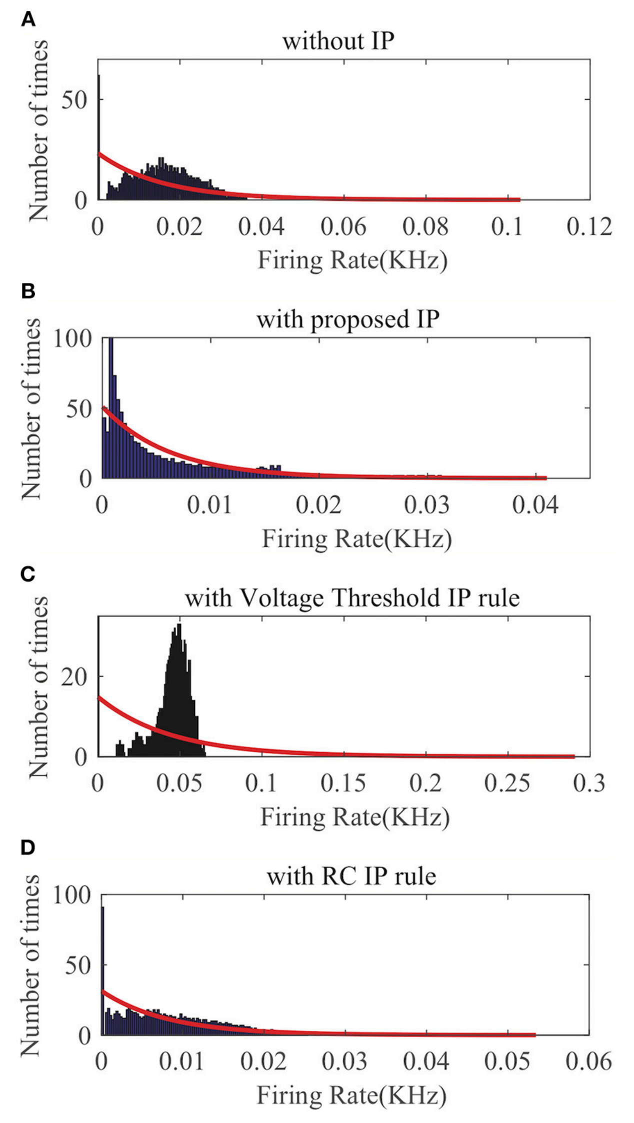


FIGURE 6 | Output firing rate distributions of a single spiking neuron: **(A)** without IP tuning, **(B)** with proposed SpiKL-IP rule, **(C)** with the Voltage Threshold IP rule, and **(D)** with the RC IP rule. The red curve in each plot represents the exponential distribution that best fits the actual output firing rate data.

network model and evaluate the performance of IP tuning using realistic speech and image recognition datasets.

LSM is a biologically plausible spiking neural network model with embedded recurrent connections (Maass et al., 2002). As shown in **Figure 9**, the LSM has an input layer, a recurrent reservoir, and a readout layer. The reservoir has a recurrent structure with a group of excitatory and inhibitory spiking neurons randomly connected in a way approximating the spatial distribution of biological neurons (Maass et al., 2002). Typically, the synaptic weights between the reservoir neurons are fixed. The input spike trains generate spatiotemporal firing patterns in the reservoir, which are projected onto the readout layer through full connectivity. In this paper, the feedforward plastic synapses between the reservoir neurons and readout are adjusted according to a bio-inspired spike-based online learning rule (Zhang et al., 2015). Several LSMs with different sizes are set up to evaluate the potential impact of an IP rule on classification performance.

For the networks evaluated using TI46, the input layer has 78 neurons. These networks have 135 (3*3*5), 270 (3*3*30), 540 (6*6*15) reservoir neurons, respectively, where each input neuron is randomly connected to 16, 24, 32 reservoir neurons with the weights set to 2 or -2 with equal probability, respectively. Among the reservoir neurons, 80% are excitatory, and 20% are inhibitory. The reservoir is composed of all types of synaptic

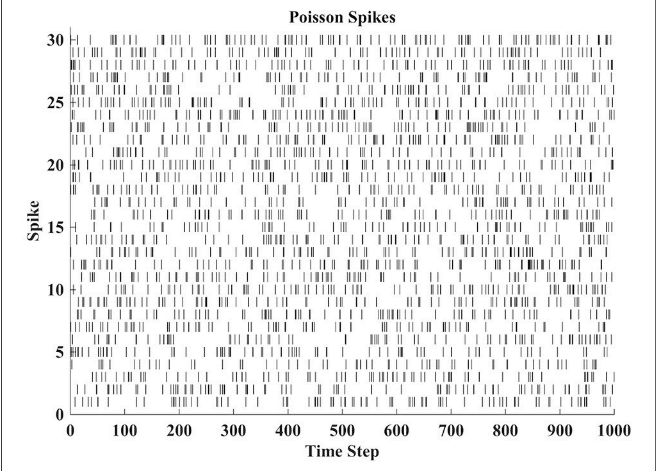


FIGURE 7 | 30 Poisson spike trains as input to a fully connected spiking neural network of 100 LIF neurons.

connections depending on the pre-neuron and post-neuron types including EE, EI, IE, II, where the first letter indicates the type of the pre-synaptic neuron, and the second letter the type of the post-synaptic neuron, and E and I mean excitatory and inhibitory neurons, respectively. The probability of a synaptic connection from neuron a to neuron b in the reservoir is defined as $C \cdot e^{-(D(a,b)/\lambda)^2}$, where λ is 3, C is 0.3 (EE), 0.2 (EI), 0.4 (IE), 0.1 (II), and D (a, b) is the Euclidean distance between neurons a and b (Maass et al., 2002). The synaptic weights in the reservoir are fixed to 1(EE, EI) or -1(IE, II). For the readout layer, the reservoir neurons are fully connected to 26 readout neurons with the weights randomly generated from -8 to 8 following the Gaussian distribution. All the readout synapses are plastic and trained according to Zhang et al. (2015). When testing an IP rule, it is only applied to the reservoir neurons. The parameters of each neuron are: $V_{th} = 20\,mV$, $t_r = 2\,ms$, $\mu = 0.2\,KHz$, $\tau_c = 64\,ms$, $\eta_1 = \eta_2 = 5$, and $\alpha_1 = \alpha_2 = 0.1$. R and τ_m are initialized to $64\,\Omega$ and $64\,ms$, respectively. The tuning ranges for R and τ_m are again set to $[32\,\Omega, 512\,\Omega]$ and $[32\,ms, 512\,ms]$, respectively. A 5-fold cross-validation scheme is adopted to obtain classification performances. Five hundred epochs are simulated, and the best results are reported.

For the networks evaluated using CityScape, the input layer has 225 neurons. These networks have 27 (3*3*3), 45 (3*3*5), 72 (3*3*8), 135 (3*3*15) reservoir neurons, each input neuron is randomly connected to 1, 4, 4, 64 reservoir neurons with the weights set to 2 or -2 with equal probability, respectively. Other settings of the networks are the same as those used for the ones evaluated based on TI46.

We also have made our implementation of SpiKL-IP rule for LSM available online[1].

[1]https://github.com/stonezwr/SpiKL-IP

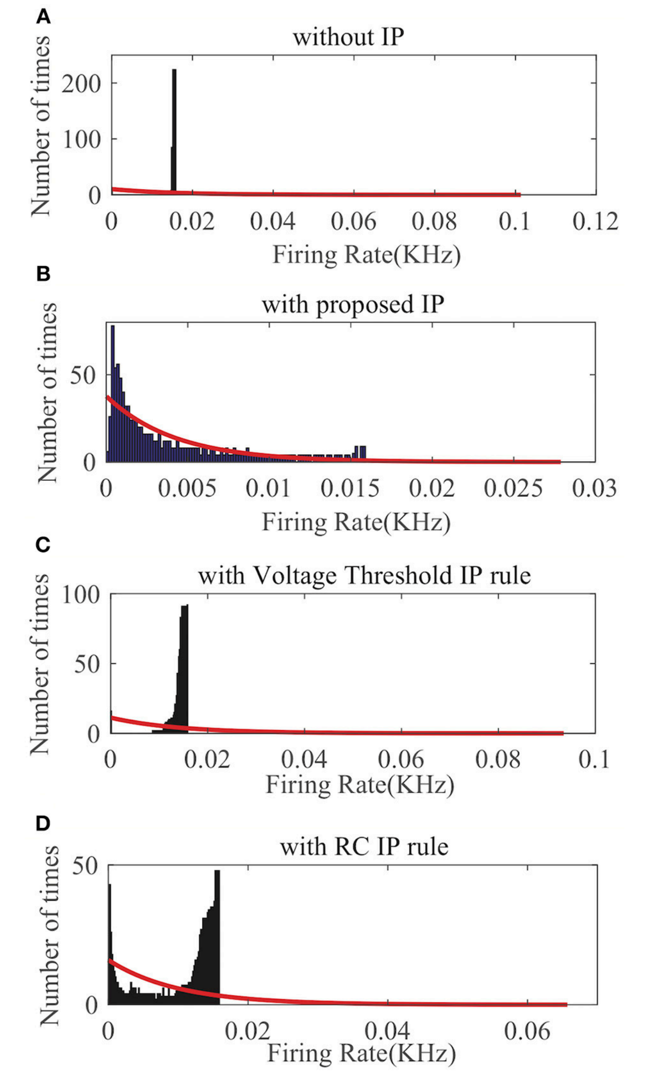


FIGURE 8 | Output firing rate distributions of one spiking neuron in a fully connected network. **(A)** without IP tuning, **(B)** with proposed SpiKL-IP rule, **(C)** with the Voltage Threshold IP rule, and **(D)** with the RC IP rule. The red curve in each plot represents the exponential distribution that best fits the actual output firing rate data.

3.3.1. Speech Recognition Using the TI46 Speech Corpus

The speech recognition task is evaluated on several subsets of the TI46 speech corpus (Liberman et al., 1991). This corpus contains spoken utterances from 16 speakers (eight males and eight females), each speaking 10 utterances of English letters from "A" to "Z". Before applying to the reservoir, each input sample is first preprocessed by the Lyon ear model (Lyon, 1982), then encoded into 78 spike trains with the BSA algorithm (Schrauwen and Van Campenhout, 2003).

Table 1 demonstrates the classification accuracy for a number of LSMs of different amounts of reservoir neurons with and without the proposed SpiKL-IP rule based on different subsets

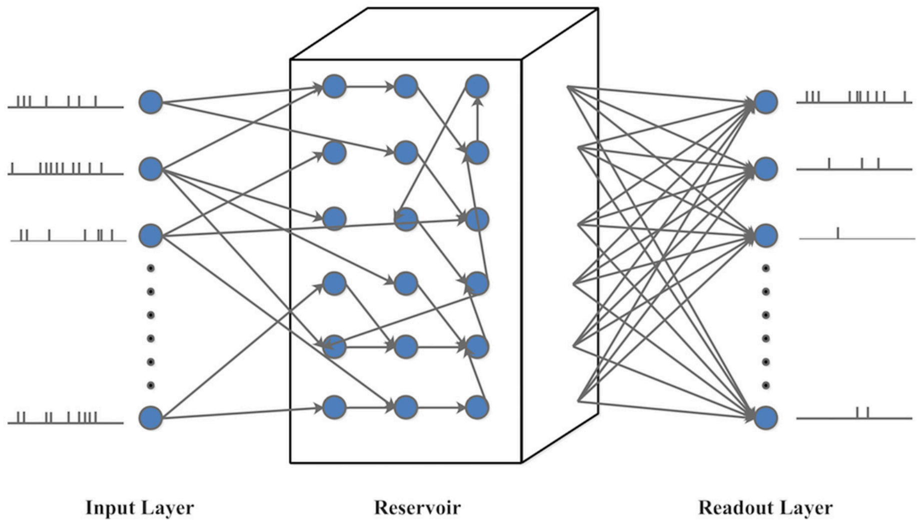


FIGURE 9 | The structure of Liquid State Machine (LSM).

TABLE 1 | The performances of LSM-based speech recognition with and without the proposed SpiKL-IP rule evaluated using the single and multi-speaker subsets of the TI46 Speech Corpus.

Dataset size	Reservoir size	Without IP (%)	With IP (%)
260 (1 Speaker)	90	88.46	97.31
	135	92.30	98.46
520 (2 Speakers)	135	86.15	92.31
	270	89.04	95.58
1,040 (4 Speakers)	135	79.04	87.69
	270	84.62	93.37
2,080 (8 Speakers)	270	72.69	86.95
	540	76.59	91.96
3,120 (12 Speakers)	270	72.17	84.25
	540	77.49	90.64
4,160 (16 Speakers)	270	70.76	83.98
	540	76.19	88.58

of the TI46 speech corpus. The 260-samples subset is a single speaker subset while ones with 520, 1,040, 2,080, 3,120, 4,160 samples contain 2, 4, 8, 12, and 16 speakers, respectively. It shall be noted that as the number of speakers increases, the recognition task becomes increasingly challenging. To the best knowledge of the authors, there exists no prior reported success on recognizing multiple-speaker subsets using spiking neural networks. As shown in **Table 1**, the recognition performs drops rapidly as the number of speakers increases without SpiKL-IP. In comparison, the use of SpiKL-IP can significantly boost the recognition accuracy by up to more than 16%. Moreover, SpiKL-IP leads to higher performance boosts as it is applied to smaller networks or more challenging subsets of greater numbers of speakers and samples.

From the LSM with 135 reservoir neurons, we randomly choose six neurons and record their firing responses on one of the speech samples after a few initial training iterations. **Figure 10** shows that most neurons' responses can follow the exponential distribution, demonstrating that the proposed SpiKL-IP rule can tune neurons to generate outputs with a distribution close to the exponential distribution in a complicated network. **Figure 11** shows the learning curves of R and τ_m for a reservoir neuron when one speech sample is repeatedly applied to the network for 15 iterations. **Figure 11B** shows that the value of R monotonically increases over time and finally converges under the proposed IP rule. However, **Figure 11A** shows that the value of τ_m fluctuates in every iteration without converging to a fixed value, but its trajectory exhibits a stable periodic pattern toward later iterations. This may be understood by the fact that to produce the desired exponential firing rate distribution, at least one of the two intrinsic neural parameters shall be dynamically adapted in response to the received time-varying input. **Figure 11C** shows the adaptation of the output firing rate y, which has also reached to a stable periodic pattern toward the end of the training process.

Figure 12 compares the recognition performances of several LSMs all with 135 reservoir neurons reported in related works. The performances are evaluated based upon the single-speaker subset with 260 samples. We adopt the LSM in Zhang et al. (2015) which makes use of a spike-based supervised learning rule for training the readout synapses and has no IP tuning as a baseline. The LSM in Jin and Li (2016) adds spike-timing-dependent plasticity (STDP) rule to the baseline to train the synaptic weights between reservoir neurons. On top of the baseline, we further implement the Voltage Threshold IP rule (Lazar et al., 2007), the RC IP rule (Li and Li, 2013), or the SpiKL-IP rule to tune the reservoir neurons. The proposed rule produces the highest recognition accuracy improvement of more than 6% over the baseline LSM.

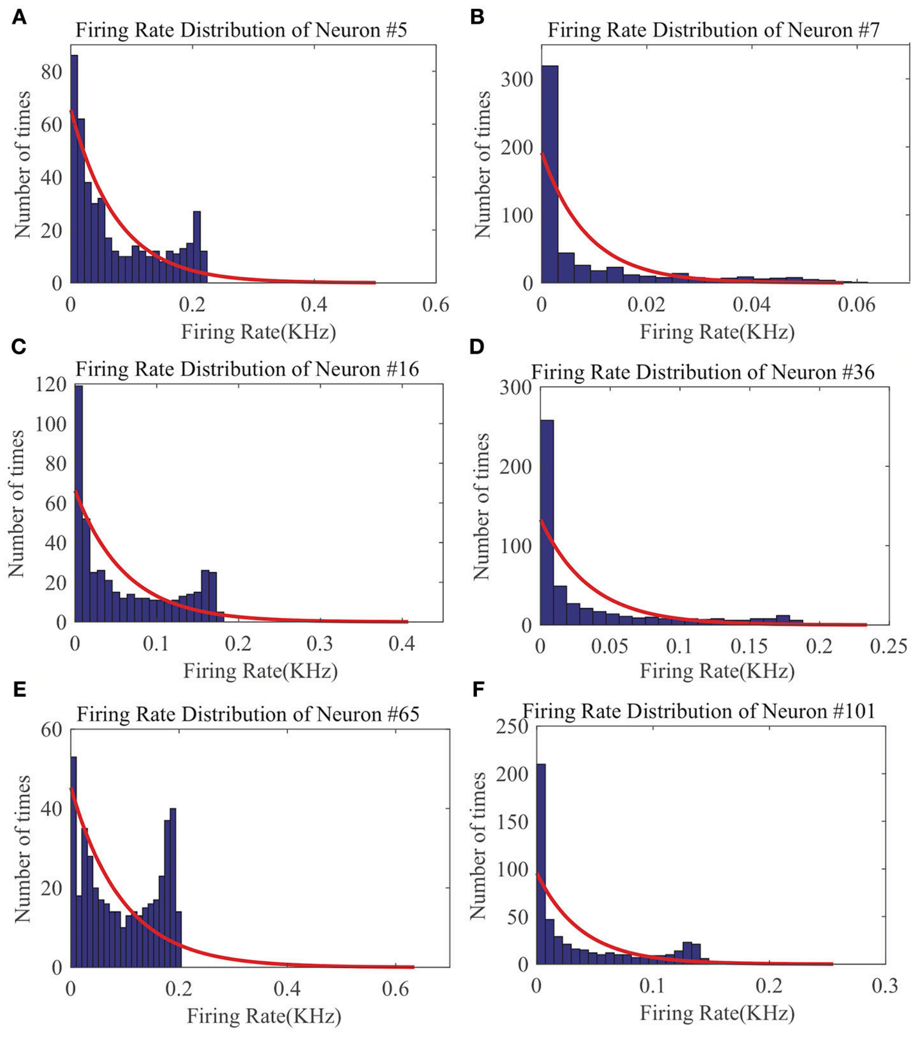


FIGURE 10 | The output firing distributions of six reservoir neurons in an LSM after the reservoir is trained by SpiKL-IP. The red curve in each plot represents the exponential distribution the best fits the actual output firing rate data. **(A)** Firing rate distribution of Neuron #5, **(B)** Firing rate distribution of Neuron #7, **(C)** Firing rate distribution of Neuron #16, **(D)** Firing rate distribution of Neuron #36, **(E)** Firing rate distribution of Neuron #65, and **(F)** Firing rate distribution of Neuron #101.

3.3.2. Image Classification Using the CityScape Dataset

The image classification task is based on the CityScape dataset (Cordts et al., 2016) which contains 18 classes of images of semantic urban scenes taken in several European cities. Each image is segmented and remapped into a size of 15×15, and then encoded into 225 input Poisson spike trains with the mean firing rate proportional to the corresponding pixel intensity. There are 1,080 images in total.

Table 2 summarizes the classification accuracy of four LSMs of different sizes with or without the SpiKL-IP rule. For each comparison, an LSM which is set up according to Zhang et al. (2015) and incorporates the same spike-based supervised learning rule of Zhang et al. (2015) for training the readout synapses without IP tuning is used as a baseline. It can be observed that the application of SpiKL-IP leads to noticeable performance improvements. For example, in the case of LSM with 45 reservoir neurons, the performance is improved from 91.74% to 94.44%.

4. DISCUSSION

While intrinsic plasticity (IP) was attempted for spiking neurons in the past, the prior IP rules lacked a rigorous treatment in their development, and the efficacy of these rules was not verified using practical learning tasks. This work aims to address the theoretical and practical limitations of the existing works by proposing the SpiKL-IP rule. SpiKL-IP is based upon a rigorous information-theoretic perspective where the target of IP tuning is to produce the maximum entropy in the resulting output firing rate distribution of each spiking neuron. The maximization of output entropy, or information transfer from the input to the output, is realized by producing a targeted optimal exponential distribution of the output firing rate.

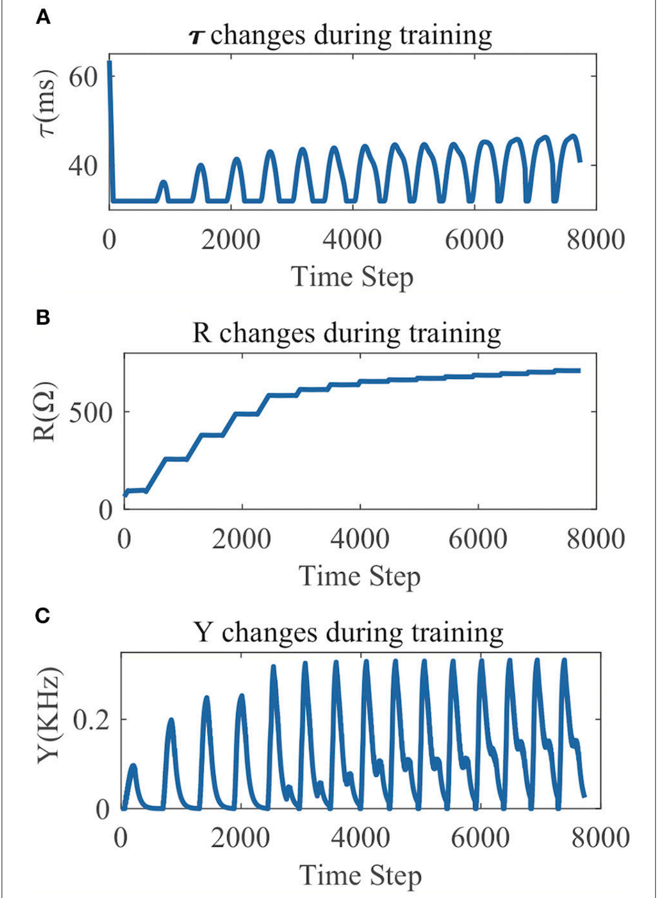


FIGURE 11 | The parameter tuning and firing rate adaption by SpiKL-IP for a reservoir neuron in an LSM during 15 iterations of training over a single speech example. **(A)** Tuning of the membrane time constant τ_m, **(B)** tuning of the leaky resistance *R*, and **(C)** adaptation of the Output firing rate.

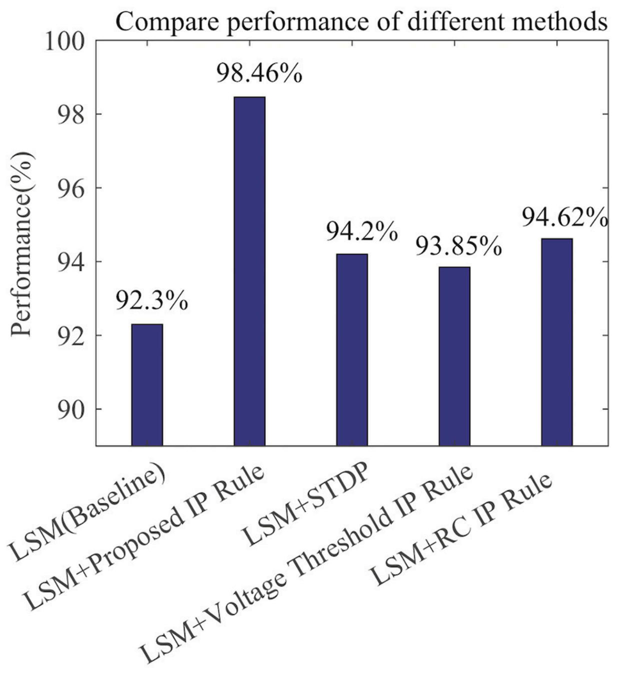


FIGURE 12 | Speech recognition performances of various learning rules when applied to a LSM with 135 reservoir neurons. The performance evaluation is based on the single-speaker subset of the TI46 Speech Corpus. (1) LSM (Baseline): with the settings and supervised readout learning rule in Zhang et al. (2015) and no reservoir tuning. All other compared networks add additional mechanisms to the baseline. (2) LSM+Proposed IP Rule: with additional reservoir neurons tuning using SpiKL-IP. (3) LSM+STDP: with additional reservoir neurons tuning using the STDP rule in Jin and Li (2016); (4) LSM+Voltage Threshold IP Rule: with additional reservoir neurons tuning using the IP rule in Lazar et al. (2007). (5) LSM+RC IP Rule: with additional reservoir neurons tuning using the IP rule in Li and Li (2013).

TABLE 2 | The performances of LSM-based image classification with and without the proposed SpiKL-IP rule evaluated using the CityScape image dataset.

Reservoir size	Without IP (%)	With IP (%)
135	96.60	97.78
72	94.90	96.48
45	91.74	94.44
27	87.33	90.19

More specifically, SpiKL-IP aims to tune the intrinsic parameters of a spiking neuron while minimizing the KL-divergence from the targeted exponential distribution to the actual output firing rate distribution. However, several challenges must be addressed as we work toward achieving the above goal. First, we rigorously relate the output firing rate with the static input current by deriving the firing-rate transfer function (FR-TF). FR-TF provides a basis for allowing the derivation of the SpiKL-IP rule that minimizes the KL-divergence. Furthermore, we cast SpiKL-IP in a suitable form to enable online application of IP tuning. Finally, we address one major challenge associated with applying SpiKL-IP under realistic contexts where the input current to each spiking neuron may be time-varying, which leads to the final IP rule that has no dependency on the instantaneous input level and effectively tuning the neural model parameters based upon averaged firing activities.

In the simulation studies, it is shown that SpiKL-IP can produce excellent performances. Under various settings, the application of SpiKL-IP to individual neurons in isolation or as part of a larger network robustly creates the desired exponential distribution for the output firing rate even when the input current is time varying. The evaluation of the learning performance of SpiKL-IP for real-world classification tasks also confirms the potential of the proposed IP rule. When applied to the reservoir neurons of LSM networks, SpiKL-IP produces significant performance boosts based on the TI46 Speech Corpus (Liberman et al., 1991) and the CityScape image dataset (Cordts et al., 2016).

Our future work will explore the potential of integrating IP tuning with Hebbian unsupervised learning mechanisms, particularly spike-timing-dependent plasticity (STDP). Jin and Li (2017) and this work respectively demonstrate that STDP and IP are effective in tuning recurrent spiking neural networks, i.e., reservoirs, and boosting the overall learning performance. Moreover, it has been suggested by Lazar et al. (2007) and Li et al. (2018) that STDP and IP may be complementary to each other. On the other hand, Watt and Desai (2010) and other related works reveal one limitation of STDP, i.e., the application of STDP can lead to network instability due to the positive feedback mechanisms created. Nevertheless, concerning the potential instability caused by STDP, it may be argued that the joint application of STDP and IP could be beneficial. This is because IP is intrinsically self-stabilizing, which may contribute to the prevention of runaway potentiation caused by STDP. We will also implement the SpiKL-IP rule on noisy leaky-integrate and fire neuron model (Brunel and Sergi, 1998) to evaluate the ability of the SpiKL-IP rule standing against noise. Moreover, since non-Hebbian plasticity and IP are supposed to work together in biological neurons (Watt and Desai, 2010), we can further explore the effects of combining Hebbian unsupervised plasticity, non-Hebbian plasticity, and intrinsic plasticity to maintain the homeostasis of networks.

AUTHOR CONTRIBUTIONS

WZ and PL developed the theoretical approach for IP tuning of spiking neurons and the SpiKL-IP rule. WZ implemented SpiKL-IP and related learning rules and performed the simulation studies. WZ and PL wrote the paper.

FUNDING

This work supported by the National Science Foundation (NSF) under Grant No.CCF-1639995 and the Semiconductor Research Corporation (SRC) under Task 2692.001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, SRC, Texas A&M University, and their contractors.

REFERENCES

Baddeley, R., Abbott, L. F., Booth, M. C., Sengpiel, F., Freeman, T., Wakeman, E. A., et al. (1997). Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proc. R. Soc. Lond. B Biol. Sci.* 264, 1775–1783. doi: 10.1098/rspb.1997.0246

Bell, A. J., and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Comput.* 7, 1129–1159. doi: 10.1162/neco.1995.7.6.1129

Brunel, N., and Sergi, S. (1998). Firing frequency of leaky intergrate-and-fire neurons with synaptic current dynamics. *J. Theor. Biol.* 195, 87–95. doi: 10.1006/jtbi.1998.0782

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., et al. (2016). "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3213–3223.

Dayan, P., and Abbott, L. F. (2001). *Theoretical Neuroscience*, Vol. 806. Cambridge, MA: MIT Press.

Desai, N. S., Rutherford, L. C., and Turrigiano, G. G. (1999). Plasticity in the intrinsic excitability of cortical pyramidal neurons. *Nat. Neurosci.* 2:515. doi: 10.1038/9165

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Jin, Y., and Li, P. (2016). "Ap-stdp: a novel self-organizing mechanism for efficient reservoir computing," in *Neural Networks (IJCNN), 2016 International Joint Conference on* (IEEE), 1158–1165.

Jin, Y., and Li, P. (2017). "Calcium-modulated supervised spike-timing-dependent plasticity for readout training and sparsification of the liquid state machine," in *Neural Networks (IJCNN), 2017 International Joint Conference on* (IEEE), 2007–2014.

Lazar, A., Pipa, G., and Triesch, J. (2007). Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Netw.* 20, 312–322. doi: 10.1016/j.neunet.2007.04.020

Legenstein, R., and Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Netw.* 20, 323–334. doi: 10.1016/j.neunet.2007.04.017

Li, C. (2011). A model of neuronal intrinsic plasticity. *IEEE Trans. Auton. Ment. Dev.* 3, 277–284. doi: 10.1109/TAMD.2011.2159379

Li, C., and Li, Y. (2013). A spike-based model of neuronal intrinsic plasticity. *IEEE Trans. Auton. Ment. Dev.* 5, 62–73. doi: 10.1109/TAMD.2012.2211101

Li, X., Wang, W., Xue, F., and Song, Y. (2018). Computational modeling of spiking neural network with learning rules from stdp and intrinsic plasticity. *Physica A* 491, 716–728. doi: 10.1016/j.physa.2017.08.053

Liberman, M., Amsler, R., Church, K., Fox, E., Hafner, C., Klavans, J., et al. (1991). TI 46-word LDC93S9.

Lyon, R. (1982). "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82*, Vol. 7 (IEEE), 1282–1285.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Netw.* 10, 1659–1671. doi: 10.1016/S0893-6080(97)00011-7

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

Marder, E., Abbott, L. F., Turrigiano, G. G., Liu, Z., and Golowasch, J. (1996). Memory from the dynamics of intrinsic membrane currents. *Proc. Natl. Acad. Sci. U.S.A.* 93, 13481–13486. doi: 10.1073/pnas.93.24.13481

Nadal, J.-P., and Parga, N. (1994). Nonlinear neurons in the low-noise limit: a factorial code maximizes information transfer. *Network* 5, 565–581. doi: 10.1088/0954-898X_5_4_008

Panda, P., and Roy, K. (2017). Learning to generate sequences with combination of hebbian and non-hebbian plasticity in recurrent spiking neural networks. *Front. Neurosci.* 11:693. doi: 10.3389/fnins.2017.00693

Papoulis, A., and Pillai, S. U. (2002). *Probability, Random Variables, and Stochastic Processes*. Tata McGraw-Hill Education.

Schrauwen, B., and Van Campenhout, J. (2003). "Bsa, a fast and accurate spike train encoding scheme," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, Vol. 4 (IEEE), 2825–2830.

Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J. J., and Stroobandt, D. (2008). Improving reservoirs using intrinsic plasticity. *Neurocomputing* 71, 1159–1171. doi: 10.1016/j.neucom.2007.12.020

Stemmler, M., and Koch, C. (1999). How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. *Nat. Neurosci.* 2:521. doi: 10.1038/9173

Triesch, J. (2005). "A gradient rule for the plasticity of a neuron's intrinsic excitability," in *International Conference on Artificial Neural Networks* (Springer), 65–70.

Watt, A. J., and Desai, N. S. (2010). Homeostatic plasticity and stdp: keeping a neuron's cool in a fluctuating world. *Front. Synaptic Neurosci.* 2:5. doi: 10.3389/fnsyn.2010.00005

Zhang, Y., Li, P., Jin, Y., and Choe, Y. (2015). A digital liquid state machine with biologically inspired learning and its application to speech recognition. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 2635–2649. doi: 10.1109/TNNLS.2015.2388544

Conflict of Interest Statement: 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.

frontiers in Neuroscience

ORIGINAL RESEARCH
published: 26 March 2019
doi: 10.3389/fnins.2019.00260

Demonstrating Advantages of Neuromorphic Computation: A Pilot Study

Timo Wunderlich[1]*, *Akos F. Kungl*[1]*, *Eric Müller*[1], *Andreas Hartel*[1], *Yannik Stradmann*[1], *Syed Ahmed Aamir*[1], *Andreas Grübl*[1], *Arthur Heimbrecht*[1], *Korbinian Schreiber*[1], *David Stöckel*[1], *Christian Pehle*[1], *Sebastian Billaudelle*[1], *Gerd Kiene*[1], *Christian Mauch*[1], *Johannes Schemmel*[1], *Karlheinz Meier*[1] and *Mihai A. Petrovici*[1,2]

[1] Department of Physics, Kirchhoff Institute for Physics, Heidelberg University, Heidelberg, Germany, [2] Department of Physiology, University of Bern, Bern, Switzerland

OPEN ACCESS

Edited by:
John V. Arthur,
IBM, United States

Reviewed by:
Timothée Masquelier,
Centre National de la Recherche Scientifique (CNRS), France
Terrence C. Stewart,
University of Waterloo, Canada

***Correspondence:**
Timo Wunderlich
timo.wunderlich@kip.uni-heidelberg.de
Akos F. Kungl
fkungl@kip.uni-heidelberg.de

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: 09 November 2018
Accepted: 05 March 2019
Published: 26 March 2019

Citation:
Wunderlich T, Kungl AF, Müller E, Hartel A, Stradmann Y, Aamir SA, Grübl A, Heimbrecht A, Schreiber K, Stöckel D, Pehle C, Billaudelle S, Kiene G, Mauch C, Schemmel J, Meier K and Petrovici MA (2019) Demonstrating Advantages of Neuromorphic Computation: A Pilot Study. Front. Neurosci. 13:260. doi: 10.3389/fnins.2019.00260

Neuromorphic devices represent an attempt to mimic aspects of the brain's architecture and dynamics with the aim of replicating its hallmark functional capabilities in terms of computational power, robust learning and energy efficiency. We employ a single-chip prototype of the BrainScaleS 2 neuromorphic system to implement a proof-of-concept demonstration of reward-modulated spike-timing-dependent plasticity in a spiking network that learns to play a simplified version of the Pong video game by smooth pursuit. This system combines an electronic mixed-signal substrate for emulating neuron and synapse dynamics with an embedded digital processor for on-chip learning, which in this work also serves to simulate the virtual environment and learning agent. The analog emulation of neuronal membrane dynamics enables a 1000-fold acceleration with respect to biological real-time, with the entire chip operating on a power budget of 57 mW. Compared to an equivalent simulation using state-of-the-art software, the on-chip emulation is at least one order of magnitude faster and three orders of magnitude more energy-efficient. We demonstrate how on-chip learning can mitigate the effects of fixed-pattern noise, which is unavoidable in analog substrates, while making use of temporal variability for action exploration. Learning compensates imperfections of the physical substrate, as manifested in neuronal parameter variability, by adapting synaptic weights to match respective excitability of individual neurons.

Keywords: BrainScaleS, mixed-signal, neuromorphic computing, spiking neural networks, reinforcement learning, STDP, plasticity

1. INTRODUCTION

Neuromorphic computing represents a novel paradigm for non-Turing computation that aims to reproduce aspects of the ongoing dynamics and computational functionality found in biological brains. This endeavor entails an abstraction of the brain's neural architecture that retains an amount of biological fidelity sufficient to reproduce its functionality while disregarding unnecessary detail. Models of neurons, which are considered the computational unit of the brain, can be emulated using electronic circuits or simulated using specialized digital systems (Indiveri et al., 2011; Furber, 2016).

BrainScaleS 2 (BSS2) is a neuromorphic architecture consisting of CMOS-based ASICs (Friedmann et al., 2017; Aamir et al., 2018) which implement physical models of neurons and synapses in analog electronic circuits while providing facilities for user-defined learning rules. A number of features distinguish BSS2 from other neuromorphic approaches, such as a speed-up factor of 10^3 compared to biological neuronal dynamics, correlation sensors for spike-timing-dependent plasticity in each synapse circuit and an embedded processor (Friedmann et al., 2017), which can use neural network observables to calculate synaptic weight updates for a broad range of plasticity rules. The flexibility enabled by the embedded processor is a particularly useful feature given the increasing effort invested in synaptic plasticity research, allowing future findings to be accommodated easily. The study at hand uses a single-chip prototype version of the full system, which allows the evaluation of the planned system design on a smaller scale.

Reinforcement learning has been prominently used as the learning paradigm of choice in machine learning systems which reproduced, or even surpassed, human performance in video and board games (Mnih et al., 2015; Silver et al., 2016, 2017). In reinforcement learning, an agent interacts with its environment and receives reward based on its behavior. This enables the agent to adapt its internal parameters so as to increase the potential for reward in the future (Sutton and Barto, 1998). In the last decades, research has found a link between reinforcement learning paradigms used in machine learning and reinforcement learning in the brain (for a review, see Niv, 2009). The neuromodulator dopamine was found to convey a reward prediction error, akin to the temporal difference error used in reinforcement learning methods (Schultz et al., 1997; Sutton and Barto, 1998). Neuromodulated plasticity can be modeled using three-factor learning rules (Frémaux et al., 2013; Frémaux and Gerstner, 2015), where the synaptic weight update depends not only on the learning rate and the pre- and post-synaptic activity but also on a third factor, representing the neuromodulatory signal, which can be a function of reward, enabling reinforcement learning.

In this work, we demonstrate the advantages of neuromorphic computation by showing how an agent controlled by a spiking neural network (SNN) learns to solve a smooth pursuit task via reinforcement learning in a fully embedded perception-action loop that simulates the classic Pong video game on the BSS2 prototype. Measurements of time-to-convergence, power consumption, and sensitivity to parameter noise demonstrate the advantages of our neuromorphic solution compared to classical simulation on a modern CPU that runs the NEST simulator (Peyser et al., 2017). The on-chip learning converges within seconds, which is equivalent to hours in biological terms, while the software simulation is at least an order of magnitude slower and three orders of magnitude less energy-efficient. We find that fixed-pattern noise on BSS2 can be compensated by the chosen learning paradigm, reducing the required calibration precision, and that the results of hyperparameter learning can be transferred between different BSS2 chips. The experiment takes place on the chip fully autonomously, i.e., both the environment and synaptic weight changes are computed using the embedded processor. As the number of neurons (32) and synapses (1,024) on the prototype chip constrain the complexity of solvable learning tasks, the agent's task in this work is simple smooth pursuit without anticipation. The full system is expected to enable more sophisticated learning, akin to the learning of Pong from pixels that was previously demonstrated using an artificial neural network (Mnih et al., 2015).

2. MATERIALS AND METHODS

2.1. The BrainScaleS 2 Neuromorphic Prototype Chip

The BSS2 prototype is a neuromorphic chip and the predecessor of a large-scale accelerated network emulation platform with flexible plasticity rules (Friedmann et al., 2017). It is manufactured using a 65 *nm* CMOS process and is designed for mixed-signal neuromorphic computation. All experiments in this work were performed on the second prototype version. Future chips will be integrated into a larger setup using wafer-scale technology (Schemmel et al., 2010; Zoschke et al., 2017), thereby enabling the emulation of large plastic neural networks.

2.1.1. Experimental Setup

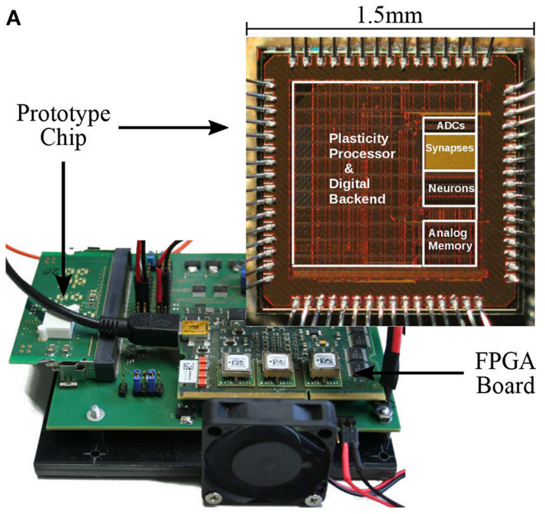
The BSS2 prototype setup is shown in **Figure 1A** and contains the neuromorphic chip mounted on a prototyping board. The chip and all of its functional units can be accessed and configured from either a Xilinx Spartan-6 FPGA or the embedded processor (see section 2.1.4). The FPGA in turn can be accessed via a USB-2.0 connection between the prototype setup and the host computer. In addition to performing chip configuration, the FPGA can also provide hard real-time playback of input and recording of output data.

Experiments are described by the user through a container-based programming interface which provides access to all functional units such as individual neuron circuits or groups of synapses. The experiment configuration is transformed into a bitstream and uploaded to DRAM attached to the FPGA. Subsequently, the software starts the experiment and a sequencer logic in the FPGA begins to play back the experiment data (e.g., input spike trains) stored in the DRAM. At the same time, output from the chip is recorded to a different memory area in the DRAM. Upon completion of the experiment, the host computer downloads all recorded output from the FPGA memory.

2.1.2. Neurons and Synapses

Our approach to neuromorphic engineering follows the idea of "physical modeling": the analog neuronal circuits are designed to have similar dynamics compared to their biological counterparts, making use of the physical characteristics of the underlying substrate. The BSS2 prototype chip contains 32 analog neurons based on the Leaky Integrate-and-Fire (LIF) model (Aamir et al., 2016, 2018). Additionally, each neuron has an 8-bit spike counter, which can be accessed and reset by the embedded processor (Friedmann et al., 2017, see section 2.1.4) for plasticity-related calculations.

In contrast to other neuromorphic approaches (Benjamin et al., 2014; Furber et al., 2014; Merolla et al., 2014; Qiao et al., 2015; Davies et al., 2018), this implementation uses the fast supra-threshold dynamics of CMOS transistors in circuits which mimic neuronal membrane dynamics. In the case of BSS2, this approach provides time constants that are smaller than their biological



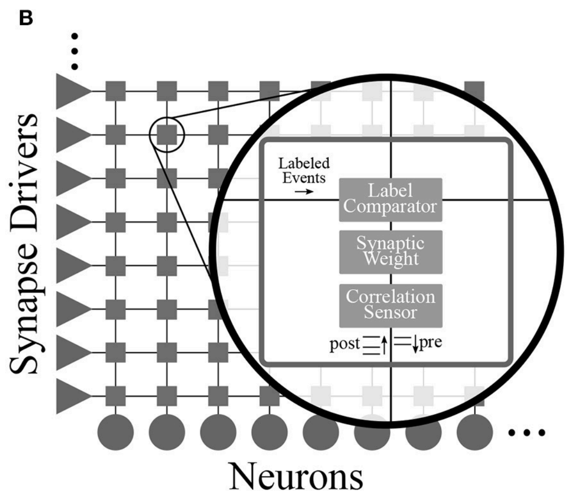


FIGURE 1 | Physical setup and neural network schematic. **(A)** In the foreground: BSS2 prototype chip with demarcation of different functional parts. In the background: the development board on which the chip is mounted. Adapted from Aamir et al. (2018). **(B)** Schematic of the on-chip neural infrastructure. Each of the 32 implemented neurons is connected to one column of the synapse array, where each column comprises 32 synapses. Synapse drivers allow row-wise injection of individually labeled (6-bit) spike events. Each synapse locally stores a 6-bit label and a 6-bit weight and converts spike events with a matching label to current pulses traveling down toward the neuron. Each synapse also contains an analogue sensor measuring the temporal correlation of pre- and post-synaptic events (see section 2.1.5).

counterparts by three orders of magnitude, i.e., the hardware operates with a speed-up factor of 10^3 compared to biology, independent of the network size or plasticity model. Throughout the manuscript, we provide the true (wall-clock time) values, which are typically on the order of microseconds, compared to the millisecond-scale values usually found in biology.

The 32-by-32 array of synapses is arranged such that each neuron can receive input from a column of 32 synapses (see **Figure 1B**). Each row consisting of 32 synapses can be individually configured as excitatory or inhibitory and receives input from a synapse driver that injects labeled digital pre-synaptic spike packets. Every synapse compares its label (a locally stored configurable address) with the label of a given spike packet and if they match, generates a current pulse with an amplitude proportional to its 6-bit weight that is sent down along the column toward the post-synaptic neuron. There, the neuron circuit converts it into an exponential post-synaptic current (PSC), which is injected into the neuronal membrane capacitor.

Post-synaptic spikes emitted by a neuron are signaled (back-propagated) to every synapse in its column, which allows the correlation sensor in each synapse to record the time elapsed between pre- and post-synaptic spikes. Thus, each synapse accumulates correlation measurements that can be read out by the embedded processor, to be used, among other observables, for calculating weight updates (see section 2.1.5 for a detailed description).

2.1.3. Calibration and Configuration of the Analog Neurons

Neurons are configured using on-chip analog capacitive memory cells (Hock et al., 2013). The ideal LIF model neuron with one synapse type and exponential PSCs can be characterized by six parameters: membrane time constant τ_{mem}, synaptic time constant τ_{syn}, refractory period τ_{ref}, resting potential v_{leak}, threshold potential v_{thresh}, reset potential v_{reset}. The neuromorphic implementation on the chip carries 18 tunable parameters per neuron and one global parameter (Aamir et al., 2018). Most of these hardware parameters are used to set the circuits to the proper point of operation and therefore have fixed values that are calibrated once for any given chip; for the experiments described here, the six LIF model parameters mentioned above are fully controlled by setting only six of the hardware parameters per neuron.

Manufacturing variations cause fixed-pattern noise (see section 2.2), therefore each neuron circuit behaves differently for any given set of hardware parameters. In particular, the time constants (τ_{mem}, τ_{syn}, τ_{ref}) display a high degree of variability. Therefore, in order to accurately map user-defined LIF time constants to hardware parameters, neuron circuits are calibrated individually. Using this calibration data reduces deviations from target values to $< 5\,\%$ (Aamir et al., 2018, see also **Figure 2**).

2.1.4. Plasticity Processing Unit

To allow for flexible implementation of plasticity algorithms, the chip uses a Plasticity Processing Unit (PPU), which is a general-purpose 32-bit processor implementing the PowerPC-ISA 2.06 instruction set and custom vector extensions (Friedmann et al., 2017). In the used prototype chip, it is clocked at a frequency of 98 *MHz* and has access to 16 *KiB* of main memory. Vector registers are 128-bit wide and can be processed in slices of eight 16-bit or sixteen 8-bit units within one clock cycle. The vector extension unit is loosely coupled to the general-purpose part.

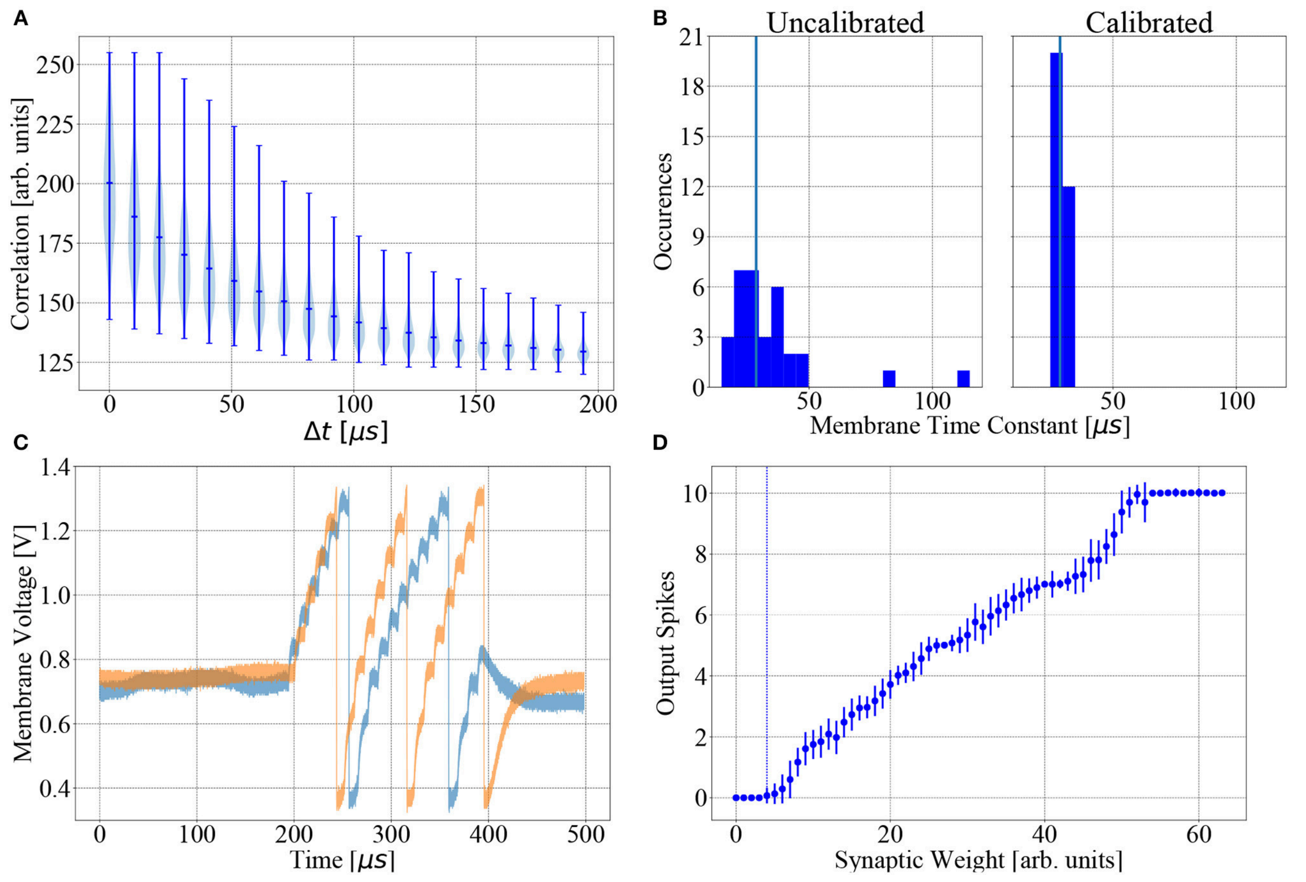


FIGURE 2 | BSS2 is subject to fixed-pattern noise and temporal variability. **(A)** Violin plot of the digitized output of the 1024 causal correlation sensors (a_+, see Equation 1) on a sample chip (chip #1) as a function of the time interval between a single pre-post spike pair. **(B)** Distribution of membrane time constants τ_m over all 32 neurons with and without calibration. The target value is 28.5 μs (vertical blue lines). **(C)** Effects of temporal variability. A regular input spike train containing twenty spikes spaced by 10 μs, as used in the learning task, transmitted via one synapse, elicits different membrane responses in two trials. **(D)** Mean and variance of the output spike count as a function of synaptic weight, averaged over 100 trials, for a single exemplary neuron receiving the input spike train from **(C)**. The spiking threshold weight (the smallest weight with a higher than 5 % probability of eliciting an output spike under the given stimulation paradigm) is indicated by the dotted blue line. Trial-to-trial variation of the number of output spikes at fixed synaptic weight is due to temporal variability and mediates action exploration.

When fetching vector instructions, the commands are inserted into a dedicated command queue which is read by the vector unit. Vector commands are decoded, distributed to the arithmetic units and executed as fast as possible.

The PPU has dedicated fully-parallel access ports to synapse rows, enabling row-wise readout and configuration of synaptic weights and labels. This enables efficient row-wise vectorized plasticity processing. Modifications of connectivity, neuron and synapse parameters are supported during neural network operation. The PPU can be programmed using assembly and higher-level languages such as C or C++ to compute a wide range of plasticity rules. Compiler support for the PPU is provided by a customized `gcc` (Electronic Vision(s), 2017; Stallman and GCC Developer Community, 2018). The software used in this work is written in C, with vectorized plasticity processing implemented using inline assembly instructions.

2.1.5. Correlation Measurement at the Synapses

Every synapse in the synapse array contains two analog units that record the temporal correlation between nearest-neighbor pairs of pre- and post-synaptic spikes. For each such pair, a dedicated circuit measures the value of either the causal (pre before post) or anti-causal (post before pre) correlation, which is modeled as an exponentially decaying function of the spike time difference (Friedmann et al., 2017). The values thus measured are accumulated onto two separate storage capacitors per synapse. In an idealized model, the voltages across the causal and anti-causal storage capacitor are

$$a_+ = \sum_{\text{pre}-\text{post}} \eta_+ \exp\left(-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau_+}\right) \tag{1}$$

and

$$a_- = \sum_{\text{post}-\text{pre}} \eta_- \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_-}\right), \tag{2}$$

respectively, with decay time constants τ_+ and τ_- and scaling factors η_+ and η_-. These accumulated voltages represent non-decaying eligibility traces that can be read out by the

PPU using column-wise 8-bit Analog-to-Digital Converters (ADCs), allowing row-wise parallel readout. Fixed-pattern noise introduces variability among the correlation units of different synapses, as visible in **Figure 2A**. The experiments described here only use the causal traces a_+ to calculate weight updates.

2.2. Types of Noise on BSS2

The BSS2 prototype has several sources of parameter variability and noise, as does any analog hardware. We distinguish between fixed-pattern noise and temporal variability.

Fixed-pattern noise refers to the systematic deviation of parameters (e.g., transistor parameters) from the values targeted during chip design. This type of noise is caused by the inaccuracies of the manufacturing process and unavoidable stochastic variations of process parameters. Fixed-pattern noise is constant in time and induces heterogeneity between neurons and synapses, but calibration can reduce it to some degree. The effects of the calibration on the distribution of the membrane time constant τ_{mem} are shown in **Figure 2B**.

Temporal variability continually influences circuits during their operation, leading to fluctuations of important dynamical variables such as membrane potentials. Typical sources of temporal variability are crosstalk, thermal noise and the limited stability of the analog parameter storage. These effects can cover multiple timescales and lead to variable neuron spike responses, even when the input spike train remains unchanged between trials. A concrete example of trial-to-trial variability of a neuron's membrane potential evolution, output spike timing and firing rate caused by temporal variability is shown in **Figures 2C,D** for two trials of the same experiment, using the same input spike train and parameters, with no chip reconfiguration between trials.

2.3. Reinforcement Learning With Reward-Modulated STDP

In reinforcement learning, a behaving agent interacts with its environment and tries to maximize the expected future reward it receives from the environment as a consequence of this interaction (Sutton and Barto, 1998). The techniques developed to solve problems of reinforcement learning generally do not involve spiking neurons and are not designed to be biologically plausible. Yet reinforcement learning evidently takes place in biological SNNs, e.g., in basic operant conditioning (Guttman, 1953; Fetz and Baker, 1973; Moritz and Fetz, 2011). The investigation of spike-based implementations with biologically inspired plasticity rules is therefore an interesting subject of research with evident applications for neuromorphic devices. The learning rule used in this work, Reward-modulated Spike-Timing Dependent Plasticity (R-STDP) (Farries and Fairhall, 2007; Izhikevich, 2007; Frémaux et al., 2010), represents one possible implementation.

R-STDP is a three-factor learning rule that modulates the effect of unsupervised STDP using a reward signal. Recent work has used R-STDP to reproduce Pavlovian conditioning as in Izhikevich (2007) on a specialized neuromorphic digital simulator, achieving real-time simulation speed (Mikaitis et al., 2018). While not yet directly applied to an analog neuromorphic substrate, aspects of this learning paradigm have already been studied in software simulations, under constraints imposed by the BSS2 system, in particular concerning the effect of discretized weights, with promising results (Friedmann et al., 2013). Furthermore, it was previously suggested that trial-to-trial variations of neuronal firing rates as observed in cortical neurons can benefit learning in a reinforcement learning paradigm, rather than being a nuisance (Xie and Seung, 2004; Legenstein et al., 2008; Maass, 2014). Our experiments corroborate this hypothesis by explicitly using trial-to-trial variations due to temporal variability for action exploration.

The reward mechanism in R-STDP is biologically inspired: the phasic activity of dopamine neurons in the brain was found to encode expected reward (Schultz et al., 1997; Hollerman and Schultz, 1998; Bayer and Glimcher, 2005) and dopamine concentration modulates STDP (Pawlak and Kerr, 2008; Edelmann and Lessmann, 2011; Brzosko et al., 2015). R-STDP and similar reward-modulated Hebbian learning rules have been used to solve a variety of learning tasks in simulations, such as reproducing temporal spike patterns and spatio-temporal trajectories (Farries and Fairhall, 2007; Vasilaki et al., 2009; Frémaux et al., 2010), reproducing the results of classical conditioning (Izhikevich, 2007), making a recurrent neural network exhibit specific periodic activity and working-memory properties (Hoerzer et al., 2014) and reproducing the seminal biofeedback experiment by Fetz and Baker (Fetz and Baker, 1973; Legenstein et al., 2008). Compared to classic unsupervised STDP, using R-STDP was shown to improve the performance of a spiking convolutional neural network tasked with visual categorization (Mozafari et al., 2018a,b).

In contrast to other learning rules in reinforcement learning, R-STDP is not derived using gradient descent on a loss function; rather, it is motivated heuristically (Frémaux and Gerstner, 2015), the idea being to multiplicatively modulate STDP using a reward term.

We employ the following form of discrete weight updates using R-STDP:

$$\Delta w_{ij} = \beta \cdot (R - b) \cdot e_{ij}\,, \tag{3}$$

where β is the learning rate, R is the reward, b is a baseline and e_{ij} is the STDP eligibility trace which is a function of the pre- and post-synaptic spikes of the synapse connecting neurons i and j. The choice of the baseline reward b is critical: a non-zero offset introduces an admixture of unsupervised learning via the unmodulated STDP term, and choosing b to be the task-specific expected reward $b = \langle R \rangle_{\mathrm{task}}$ leads to weight updates that capture the covariance of reward and synaptic activity (Frémaux and Gerstner, 2015):

$$\langle \Delta w_{ij} \rangle_{\mathrm{task}} = \langle R \cdot e_{ij} \rangle_{\mathrm{task}} - \langle R \rangle_{\mathrm{task}} \cdot \langle e_{ij} \rangle_{\mathrm{task}} = \mathrm{Cov}(R, e_{ij})\,. \tag{4}$$

This setting, which we also employ in our experiments, makes R-STDP a statistical learning rule in the sense that it captures correlations of joint pre- and post-synaptic activity and reward; this information is collected over many trials of any single learning task. The expected reward may be estimated as a moving average of the reward over the last trials of that specific task; task specificity of the expected reward is required when multiple tasks need to be learned in parallel (Frémaux et al., 2010).

2.4. Learning Task and Simulated Environment

Using the PPU, we simulate a simple virtual environment inspired by the Pong video game. The components of the game are a two-dimensional square playing field, a ball and the player's paddle (see **Figure 3A**). The paddle is controlled by the chip and the goal of the learning task is to trace the ball. Three of the four playing field sides are solid walls, with the paddle moving along the open side. The experiment proceeds iteratively and fully on-chip (see **Figure 3B**). A single experiment iteration consists of neural network emulation, weight modification and environment simulation. We visualize one iteration as a flowchart in **Figure 4** and provide a detailed account in the following.

The game dynamics consist of the ball starting in the middle of the playing field in a random direction with velocity $\vec{v}$ and the paddle which moves with velocity v_p along its baseline. Surfaces elastically reflect the ball. If the paddle misses the ball, the game is reset, i.e., the ball starts from the middle of the field in a random direction. Specific parameter values are given in **Table 1**.

The paddle is controlled by the neural network emulated on the neuromorphic chip. The network receives the ball position along the paddle's baseline as input and determines the movement of its paddle via the resulting network activity. The neural network consists of two 32-unit layers which are initialized with all-to-all feed-forward connections (see **Table 1**), where the individual weights are drawn from a Gaussian distribution and where the first layer represents input/state units u_i and the second layer represents output/action units v_i. The first layer is a virtual layer for providing input spikes and the second layer consists of the chip's LIF neurons. All synaptic connections are excitatory. Discretizing the playing field into 32 columns along the paddle baseline, we assign a state unit u_i to each column i where $i \in [0, 31]$ and provide input to this unit if the ball is in the corresponding column via a uniform spike train (see **Table 1** for parameters).

The action neurons' spike counts ρ_i are used to determine the paddle movement: the unit with the highest number of output spikes $j = \mathrm{argmax}_i(\rho_i)$ determines the paddle's target column j, toward which it moves with constant velocity v_p (see **Table 1**). If the target column and center position of the paddle match, no movement is performed. Spike counts and in consequence, the target row j are determined after the input spike train has been delivered. If several output units have the same spike count, the winner is chosen randomly among them. Afterwards, the reward R is calculated based on the distance between the target column j and the current column of the ball, k:

$$R = \begin{cases} 1 - |j-k| \cdot 0.3 & \text{if } |j-k| \le 3\,, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Learning success is therefore graded, i.e., the network obtains reduced reward for less than optimal aiming. The size of the

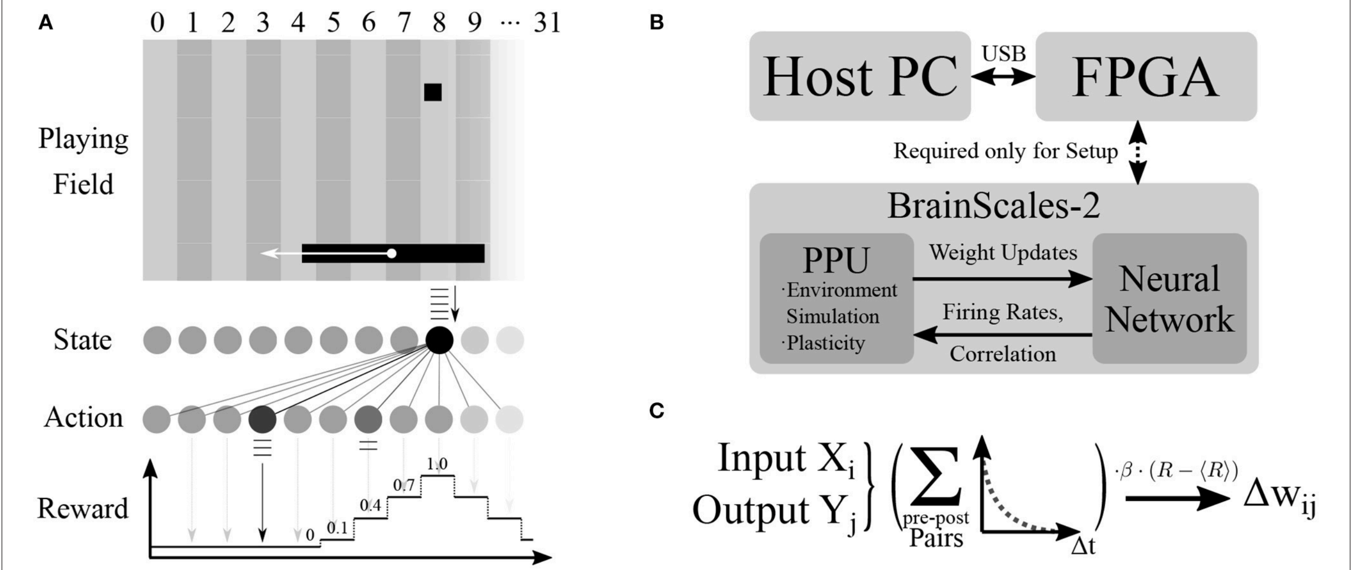


FIGURE 3 | Overview of the experimental setup. **(A)** The components of the environment are the playing field with three reflective walls (top, left, right), the ball and the paddle. In the depicted situation, the ball is in column 8 and therefore a uniform spike train is sent to output neurons via the synapses of input unit 8. Output unit 3 fires the most spikes and the paddle therefore moves toward column 3. As the target column is too far away from the ball column, it lies outside of the graded reward window and the reward received by the network is zero (see Equation 5). **(B)** The chip performs the experiment completely autonomously: the environment simulation, spiking network emulation and synaptic plasticity via the PPU are all handled on-chip. The FPGA, which is only used for the initial configuration of the chip, is controlled via USB from the host PC. **(C)** The plasticity rule calculated by the PPU. Pre-before-post spike pairs of input unit X_i and output unit Y_j are exponentially weighted with the corresponding temporal distance and added up. This correlation measure is then multiplied with the learning rate and the difference of instantaneous and expected reward to yield the weight change for synapse (i, j).

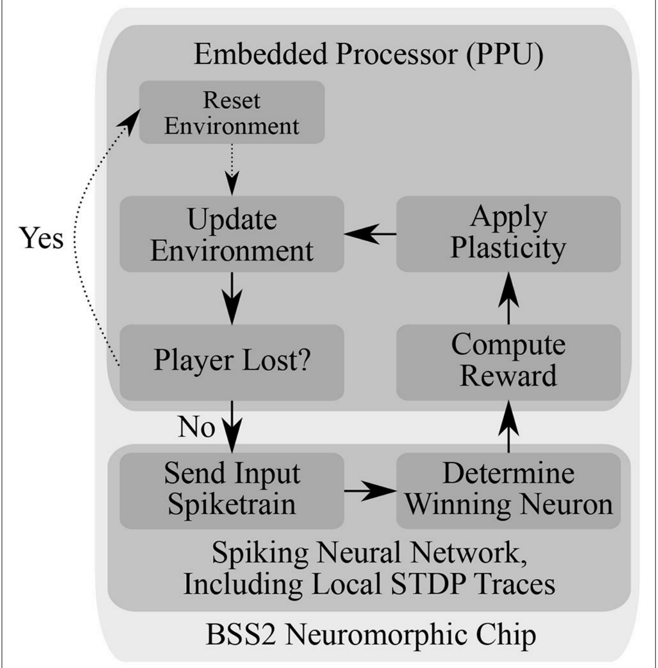


FIGURE 4 | Flowchart of the experiment loop running autonomously on BSS2, using both the analogue Spiking Neural Network (SNN) and the embedded processor. The environment is reset by positioning the ball in the middle of the playing field with a random direction of movement at the start of the experiment or upon the agent's failure to reflect the ball. In the main loop, the (virtual) state unit corresponding to the current ball position transmits a spike train to all neurons in the action layer (see **Figure 3**). Afterwards, the winning action neuron, i.e., the action neuron that had the highest output spike count, is determined. Then, the reward is determined based on the difference between the ball position and the target paddle position as dictated by the winning neuron (Equation 5). Using the reward, a stored running average of the reward and the STDP correlation traces computed locally at each synapse during SNN emulation, the weight updates of all synapses are computed (Equation 7) and applied. The environment, i.e., the position of ball and paddle, are then updated, and the loop starts over.

reward window defined by Equation (5) is chosen to match the paddle length.

For every possible task (corresponding to a ball column k), the PPU holds task-specific expected rewards $\bar{R}_k$, $k \in [0, 31]$ in its memory, which it subtracts from the instantaneous reward R to yield the neuromodulating factor $R - \bar{R}_k$, which is also used to update the task-specific expected reward as an exponentially weighted moving average:

$$\bar{R}_k \leftarrow \bar{R}_k + \gamma \left(R - \bar{R}_k\right) , \tag{6}$$

where γ controls the influence of previous iterations. The expected reward of any state is initialized as the first reward received in that state. All 1,024 synapse weights w_{mn} from input unit m to output unit n are then updated according to the three-factor rule (see **Figure 3C**)

$$\Delta w_{mn} = \beta \cdot \left(R - \bar{R}_k\right) \cdot A^{+}_{mn} , \tag{7}$$

where β is a learning rate and A^{+}_{mn} is a modified version of a^{+}_{mn} (see Equation 1) which has been corrected for offset, digitized to 8 bit and right-shifted by one bit in order to reduce noise. After the weights have been modified, the Pong environment is updated according to the described dynamics and the next iteration begins.

The *mean expected reward*

$$\langle \bar{R} \rangle = \frac{1}{32} \sum_{i=0}^{31} \bar{R}_i , \tag{8}$$

i.e., the average of the expected rewards over all states, represents a measure of the progress of learning in any given iteration. Due to the paddle width and correspondingly graded reward scheme, the agent is able to catch the ball even when it is not perfectly centered below the ball. Therefore, the *performance* in playing Pong can be quantified by

$$P = \frac{1}{32} \sum_{i=0}^{31} \lceil R_i \rceil , \tag{9}$$

where R_i is the last reward received in state i. This provides the percentage of states in which the agent has aimed the paddle such that it is able to catch the ball.

In order to find suitable hyperparameters for the chip configuration, we used a Bayesian parameter optimization based on sequential optimization using decision trees to explore the neuronal and synaptic parameter space while keeping parameters such as game dynamics, input spike train and initial weight distribution fixed. The software package used was scikit-optimize (Head et al., 2018). Initially, 30 random data points were taken, followed by 300 iterations with the next evaluated parameter set being determined by the optimization algorithm FOREST_MINIMIZE with default parameters (maximizing expected improvement, extra trees regressor model, 10,000 acquisition function samples, target improvement of 0.01). All neuron and synapse parameters were subject to the optimization, i.e., all neuronal time constants and voltages as well as the time constant and amplitude of the synapse correlation sensors. The results are a common set of parameters for all neurons and synapses (see **Table 1**).

2.5. Software Simulation With NEST

In order to compare the learning performance and speed of the chip to a network of deterministic, perfectly identical LIF neurons, we ran a software simulation of the experiment using the NEST v2.14.0 SNN simulator (Peyser et al., 2017), using the same target LIF parameters as in our experiments using the chip, with time constants scaled by a factor of 10^3. We did not include fixed-pattern noise of neuron parameters in the simulation, i.e., all neurons had identical parameters. We used the *iaf_psc_exp* integrate-and-fire neuron model available in NEST, with exponential PSC kernels and current-based synapses. Using NEST's *noise_generator*, we are able to investigate the effect of injecting Gaussian current noise into each neuron. The scaling factors η_+ and η_-, as well as the time constants τ_+ and τ_- of

TABLE 1 | Parameters used in the experiment.

Symbol	Description	Value
	Neuromorphic hardware	BrainScaleS 2 (2nd prototype version)
N	Number of action/output neurons (LIF)	32
N_{S}	Number of state/input units	32
N_{syn}	Number of synapses	$32 \cdot 32 = 1024$
N_{spikes}	Number of spikes from input unit	20
T_{ISI}	ISI of spikes from input unit	10 µs
w	Mean of distribution of initial weights (digital value)	14
σ_w	Standard deviation of distribution of initial weights	2
L	Length and width of quadratic playing field	1
$\Vert\vec{v}\Vert_1$	L1-norm of ball velocity	0.025 per iteration
v_{p}	Velocity of paddle controlled by BSS2	0.05 per iteration
r_{b}	Radius of ball	0.02
r_{p}	Length of paddle	0.20
γ	Decay constant of reward	0.5
β	Learning rate	0.125
	NEST version (software simulation)	2.14.0
	NEST timestep	0.1 ms
	CPU (software simulation, one core used)	Intel i7-4771

		Set #1 (standard)	Set #2	Set #3
τ_{mem}	LIF membrane time constant	28.5 µs	18.4 µs	24.8 µs
τ_{ref}	LIF refractory time constant	4 µs	14.3 µs	13.8 µs
τ_{syn}	LIF excitatory synaptic time constant	1.8 µs	2.4 µs	1.4 µs
v_{leak}	LIF leak voltage	0.62 V	0.56 V	0.87 V
v_{reset}	LIF reset voltage	0.36 V	0.36 V	0.30 V
v_{thresh}	LIF threshold voltage	1.28 V	1.31 V	1.21 V
η_+	Amplitude of correlation function a_+ (digital value)	72	114	70
τ_+	Time constant of correlation function a_+	64 µs	80 µs	60 µs

The different parameter sets are the result of optimizing parameters on three different chips. If not mentioned otherwise, results were obtained using set #1. LIF, Leaky Integrate-and-Fire; ISI, Inter-Spike Interval.

the correlation sensors were chosen to match the mean values on BSS2. The correlation factor a_+ was calculated within the Python script controlling the experiment using Equation (1) and the spike times provided by the NEST simulator. Hyperparameters such as learning rate and game dynamics (e.g., the reward window defined in Equation 5) were set to be equivalent to BSS2 and weights were scaled to a dimensionless quantity and discretized to match the neuromorphic emulation.

The synaptic weight updates in each iteration were restricted to those synapses which transmitted spikes, i.e., the synapses from the active input unit to all output units (32 out of the $1,024$ synapses), as the correlation a_+ of all other synapses is zero in a perfect simulation without fixed-pattern noise. This has the effect of reducing the overall time required to simulate one iteration and is in contrast to the implementation on BSS2, where all synapses are updated in each iteration as there is no guarantee that correlation traces are zero and we excluded this kind of "expert knowledge" from the implementation.

The source code of the simulation is publicly available (Wunderlich, 2019).

3. RESULTS

3.1. Learning Performance

The progress of learning over 10^5 iterations is shown in **Figure 5** for both BSS2 (subplot A) and an ideal software simulation with and without injected noise (subplot B). We use both measures described above to quantify the agent's success: the mean expected reward (Equation 8) reflects the agent's aiming accuracy and the Pong performance (Equation 9) represents the ability of the agent to catch the ball using its elongated paddle. By repeating the procedure with ten randomly initialized weight matrices, we show that learning is reproducible, with little variation in the overall progress and outcome.

The optimal solution of the learning task is a one-to-one mapping of states to actions that place the center of the paddle directly below the ball at all times. In terms of the neural network, this means that the randomly initialized weight matrix should be dominated by its diagonal elements. We show the weight matrix on BSS2 after 10^5 learning iterations, averaged over the ten different trials depicted in **Figures 5**, **6A**. As expected, the

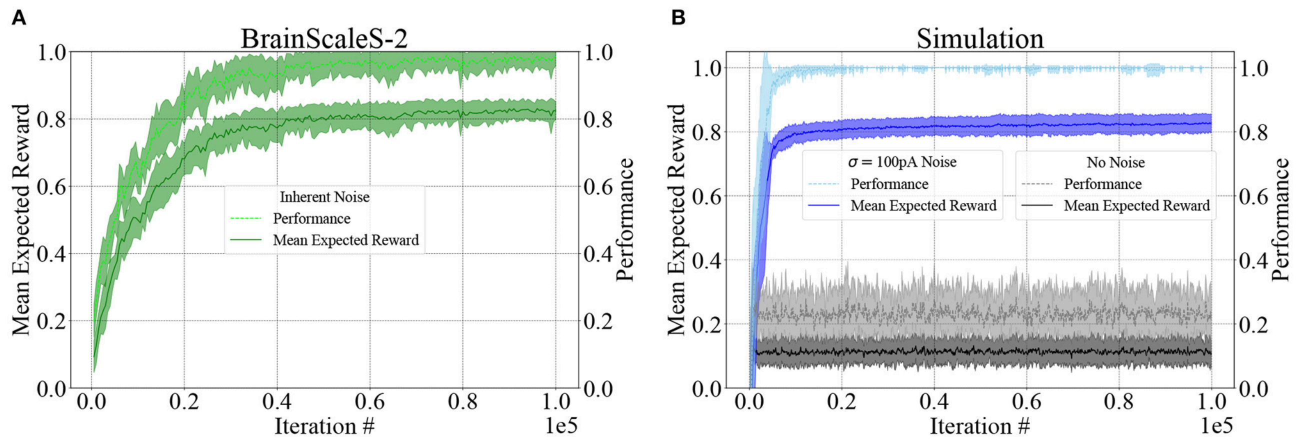


FIGURE 5 | Learning results for BSS2 and the software simulation using NEST, in terms of Mean expected reward (Equation 8) and Pong performance (Equation 9). In both cases, we plot the mean and standard deviation (shaded area) of 10 experiments. **(A)** BSS2 uses its intrinsic noise as an action exploration mechanism that drives learning. **(B)** The software simulation without noise is unable to learn and does not progress beyond chance level. Adding Gaussian zero-mean current noise with $\sigma = 100$ pA to each neuron allows the network to explore actions and enables learning. The simulation converges faster due to the idealized simulated scenario where no fixed-pattern noise is present.

diagonals of the matrix are dominant. Note that also slightly off-diagonal synapses are also strengthened, as dictated by the graded reward scheme. The visibly distinguishable vertical lines stem from neuronal fixed-pattern noise, as learning adapts weights to compensate for neuronal variability and one column in the weight matrix corresponds to one neuron.

A screen recording of a live demonstration of the experiment is available at https://www.youtube.com/watch?v=LW0Y5SSIQU4 and allows the viewer to follow the learning progress in a single experiment.

3.1.1. Temporal Variability on BSS2 Causes Exploration

On BSS2, action exploration and thereby learning is driven by trial-to-trial variations of neuronal spike counts that are due to temporal variability (see **Figures 2C,D**).

In contrast, we find that the software simulation without injected noise (see section 2.5) is unable to progress beyond the mean expected reward received by an agent choosing random actions, which is around $\langle \bar{R} \rangle = 0.1$ (the randomness in the weight matrix initialization leads to some variation in the mean expected reward after learning). The only non-deterministic mechanism in the software simulation without noise is due to the fact that if several action neurons elicit the same number of spikes, the winner is chosen randomly among them, but its effects are negligible. Injecting Gaussian current noise with zero mean and a standard deviation of $\sigma = 100\,pA$ into each neuron independently enables enough action exploration to converge to similar performance as BSS2. Compared to BSS2, the simulation with injected noise converges faster and to a higher level of Pong performance; this is due to the fact that the simulation contains no fixed-pattern noise, the network starts from an unbiased, perfectly balanced state and that Gaussian current noise is not an exact model of the temporal variability found in BSS2.

We can therefore conclude that under an appropriate learning paradigm, analog-hardware-specific temporal variability that is generally regarded as a nuisance can become a useful feature. Adding an action exploration mechanism similar to ϵ-greedy action selection would enable the software simulation to learn with guaranteed convergence to optimal performance, but would come at the cost of additional computational resources required for emulating such a mechanism.

3.2. Learning Is Calibration

The learning process adjusts synaptic weights such that individual differences in neuronal excitability on BSS2 are compensated. We correlated learned weights to neuronal properties and found that learning shapes a weight matrix that is adapted to a specific pattern of neuronal variability.

Each neuron is the target of 32 synapses. A subset of these are systematically potentiated, as they correspond to actions yielding a reward higher than the current expected reward, while the remainder is depressed, as these synapses correspond to actions yielding less reward. This leads to the diagonally-dominant matrix depicted in **Figure 6A**. At the same time, neuronal variability leads to different spiking-threshold weights, i.e., the smallest synaptic weight for which the neuron elicits one spike with a probability higher than 5 %, given the fixed input spike train used in the experiment (see **Figure 2D**). The learning process pushes the weights of unrewarded ($R = 0$) synapses below the spiking threshold of the respective neuron, thereby compensating variations of neuronal excitability, leading to a correlation of both quantities. Using the weights depicted in (A) and empirically determined spiking thresholds, we found that for each neuron the weights of synapses which correspond to unrewarded actions were correlated with the spiking-threshold with a correlation coefficient of $r = 0.76$ (see **Figure 6B**).

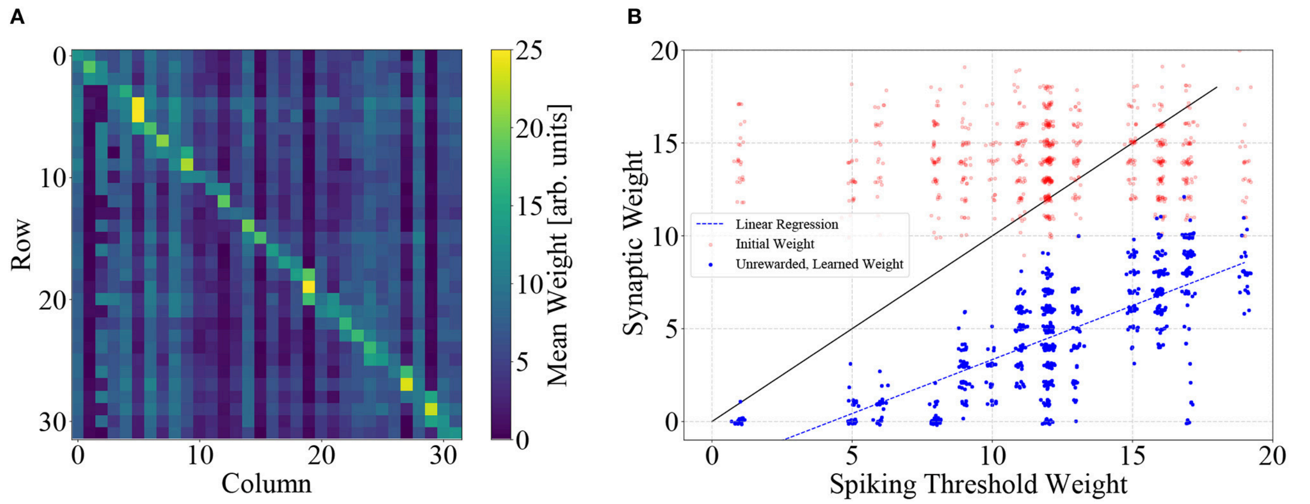


FIGURE 6 | (A) Synaptic weight matrix after learning, averaged over the 10 trials depicted in **Figure 5**. Weights on and near the diagonal dominate, corresponding to the goal of the learning task. The noticeable vertical stripes are a direct consequence of learning, which implicitly compensates neuronal variability (each column represents all input synapses of an action neuron). **(B)** Learning compensates circuit variability. Weights corresponding to unrewarded actions ($R = 0$) are systematically pushed below the threshold weight of the respective neuron, i.e., below the main diagonal. This leads to a correlation of learned weight and threshold (Pearson's $r = 0.76$). Weights are plotted with slight jitter along both axes for better visibility.

The learned adaptation of the synaptic weights to neuronal variability can be disturbed by randomly shuffling the assignment of logical action units to physical neurons. This is equivalent to the thought experiment of physically shuffling neurons on the chip and leads to synaptic weights which are maladapted to their physical substrate, i.e., efferent neuronal properties. In the following, we demonstrate the detrimental effects of such neuronal permutations and that the system can recover performance by subsequent learning.

We considered a weight matrix after 50,000 learning iterations and measured the resulting reward distribution over 100 experiments with learning turned off. Here, "reward distribution" refers to the distribution of the most recent reward over all 32 states. The top panel of **Figure 7A** shows the reward distribution for this baseline measurement. Mean expected reward and performance for this measurement are $\langle \bar{R} \rangle = 0.73 \pm 0.05$ and $P = 0.85 \pm 0.06$, respectively.

The same weight matrix was applied to 100 systems with randomly shuffled physical neuron assignment and the reward distribution measured as before with learning switched off, yielding the distribution depicted in the middle panel of the same plot, with mean expected reward and performance of $\langle \bar{R} \rangle = 0.37 \pm 0.09$ and $P = 0.47 \pm 0.11$. Each of the 100 randomly shuffled systems was then subjected to 50,000 learning iterations, starting from the maladapted weight matrix, leading to the distribution shown in the bottom panel, with mean expected reward and performance of $\langle \bar{R} \rangle = 0.67 \pm 0.07$ and $P = 0.81 \pm 0.09$.

This demonstrates that our learning paradigm implicitly adapts synaptic weights to a specific pattern of neuronal variability and compensates for fixed-pattern noise. In a more general context, these findings support the idea that for neuromorphic systems endowed with synaptic plasticity, time-consuming and resource-intensive calibration of individual components can be, to a large extent, supplanted by on-chip learning.

3.3. Learning Robustness

One of the major concerns when using analog electronics is the control of fixed-pattern noise. We can partly compensate for fixed-pattern noise using a calibration routine (Aamir et al., 2018), but this procedure is subject to a trade-off between accuracy and the time and computational resources required for obtaining the calibration data. Highly precise calibration is costly because it requires an exhaustive mapping of a high-dimensional parameter space, which has to be done for each chip individually. A faster calibration routine, on the other hand, necessarily involves taking shortcuts, such as assuming independence between the influence of hardware parameters, thereby potentially leading to systematic deviations from the target behavior. Furthermore, remaining variations can affect the transfer of networks between chips and therefore potentially impact learning success when using a given set of (hyper-)parameters. We discuss these issues in the following. All results in this section were obtained after using 50,000 training iterations, which take around 25 s of wall-clock time on our BSS2 prototypes.

3.3.1. Impact of Time Constant Calibration

The neuronal calibration (see section 2.1.3) adjusts hardware parameters controlling the LIF time constants (τ_{mem}, τ_{ref}, and τ_{syn}) on a per-neuron basis to optimally match target time constants (**Table 1**), compensating neuronal variability. Depending on the target parameter value, it is possible to reduce

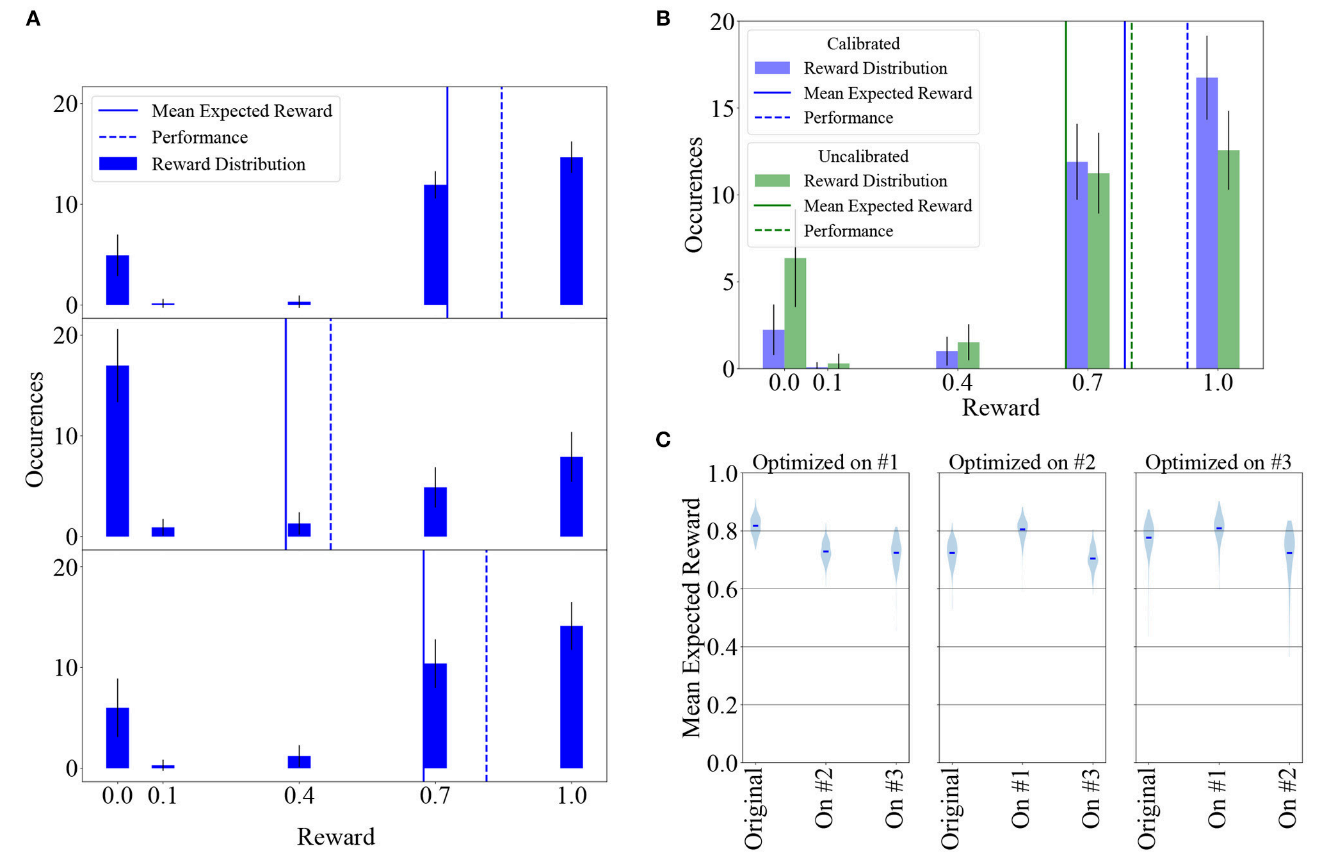


FIGURE 7 | Learning can largely supplant individual calibration of neuron parameters and adapts synaptic weights to compensate for neuronal variability. **(A)** Top: Reward distribution over 100 experiments measured with a previously learned, fixed weight matrix. Middle: Same as above, with randomly permuted neurons in each of the 100 experiments. The agent's performance and therefore its received reward decline due to the weight matrix being adapted to a specific pattern of fixed-pattern noise. Bottom: Allowing the agent to learn for 50, 000 additional iterations after having randomly shuffled its neurons leads to weight re-adaptation, increasing its performance and received reward. In these experiments, LIF parameters were not calibrated individually per neuron. **(B)** Reward distribution after 50, 000 learning iterations in 100 experiments for a calibrated and an uncalibrated system, with learning being largely able to compensate for the difference. **(C)** Results can be reproduced on different chips. Violin plot of mean expected reward after hyperparameter optimization on chips #1, #2, and #3. Results are shown for the calibrated case. All other results in this manuscript were obtained using Chip #1.

the standard deviations of the LIF time constants across a chip by up to an order of magnitude (Aamir et al., 2018).

We investigated the effect of uncalibrated neuronal time constants on learning. The uncalibrated state is defined by using the same value for a given hardware parameter for all neurons on a chip. To set up a reasonable working point, we chose this to be the average of the calibrated values of all neurons on the chip. A histogram of the membrane time constants of all neurons on the chip in the calibrated and uncalibrated state is given in **Figure 2B**. In both cases, voltages ($v_{\text{leak}}, v_{\text{thresh}}, v_{\text{reset}}$) were uncalibrated.

We measured the reward distribution after learning in both the calibrated and the uncalibrated state, performing 100 experiments in both cases (**Figure 7B**). Even in the uncalibrated state, learning was possible using only the inherent hardware noise for action exploration, albeit with some loss (around 17 %) in mean expected reward. The mean expected reward and performance in the calibrated state are $\langle\bar{R}\rangle = 0.79 \pm 0.05$ and $P = 0.93 \pm 0.05$, respectively. In the uncalibrated state, the values are $\langle\bar{R}\rangle = 0.65 \pm 0.08$ and $P = 0.80 \pm 0.09$.

As a corollary, the reward distributions depicted in **Figure 7B** suggest that narrowing the reward window (defined in Equation 5) to half the original size (i.e., removing the $R \leq 0.4$ part) would only have a small effect on converged Pong performance.

3.3.2. Transferability of Results Between Chips

All results presented thus far were obtained using one specific chip (henceforth called chip #1) and parameter set (given in **Table 1** as set #1). These parameters were found using a hyper-parameter optimization procedure (see section 2) that was performed on this chip. In addition, we performed the same optimization procedure on two other chips, using the original optimization results as a starting point, which yielded three parameter sets in total. The two additional parameter sets, sets #2 and #3 corresponding to chips #2 and #3, are also given in **Table 1**.

We then investigated the effects of transferring each parameter set to the other chips, testing all six possible transitions of learned parameters between the chips. To this end, we conducted 200 trials consisting of 50,000 learning iterations on every chip for every parameter set and compare the resulting mean expected reward in **Figure 7C**. Learning results were similar across all nine experiments. As expected due to process variations, there were small systematic deviations between the chips, with chip #1 slightly outperforming the other two for all three hyperparameter sets, but in all scenarios the agent successfully learned to play the game. These results suggest that chips can be used as drop-in replacements for other chips and that the hyperparameter optimization does not have to be specific to a particular substrate to yield good results.

3.4. Speed and Power Consumption

3.4.1. Speed

An essential feature of BSS2 is its 10^3 speed-up factor compared to biological real-time. To put this in perspective, we compared the emulation on our BSS2 prototypes to a software simulation with NEST running on a conventional CPU (see section 2.5). We found that a single experiment iteration in the simulation takes 50 *ms* when running it on a single core of an Intel i7-4771 CPU (utilizing more cores does not lead to a faster simulation due to the small size of the network). For our speed comparison, we differentiate the case of no injected noise (where the network's time evolution is fully deterministic) and the case where we use NEST's *noise_generator* module to inject current noise into each neuron. When noise is injected, the 200 *ms* of neural network activity is simulated in 4.3 *ms*; without noise, the activity is simulated in 1.2 *ms*. This is the time that is spent in NEST's state propagation routine (the *Simulate* routine). The remainder is spent calculating the plasticity rule and environment in the additional Python script, which we consider separately as it was external to NEST.

In contrast, BSS2 takes 0.4 *ms* per iteration as measured using the time stamp register of the PPU. This time is approximately equally divided between neural network emulation and other calculations, including the updates of the synaptic weights and the environment. The total time duration of an experiment with 50,000 learning iterations is 25 *s* for BSS2 and 40 *min* for the software simulation. A constant overhead of around 5 *s* when using the chip is spent on calibration, connection setup and configuring the chip.

The comparison between BSS2 and the software simulation is visualized in **Figure 8**. Note that while the calculation of eligibility traces in software was not optimized for speed, it incurs a significant computational cost, especially as it scales linearly with the number of synapses and therefore approximately quadratically with the network size. This is in contrast to the emulation on BSS2, where the eligibility trace is measured locally with analog circuitry at each synapse during network emulation.

In both cases, the time taken for environment simulation is negligible compared to plasticity calculation. We have confirmed that these measurements are independent on learning progress (i.e., measurements during the first iterations and with a diagonal weight matrix are equal).

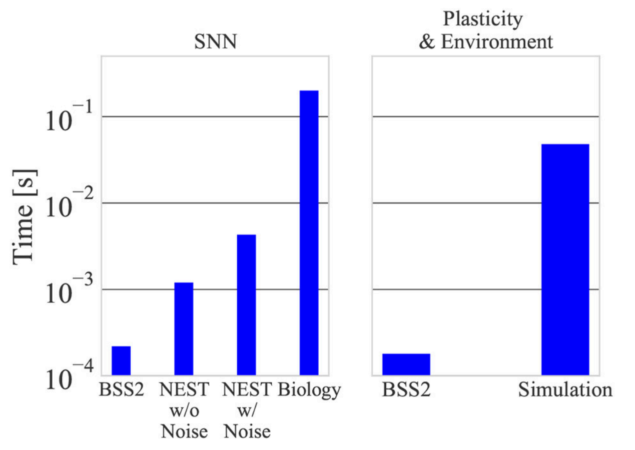


FIGURE 8 | A single experiment iteration on BSS2 takes around 400 µs, with 220 µs devoted to network emulation and 180 µs to plasticity calculation. The spiking network activity corresponds to 200 biological ms. In the software simulation, the neural network with and without noise is simulated in 4.3 and 1.2 ms, respectively, while plasticity calculations take around 50 ms.

3.4.2. Power Consumption

We measured the current drawn by the CPU during the software simulation of the neural network, i.e., during NEST's numerical simulation routine as well as when idling using the EPS 12 *V* power supply cable that supplies only the CPU. Again, we differentiate the cases of no noise and injected current noise. Based on these measurements, we determined a lower bound of 24 *W* for the CPU power consumption during SNN simulation without noise and a lower bound of 25 *W* when injecting noise. A single simulation iteration in software without noise, taking 1.2 *ms* and excluding plasticity and environment simulation, therefore consumes at least 29 *mJ*. When we inject current noise, the simulation takes 4.3 *ms* which corresponds to an energy consumption of 106 *mJ* per iteration.

By measuring the current drawn by BSS2 during the experiment, we calculated the power dissipated by it to be 57 *mW*, consistent with the measurement of another network in (Aamir et al., 2018) (this is not considering the power drawn by the FPGA, which is only used for initial configuration, and other prototype board components). This implies a per-iteration energy consumption, including plasticity and environment simulation, of around 23 µ*J* by the chip.

These measurements imply that, in a conservative comparison, the emulation using BSS2 is at least three orders of magnitude more energy-efficient than the software simulation.

4. DISCUSSION AND OUTLOOK

We demonstrated key advantages of our approach to neuromorphic computing in terms of speed and energy efficiency compared to a conventional approach using dedicated software. The already observable order-of-magnitude speed-up of our BSS2 prototypes compared to software for our small-scale test case is expected to increase significantly for larger networks.

At the same time, this performance can be achieved with a three-orders-of-magnitude advantage in terms of power consumption, as our conservative estimate shows.

Furthermore, we showed how substrate imperfections, which inevitably lead to parameter variations in analog neuro-synaptic circuits, can be compensated using an implementation of reinforcement learning via neuromodulated plasticity based on R-STDP. Meta-learning can be done efficiently despite substrate variability, as results of hyperparameter optimization can be transferred across chips. We further find that learning is not only robust against analog temporal variability, but that trial-to-trial variations are in fact used as a computational resource for action exploration.

In this context, it is important to mention that temporal variability is generally undesirable due to its uncontrollable nature and chips are designed with the intention of keeping it to a minimum. Still, learning paradigms that handle such variability gracefully are a natural fit for analogue neuromorphic hardware, where noise inevitably plays a role.

Due to the limited size of the chips used in this pilot study, the neural networks discussed here are constrained in size. However, the final system will represent a significant scale-up of this prototype. The next hardware revision will be the first full-size BSS2 chip and will provide 512 neurons, 130k synapses and two embedded processors with appropriately scaled vector units. Neurons on the chip will emulate the Adaptive-Exponential (AdEx) Integrate-and-Fire model with multiple compartments, while synapses will contain additional features for Short-Term Plasticity (STP). Poisson-like input stimuli with rates in the order of *MHz*, i.e., biological *kHz*, will be realized using pseudo-random number generators to provide the possibility for stochastic spike input to neurons, yielding controllable noise which can be used for emulation of *in-vivo* activity and large-scale functional neural networks (Destexhe et al., 2003; Petrovici et al., 2014, 2016). The full BSS2 neuromorphic system will comprise hundreds of such chips on a single wafer which itself will be interconnected to other wafers, similar to its predecessor BrainScaleS 1 (Schemmel et al., 2010). The study at hand lays the groundwork for future experiments with reinforcement learning in neuromorphic SNNs, where an expanded hardware real-estate will allow the emulation of more complex agents learning to navigate more difficult environments.

The advantages in speed and energy consumption will become even more significant when moving to large networks. In our experiments, the relative speed of the software simulation was due to the small size of the network. State-of-the-art software simulations of large LIF neural networks take minutes to simulate a biological second (Jordan et al., 2018) and even digital simulations on specialized neuromorphic hardware typically achieve real-time operation (Mikaitis et al., 2018; van Albada et al., 2018). On BSS2, the speed-up factor of 10^3 is independent of network size, which, combined with the two embedded processors per chip and the dedicated synapse circuits containing local correlation sensors, enables the accelerated emulation of large-scale plastic neural networks.

The speed-up factor of BSS2 is both an opportunity and a challenge. In general, rate-based or sampling-based codes would profit from longer integration times enabled by the acceleration. On the other hand, interfacing BSS2 to the real world (e.g., using robotics) requires fast sensors; the speed-up could enable fast sensor-motor loops with possible applications in, for example, radar beam shaping. In general, however, the main advantage of an accelerated system becomes most evident when learning is involved: long-term learning processes lasting several years in biological spiking networks can be emulated in mere hours on BSS2, whereas real-time simulations are unfeasible.

The implemented learning paradigm (R-STDP) and simulated environment (Pong) were kept simple in order to focus our study on the properties of the prototype hardware. R-STDP is a well-studied model that lends itself well to hardware implementation, while the simplified Pong game is a suitable learning task that can be realized on the prototype chip and provides an accessible, intuitive interpretation. Still, the PPU's computational power limits the complexity of environment simulations that can be realized on-chip, especially when simulations have hard real-time constraints. However, such simulations could take place on a separate system that merely provides spike input, collects spike output and provides reward to the neuromorphic system. Alternatively, simulations could be horizontally distributed across PPUs.

Solving more complex learning tasks demands more complex network models. We expect that the future large-scale BSS2 system will be able to instantiate not only larger, but also more complex network models, by offering more flexibility in the choice of spiking neuron dynamics (e.g., AdEx, LIF), short-term synaptic plasticity and enhanced PPU capabilities. However, further theoretical work is required for mapping certain state-of-the-art reinforcement learning models, such as DQN (Mnih et al., 2015) and AlphaGo (Zero) (Silver et al., 2016, 2017) to this substrate. On the other hand, learning paradigms like TD-STDP (Frémaux et al., 2013), which implements actor-critic reinforcement learning using a SNN, already match the capabilities of a large-scale version of the BSS2 system and therefore represent suitable candidates for learning in more complex environments with sparse rewards and agent-dependent state transitions.

DATA AVAILABILITY

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

AUTHOR CONTRIBUTIONS

AnH, AK, TW, and MP designed the study. TW conducted the experiments and the evaluations. TW and AK wrote the initial manuscript. EM and CM supported experiment realization. EM coordinated the software development for the neuromorphic systems. YS contributed with characterization, calibration testing and debugging of the prototype system. SA designed the neuron circuits. AG was responsible for chip assembly and did the digital front- and backend implementation. ArH extended the GNU Compiler Collection backend support for the embedded processor. DS contributed to the host control software development and supported chip commissioning. KS designed and assembled the development board. CP provided FPGA firmware and supported chip commissioning. GK contributed to the verification of the synaptic input circuits. JS is the architect and lead designer of the neuromorphic platform. MP, KM, JS, SB, and EM provided conceptual and scientific advice. All authors contributed to the final manuscript.

FUNDING

This research was supported by the EU 7th Framework Program under grant agreements 269921 (BrainScaleS), 243914 (Brain-i-Nets), 604102 (Human Brain Project), the Horizon 2020 Framework Program under grant agreement 720270 (Human Brain Project), the Manfred Stärk Foundation and the Deutsche Forschungsgemeinschaft within the funding programme Open Access Publishing, by the Baden-Württemberg Ministry of Science, Research and the Arts and by Ruprecht-Karls-Universität Heidelberg.

ACKNOWLEDGMENTS

The authors wish to thank Simon Friedmann, Matthias Hock, and Paul Müller. They contributed to developing the hardware and software that enabled this experiment.

REFERENCES

Electronic Vision(s) (2017). Available online at: https://github.com/electronicvisions/gcc

Aamir, S. A., Müller, P., Hartel, A., Schemmel, J., and Meier, K. (2016). "A highly tunable 65-nm CMOS LIF neuron for a large scale neuromorphic system," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference* (Lausanne: IEEE), 71–74.

Aamir, S. A., Stradmann, Y., Müller, P., Pehle, C., Hartel, A., Grübl, A., Schemmel, J., and Meier, K. (2018). "An accelerated LIF neuronal network array for a large scale mixed-signal neuromorphic architecture," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, 1–14.

Bayer, H. M., and Glimcher, P. W. (2005). Midbrain dopamine neurons encode a quantitative reward prediction error signal. *Neuron* 47, 129–141. doi: 10.1016/j.neuron.2005.05.020

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J.-M., et al. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 699–716. doi: 10.1109/JPROC.2014.2313565

Brzosko, Z., Schultz, W., and Paulsen, O. (2015). Retroactive modulation of spike timing-dependent plasticity by dopamine. *eLife* 4:e09685. doi: 10.7554/eLife.09685

Davies, M., Srinivasa, N., Lin, T., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Destexhe, A., Rudolph, M., and Paré, D. (2003). The high-conductance state of neocortical neurons *in vivo*. *Nat. Rev. Neurosci.* 4, 739–751. doi: 10.1038/nrn1198

Edelmann, E. and Lessmann, V. (2011). Dopamine modulates spike timing-dependent plasticity and action potential properties in CA1 pyramidal neurons of acute rat hippocampal slices. *Front. Synap. Neurosci.* 3:6. doi: 10.3389/fnsyn.2011.00006

Farries, M. A., and Fairhall, A. L. (2007). Reinforcement learning with modulated spike timing-dependent synaptic plasticity. *J. Neurophysiol.* 98, 3648–3665. doi: 10.1152/jn.00364.2007

Fetz, E. E., and Baker, M. A. (1973). Operantly conditioned patterns on precentral unit activity and correlated responses in adjacent cells and contralateral muscles. *J. Neurophysiol.* 36, 179–204. doi: 10.1152/jn.1973.36.2.179

Frémaux, N., and Gerstner, W. (2015). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circ.* 9:85. doi: 10.3389/fncir.2015.00085

Frémaux, N., Sprekeler, H., and Gerstner, W. (2010). Functional requirements for reward-modulated spike-timing-dependent plasticity. *J. Neurosci.* 30, 13326–13337. doi: 10.1523/JNEUROSCI.6249-09.2010

Frémaux, N., Sprekeler, H., and Gerstner, W. (2013). Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS Comput. Biol.* 9:e1003024. doi: 10.1371/journal.pcbi.1003024

Friedmann, S., Frémaux, N., Schemmel, J., Gerstner, W., and Meier, K. (2013). Reward-based learning under hardware constraints - Using a RISC processor embedded in a neuromorphic substrate. *Front. Neurosci.* 7:160. doi: 10.3389/fnins.2013.00160

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., and Meier, K. (2017). Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Trans. Biomed. Circ. Syst.* 11, 128–142. doi: 10.1109/TBCAS.2016.2579164

Furber, S. (2016). Large-scale neuromorphic computing systems. *J. Neural Eng.* 13:051001. doi: 10.1088/1741-2560/13/5/051001

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. (2014). The SpiNNaker project. *Proc. IEEE* 102, 652–665. doi: 10.1109/JPROC.2014.2304638

Guttman, N. (1953). Operant conditioning, extinction, and periodic reinforcement in relation to concentration of sucrose used as reinforcing agent. *J. Exp. Psychol.* 46, 213–224. doi: 10.1037/h0061893

Head, T., MechCoder, Louppe, G., Shcherbatyi, I., fcharras, Vinícius, Z., et al. (2018). *scikit-optimize/scikit-optimize: v0.5.2*. Zenodo.

Hock, M., Hartel, A., Schemmel, J., and Meier, K. (2013). "An analog dynamic memory array for neuromorphic hardware," in *2013 European Conference on Circuit Theory and Design (ECCTD)* (Dresden: IEEE), 1–4.

Hoerzer, G. M., Legenstein, R., and Maass, W. (2014). Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cereb. Cortex* 24, 677–690. doi: 10.1093/cercor/bhs348

Hollerman, J. R., and Schultz, W. (1998). Dopamine neurons report an error in the temporal prediction of reward during learning. *Nat. Neurosci.* 1, 304–309. doi: 10.1038/1124

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Schaik, A. V., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex* 17, 2443–2452. doi: 10.1093/cercor/bhl152

Jordan, J., Ippen, T., Helias, M., Kitayama, I., Sato, M., Igarashi, J., et al. (2018). Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Front. Neuroinform.* 12:2. doi: 10.3389/fninf.2018.00002

Legenstein, R., Pecevski, D., and Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput. Biol.* 4:e1000180. doi: 10.1371/journal.pcbi.1000180

Maass, W. (2014). Noise as a resource for computation and learning in networks of spiking neurons. *Proc. IEEE* 102, 860–880. doi: 10.1109/JPROC.2014.2310593

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345:668. doi: 10.1126/science.1254642

Mikaitis, M., Pineda García, G., Knight, J. C., and Furber, S. B. (2018). Neuromodulated synaptic plasticity on the spinnaker neuromorphic system. *Front. Neurosci.* 12:105. doi: 10.3389/fnins.2018.00105

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Moritz, C. T., and Fetz, E. E. (2011). Volitional control of single cortical neurons in a brain-machine interface. *J Neural Eng.* 8:025017. doi: 10.1088/1741-2560/8/2/025017

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., and Masquelier, T. (2018a). Combining STDP and reward-modulated STDP in deep convolutional spiking neural networks for digit recognition. arXiv:1804.00227.

Mozafari, M., Kheradpisheh, S. R., Masquelier, T., Nowzari-Dalini, A., and Ganjtabesh, M. (2018b). "First-spike-based visual categorization using reward-modulated STDP," in *IEEE Transactions on Neural Networks and Learning Systems*, 1–13.

Niv, Y. (2009). Reinforcement learning in the brain. *J. Math. Psychol.* 53, 139–154. doi: 10.1016/j.jmp.2008.12.005

Pawlak, V., and Kerr, J. N. D. (2008). Dopamine receptor activation is required for corticostriatal spike-timing-dependent plasticity. *J. Neurosci.* 28, 2435–2446. doi: 10.1523/JNEUROSCI.4402-07.2008

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2016). Stochastic inference with spiking neurons in the high-conductance state. *Phys. Rev. E* 94:042312. doi: 10.1103/PhysRevE.94.042312

Petrovici, M. A., Vogginger, B., Müller, P., Breitwieser, O., Lundqvist, M., Muller, L., et al. (2014). Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLoS ONE* 9:e108590. doi: 10.1371/journal.pone.0108590

Peyser, A., Sinha, A., Vennemo, S. B., Ippen, T., Jordan, J., Graber, S., et al. (2017). *NEST 2.14.0*. Zenodo.

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., et al. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Front. Neurosci.* 9:141. doi: 10.3389/fnins.2015.00141

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., and Millner, S. (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS"10)*, 1947–1950.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science* 275, 1593–1599. doi: 10.1126/science.275.5306.1593

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484-489. doi: 10.1038/nature16961

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al (2017). Mastering the game of Go without human knowledge. *Nature* 550:354. doi: 10.1038/nature24270

Stallman, R. M., and GCC Developer Community (2018). *GCC 8.0 GNU Compiler Collection Internals*. 12th Media Services.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

van Albada, S. J., Rowley, A. G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A. B., et al. (2018). Performance Comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software nest for a full-scale cortical microcircuit model. *Front. Neurosci.* 12:291. doi: 10.3389/fnins.2018.00291

Vasilaki, E., Frémaux, N., Urbanczik, R., Senn, W., and Gerstner, W. (2009). Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail. *PLoS Comput. Biol.* 5:e1000586. doi: 10.1371/journal.pcbi.1000586

Wunderlich, T. (2019). *Neuromorphic R-STDP Experiment Simulation*. Available online at: https://github.com/electronicvisions/model-sw-pong

Xie, X., and Seung, H. S. (2004). Learning in neural networks by reinforcement of irregular spiking. *Phys. Rev. E* 69:041909. doi: 10.1103/PhysRevE.69.041909

Zoschke, K., Güttler, M., Böttcher, L., Grübl, A., Husmann, D., Schemmel, J., et al. (2017). Full wafer redistribution and wafer embedding as key Technologies for a multi-scale neuromorphic hardware cluster. *arXiv:1801.04734*. doi: 10.1109/EPTC.2017.8277579

Conflict of Interest Statement: 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.

frontiers in Neuroscience

ORIGINAL RESEARCH
published: 28 May 2019
doi: 10.3389/fnins.2019.00504

Analysis of Liquid Ensembles for Enhancing the Performance and Accuracy of Liquid State Machines

Parami Wijesinghe, Gopalakrishnan Srinivasan, Priyadarshini Panda and Kaushik Roy*

School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, United States

Liquid state machine (LSM), a bio-inspired computing model consisting of the input sparsely connected to a randomly interlinked reservoir (or liquid) of spiking neurons followed by a readout layer, finds utility in a range of applications varying from robot control and sequence generation to action, speech, and image recognition. LSMs stand out among other Recurrent Neural Network (RNN) architectures due to their simplistic structure and lower training complexity. Plethora of recent efforts have been focused toward mimicking certain characteristics of biological systems to enhance the performance of modern artificial neural networks. It has been shown that biological neurons are more likely to be connected to other neurons in the close proximity, and tend to be disconnected as the neurons are spatially far apart. Inspired by this, we propose a group of locally connected neuron reservoirs, or an ensemble of liquids approach, for LSMs. We analyze how the segmentation of a single large liquid to create an ensemble of multiple smaller liquids affects the latency and accuracy of an LSM. In our analysis, we quantify the ability of the proposed ensemble approach to provide an improved representation of the input using the Separation Property (SP) and Approximation Property (AP). Our results illustrate that the ensemble approach enhances class discrimination (quantified as the ratio between the SP and AP), leading to better accuracy in speech and image recognition tasks, when compared to a single large liquid. Furthermore, we obtain performance benefits in terms of improved inference time and reduced memory requirements, due to lowered number of connections and the freedom to parallelize the liquid evaluation process.

Keywords: liquid state machines, ensembles, spiking neural networks, separation property, approximation property, discriminant ratio

OPEN ACCESS

Edited by:
*Arindam Basu,
Nanyang Technological University,
Singapore*

Reviewed by:
*Peng Li,
Texas A&M University, United States*
*Subhrajit Roy,
IBM Research, Australia*

***Correspondence:**
*Parami Wijesinghe
pwijesin@purdue.edu*

Specialty section:
*This article was submitted to
Neuromorphic Engineering,
a section of the journal
Frontiers in Neuroscience*

Received: 15 February 2019
Accepted: 01 May 2019
Published: 28 May 2019

Citation:
Wijesinghe P, Srinivasan G, Panda P and Roy K (2019) Analysis of Liquid Ensembles for Enhancing the Performance and Accuracy of Liquid State Machines. Front. Neurosci. 13:504. doi: 10.3389/fnins.2019.00504

1. INTRODUCTION

Over the past few decades, artificial neural algorithms have developed to an extent that they can perform more human-like functions. Recurrent Neural Networks (RNNs) and their variants such as Long Short Term Memory (LSTM) networks have become the state-of-the-art for processing spatio-temporal data. The massive RNNs of today, can describe images in natural language (Xie, 2017), produce handwriting (Graves, 2013), and even make phone calls to book appointments (Yaniv and Yossi, 2018). Such fascinating, human-like capabilities are obtained at the cost of increased structural and training complexity, and thus significant power consumption, storage requirements, and delay.

In this work we focus on a particular type of spiking RNN; the Liquid State Machine (LSM) (Maass et al., 2002). An LSM consists of a set of inputs sparsely connected to a randomly and recurrently interlinked pool of spiking neurons called the "liquid". The liquid is connected to an output classifier, which can be trained using standard methods such as Spike Timing Dependent Plasticity (STDP), backpropagation, delta rule *etc*. (Kötter, 2012) and using enhanced learning rules (Roy et al., 2013). LSMs have been used for a variety of applications including robot control (Urbain et al., 2017), sequence generation (Panda and Roy, 2017), decoding actual brain activity (Nikolić et al., 2009), action recognition (Panda and Srinivasa, 2018), speech recognition (Maass et al., 2002; Verstraeten et al., 2005; Goodman and Ventura, 2006; Zhang et al., 2015; Wu et al., 2018; Zhang and Li, 2019), and image recognition (Grzyb et al., 2009; Wang and Li, 2016; Srinivasan et al., 2018; Zhang and Li, 2019).

LSMs gained their popularity due to two main reasons. First, the LSM architecture is neuro-inspired. By conserving energy via spike-based operation, the brain has evolved to achieve its prodigious signal-processing capabilities using orders of magnitude less energy than the state-of-the-art supercomputers (Anastassiou et al., 2011; Cruz-Albrecht et al., 2012). Therefore, with the intention to pave pathways to low power neuromorphic computing, much consideration is given to realistic artificial brain modeling (Waldrop, 2012; Neftci et al., 2016; Wijesinghe et al., 2018). Furthermore, the gene regulation network (GRN) of the Bacterium "Escherichia Coli" (E-Coli) was experimentally assessed and shown to behave similar to an LSM (Jones et al., 2007). The E. Coli has the capacity for perceptual categorization, especially for discrimination between complex temporal patterns of chemical inputs. Second, LSMs have simple structure and lower training complexity among other RNNs. The claim is that, sufficiently large and complex liquids inherently possess large computational power for real-time computing. Therefore, it is not necessary to "construct" circuits to achieve substantial computational power. However, such simple structure of LSMs comes with an accuracy trade-off. A plethora of work in the literature suggests mechanisms for improving the accuracy of LSMs including training the liquid connections (Wang and Li, 2016) and involving multiple layers of liquids to form deep LSMs (Xue et al., 2016). Despite the accuracy improvement, these mechanisms found in literature tend to alter the standard simple structure of LSMs. Choosing an LSM for a particular application and improving its accuracy at the cost of added complexity, nonetheless questions the motivation behind choosing an LSM in the first place.

Without deviating from the inherent simplicity of the LSM structure, several basic approaches can be used to improve its accuracy. One such fundamental approach is to increase the number of neurons within the liquid. However, the number of connections within the liquid also increases following a quadratic relationship with the number of neurons. Furthermore, the sensitivity of accuracy to the liquid neuron count decreases with the number of neurons beyond a certain point. In other words, enlarging the liquid introduces scalability challenges, and the accompanied cost tends to veil the accuracy benefits. The percentage connectivity also plays a role in improving the accuracy. Either high or low percentage connectivity results in accuracy degradation, signaling the existence of an optimum connectivity.

Note, there are two key properties that measure the capacity of an LSM: *separation* and *approximation* (Maass et al., 2002). Aforementioned basic approaches; changing the number of neurons and connectivity in the liquid, indeed has an impact on the above measures. Based on separation and approximation, we propose an "ensemble of liquids approach" that can improve the classification accuracy (compared to a single large LSM) with reduced connectivity. The approach is scalable and preserves the simplicity of the network structure. In our ensemble of liquids approach, we split a large liquid into multiple smaller liquids. These resultant liquids can be evaluated in parallel since they are independent of each other, which leads to performance benefits. Furthermore, for a given percentage connectivity, the number of connections available in the ensemble approach is less than that of a single liquid with the same number of neurons. This reduces the storage requirement of the LSM as well. We used a variant of the Fisher's linear discriminant ratio (Fisher, 1936; Fukunaga and Mantock, 1983) (the ratio between the separation and approximation) to quantify how well the ensemble of liquids represents the spatio-temporal input data. We observed that increasing the liquid count beyond a certain point reduces the accuracy of the LSM. This signals the existence of an optimum number of liquids, which is highly dependent upon the application and the number of neurons in the liquid. We show that dividing the liquid provides both accuracy and performance benefits for spatial and temporal pattern recognition tasks, on standard speech and image data sets.

The "ensemble" concept has been previously used (Yao et al., 2013) for echo state networks or ESNs (Jaeger, 2007), which are similar in architecture to LSMs but use artificial rate-based neurons. Rather than using a single ESN predictor, multiple predictors (component predictors) were used and their predictions were combined together to obtain the final outcome. This approach was proposed to avoid the instability of the output of each individual predictor, since the input and internal connection weights are assigned randomly. The final ensemble outcome was obtained by averaging the predictions of the component predictors. The approach in Yao et al. (2013) is different from our work since we design the ensemble of liquids by removing certain connections from a bigger reservoir. Furthermore, only a single classifier is used at the output in our work in contrast to Yao et al. (2013). The authors in Maass et al. (2002) conducted a small experiment with two time-varying signals, which shows that using four liquids is better than using a single liquid in terms of enhancing the separation property. However, in their experiments, the four liquids in total have four times the number of neurons as the single liquid case. Therefore, it is not obvious whether the improvement in separation is solely due to having four "separate" liquids. The increased number of neurons itself might have played a role in enhancing the separation. In contrast, we analyze the effects of dividing a large liquid into multiple smaller units, while leaving the total number of neurons the same. Research (Srinivasan et al., 2018) also shows that multiple liquids perform better than a single liquid, at higher number of neurons. The input to liquid connections in Srinivasan et al. (2018) were trained in an unsupervised manner. Also note that each liquid was fed with distinct parts of an input, and hence is different from this work.

2. MATERIALS AND METHODS

2.1. Liquid State Machine (baseline)

In this section, we explain the structure and training of the LSM used in this work. The conventional structure of an LSM consists of an input layer sparsely connected to a randomly interlinked pool of spiking neurons called the liquid. The liquid is then connected to a classifier which has synaptic weights that could be learnt using supervised training algorithms for inference.

2.1.1. Liquid Neurons

The neurons within the liquid are leaky integrate-and-fire neurons (Abbott, 1999) of two types; excitatory and inhibitory neurons. The number of excitatory (E) neurons and inhibitory (I) neurons were selected according to a $4:1$ ratio, as observed in the auditory cortex (Wehr and Zador, 2003). The membrane potential (V) of a neuron increases/decreases as a pre excitatory/inhibitory neuron connected to it spikes that is described by

$$\tau \frac{dV}{dt} = (E_{rest} - V) + g_e(E_{exc} - V) + g_i(E_{inh} - V) \qquad (1)$$

where E_{rest} is the resting membrane potential, τ is the membrane potential decay time constant, E_{exc} and E_{inh} are the equilibrium potentials of excitatory and inhibitory synapses, and g_e and g_i are the conductance values of the excitatory and inhibitory synapses, respectively. As the membrane potential reaches a certain threshold, the neuron generates a spike. The membrane potential drops to its reset potential upon generating a spike as shown in **Figure 1A**, and then enters its refractory period t_{refrac} during which it produces no further spikes. The formulations elaborated in Diehl and Cook (2015) were used for modeling the dynamics of the spiking neurons.

2.1.2. Liquid Connections

The input is sparsely connected to the liquid ($In{\rightarrow}E$ connections). The percentage input to liquid connectivity ($P_{IN\rightarrow E}$) plays an important role in achieving good accuracy as will be explained in section 3.4.2. The liquid is composed of connections from excitatory to excitatory neurons ($E{\rightarrow}E$), excitatory to inhibitory neurons ($E{\rightarrow}I$), inhibitory to excitatory neurons ($I{\rightarrow}E$), and inhibitory to inhibitory neurons ($I{\rightarrow}I$). In our notation, the first letter indicates the pre-neuron type (PRE) and the second letter denotes the post-neuron type ($POST$). The selected percentage connectivity ($P_{IN\rightarrow E}$, $P_{E\rightarrow E}$, $P_{E\rightarrow I}$, $P_{I\rightarrow E}$, $P_{I\rightarrow I}$) within the liquid are shown in **Table 1**. These percentage connectivity values worked the best in terms of accuracy, for the neuron parameter selections in this work shown in **Table 2**. The strengths of all the connections ($W \in [0,1]^{N_{PRE}\times N_{POST}}$) were selected randomly (Maass et al., 2002) from a uniform distribution $U(0,1)$ (Toledo-Suárez et al., 2014; Srinivasan et al., 2018). A randomly generated mask $\left(M \in \{0,1\}^{N_{PRE}\times N_{POST}}, m_{ij} \in M\right)$ decides which connections exist to obtain the desired sparsity/percentage connectivity $\left(P_{PRE\rightarrow POST} = \frac{\sum_{(\forall i,\forall j)} m_{ij}}{(N_{PRE}\times N_{POST})} \times 100\%\right)$. Here N_{PRE} and N_{POST} are the number of PRE and $POST$ neurons, respectively. The dynamic conductance change model was used for synapses. i.e., when a pre-synaptic neuron fires, the synaptic conductance instantaneously changes according to their strengths and then decays exponentially with a time constant (Diehl and Cook, 2015). Following equation shows the dynamics of a synapse (g_e) with an excitatory pre-neuron. τ_{g_e} is the decay time constant. This is similar to the post-synaptic current model in Maass et al. (2003).

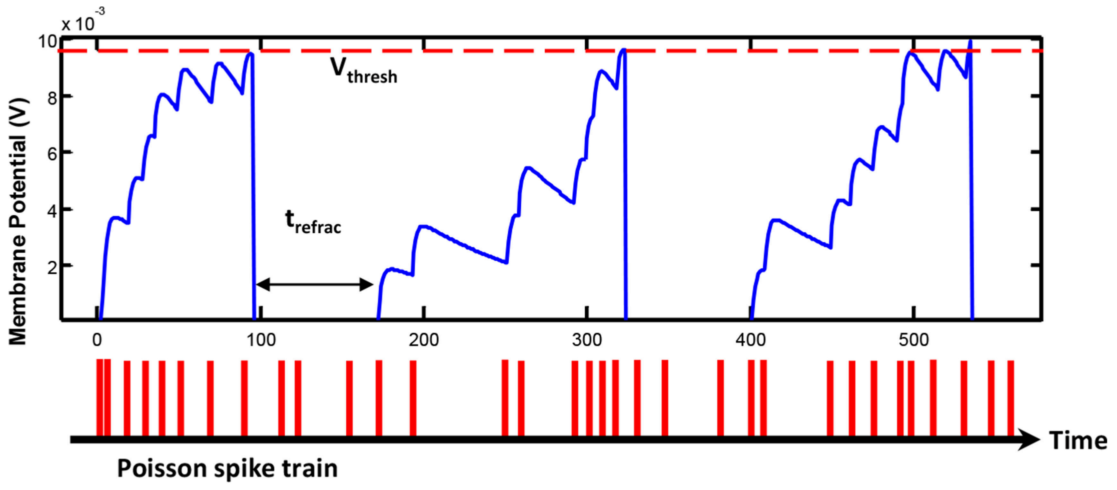
$$\tau_{g_e} \frac{dg_e}{dt} = -g_e \qquad (2)$$

2.1.3. Output Classifier

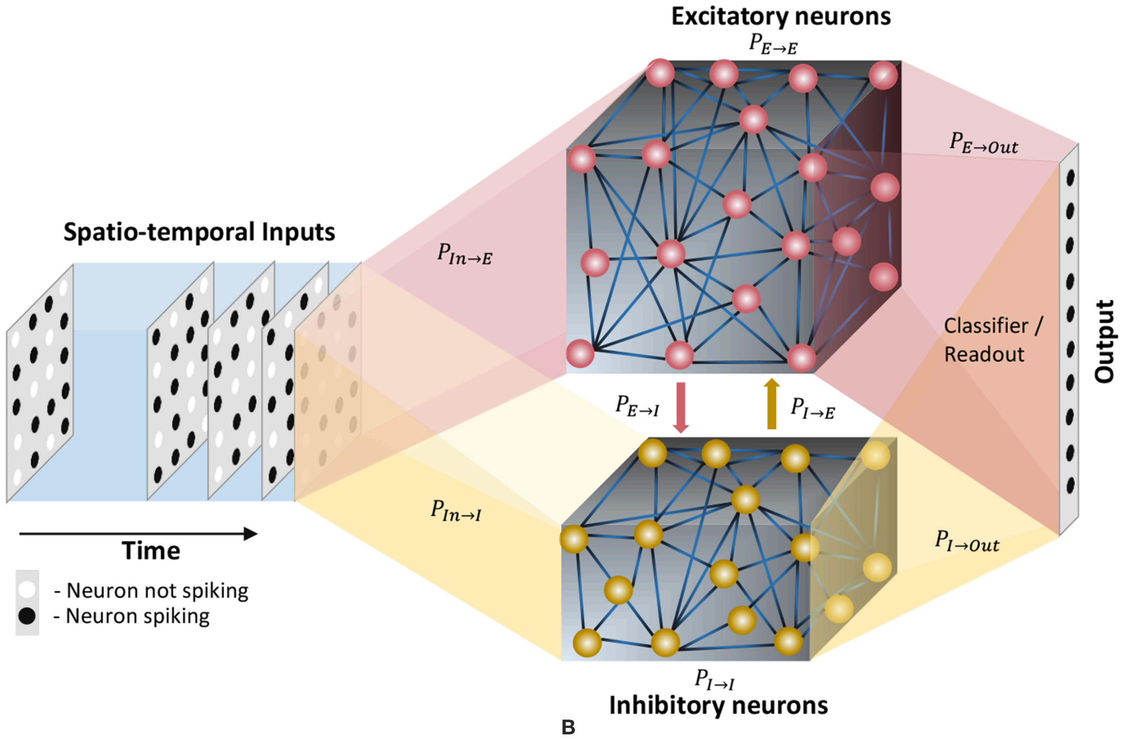
The liquid is connected to a classifier which is trained in a supervised manner. As suggested in Maass et al. (2002), a memory-less readout (the readout is not required to retain any memory of previous states) can be used to classify the states of the liquid. The liquid state in this work is the normalized spike count of the excitatory neurons (Kaiser et al., 2017) within a duration of T, when the input is applied. There is a liquid state vector ($s_i \in [0,1]^{N_E}$, N_E is the number of excitatory neurons) per applied input (i). The collection of all the state vectors were then used to train the classifier using gradient descent error backpropagation algorithm (Rumelhart et al., 1986), similar to Srinivasan et al. (2018). By doing this, we do discard some temporal information. However, since we do not use "anytime-speech-recognition" (a liquid with a classifier which is capable of recognizing a speech input, before the entire temporal signal is applied to the liquid) proposed in Maass et al. (2004), the above classification method is sufficient to achieve reasonable accuracy (as per the accuracy values reported in other LSM works) for the applications we are considering in this work.

2.2. Ensemble Approach for LSMs

In this section, we explain our proposed ensemble of liquids approach, which improves the scalability of LSMs. The proposed approach is different from the ensemble works available in literature on a variety of network types (feed-forward fully connected spiking and analog neural networks), where multiple classified outputs of independently trained networks are combined together to increase the probability of correct classification (Jacobs et al., 1991; Shim et al., 2016). In this work, we analyze the impact of dividing a reservoir, such that all the resultant small reservoirs can potentially be evaluated in parallel, for an applied input. As explained in the previous section, the typical structure of an LSM has an input, a liquid where neurons are sparsely interlinked, and a readout trained using supervised learning methods (**Figure 1B**). In our ensemble approach, the same number of liquid neurons (N_{tot}) is divided to create an N_{ens} number of smaller liquids, as shown in **Figure 2**. While dividing the liquid, the number of excitatory neurons (N_E^i) to inhibitory neurons (N_I^i) ratio in the ith ($i = 0, 1, ..., N_{ens}$) liquid is



A



B

FIGURE 1 | (A) The dynamics of the membrane potential (*V*) of a spiking neuron. Each spike shown below the graph will increase the membrane potential. When *V* reaches the threshold V_{thresh}, the neuron will generate a spike and *V* will drop to the rest potential V_{rest} for a t_{refrac} duration of time. This duration is called the refractory period, and the neuron stays idle within this period. Reprinted with the permission from Liyanagedera et al. (2017). **(B)** The structure of the liquid state machine. The input is connected to a reservoir with two types of neurons; inhibitory and excitatory. The reservoir is then connected to a classifier which is typically trained using supervised learning methods. The percentage connectivity between different types of *pre* and *post* neurons ($P_{pre\rightarrow post}$) are as indicated in the figure.

maintained at 4 : 1. The percentage connectivity is also adjusted to suit the new reduced number of neurons ($\frac{N_{tot}}{N_{ens}}$) in a liquid. This is done by first creating a standard LSM explained in the previous section with $\frac{N_{tot}}{N_{ens}}$ number of neurons and adjusting all the percentage connectivity values till we get a reasonable accuracy. Then the input to liquid percentage connectivity ($P_{IN\rightarrow E}$) was exhaustively changed until the accuracy peaks for a given application, which is then used for all the experiments reported in this work.

Each small liquid has its own liquid state vector, which is the normalized spike count of all the excitatory neurons in the respective liquid within a duration of T, as explained in the previous section. All the state vectors produced by each individual liquid in the ensemble are concatenated to form

TABLE 1 | Percentage connectivity within the liquid.

Type of connectivity	Percentage connectivity (Speech recognition)		Percentage connectivity (Image recognition)	
	TI-alpha(%)	$TI-10$(%)	MNIST(%)	E-MNIST(%)
Input–Excitatory	34	23	10	10
Input–Inhibitory	0	0	0	0
Excitatory–Excitatory	40	40	40	40
Excitatory–Inhibitory	40	40	40	40
Inhibitory–Excitatory	50	50	50	50
Inhibitory–Inhibitory	0	0	0	0

TABLE 2 | Spiking neuron parameters of the liquid state machine.

Parameter name	Parameter value
Excitatory weight decay time constant, t_{ge}	1 ms
Inhibitory weight decay time constant, t_{gi}	2 ms
Threshold inhibitory, $thresh_i$	−40 mV
Threshold excitatory, $thresh_e$	−52 mV
Inhibitory rest potential, $v_{rest,i}$	−60 mV
Excitatory rest potential, $v_{rest,e}$	−65 mV

one large state vector per input. Note that the length of the concatenated state vectors are the same for both the single liquid case (baseline) and for the ensemble of liquids, since the total number of neurons are held constant for a fair comparison. The concatenated state vector is used to train a single readout using gradient descent backpropagation. This division of one large liquid to form an ensemble of liquids enhances class discrimination associated with LSMs as elaborated in the next section.

2.3. Properties of LSMs

Two macroscopic properties of an LSM, namely, Separation Property (SP) and Approximation Property (AP), can be used to quantify a liquid's ability to provide an improved projection of the input data. With respect to classification applications, SP gives a measure of the liquid's ability to separate input instances that belong to different classes. AP, on the other hand, gives a measure of the closeness of the liquid's representation of two inputs that belong to the same class.

Several methods of quantifying the SP and AP as a measure of the computational power (kernel quality) of an LSM are suggested in Maass et al. (2002, 2005). Two methods of measuring the SP are *pairwise separation property* and *linear separation property*. The pairwise separation property is the distance between two continuous time states of the liquid ($x_u(t)$ and $x_v(t)$), resulting from two different input spike trains ($u(t)$ and $v(t)$). Here the continuous time states $x(t)$ are defined as the vector of output values of linear filters with exponential decay (with time constant 30ms Maass et al., 2002) at time t. The distance can be calculated by the Euclidean norm between $x_u(t_n)$ and $x_v(t_n)$ at sample point t_n. The final pairwise separation property can be evaluated by obtaining the average across all the sampled instances (at t_n), as explained in the following equation

$$SP_{pw} = \frac{1}{N_{samples}} \sum_{n=1(0<t_n<T)}^{N_{samples}} ||x_u(t_n) - x_v(t_n)|| \tag{3}$$

where $N_{samples}$ is the number of sample points. The pairwise separation property (SP_{pw}) can be used as a measure of the kernel quality for two given inputs. However, most real life applications deal with more than two input spike trains. To address this, linear separation property is proposed as a more suitable quantitative measure to evaluate the computational power of a liquid in an LSM. The linear separation property (SP_{lin}) is the rank of the $N \times m$ matrix M_s, which contains the continuous time states ($x_{u_1}(t_0), ..., x_{u_m}(t_0)$) of the liquid as its columns. The continuous time state $x_{u_i}(t_0)$ is the liquid response to the input u_i (these inputs are from the training set), sampled at $t = t_0$. If the rank of the matrix is m, it guarantees that any given assignment of target outputs $y_i \in \mathbb{R}$ at time t_0 can be attained by means of a linear readout (Maass et al., 2005). The rank of M_s is the degree of freedom the linear readout has, when mapping x_{u_i} to y_i. Even though the rank is $< m$, it can still be used as a measure of kernel quality of the LSM (Maass et al., 2005).

$$M_s = \left[x_{u_1}(t_0), ..., x_{u_i}(t_0), ..., x_{u_m}(t_0)\right] \tag{4}$$

$$SP_{lin} = rank\left(M_s\right)$$

The AP of the LSM can also be measured by the aforementioned rank concept as shown in Maass et al. (2005); Roy and Basu (2016). Instead of using significantly different examples in the training set, now the continuous time states $x_{u_i^j}(t_0)$ of the liquid are measured by feeding jittered versions of u_i (u_i^j) to the liquid. The rank of the matrix M_a that has m such continuous time states $x_{u_1^j}(t_0), ..., x_{u_m^j}(t_0)$ sampled at t_0 as its columns, is evaluated as a measure of the generalization capability of the liquid for unseen inputs. Unlike SP_{lin}, lower rank of M_a suggests better generalization.

Both AP and SP are important in measuring the computational quality of a liquid. For example, very high quantitative measure for SP and very low measure for AP is ideal. If one liquid has very high SP and a mediocre AP, it is hard to decide whether the particular liquid is better than another liquid with mediocre SP and a very small AP. Therefore, in order to compare the quality of different liquid configurations, a combined measure that involves both SP and AP is required. To address this, we use some insights from Fisher's Linear Discriminant Analysis (LDA) (Fisher, 1936; Fukunaga and Mantock, 1983; Hourdakis and Trahanias, 2013). LDA is utilized to find a linear combination ($f(.)$) of d features that characterizes or separates two or more classes (ω_i) of objects. The linear combination as shown in the equation below can be used as a classifier, or as a dimensionality reduction method before classification.

$$y_i = f(x_i) = Wx_i \tag{5}$$

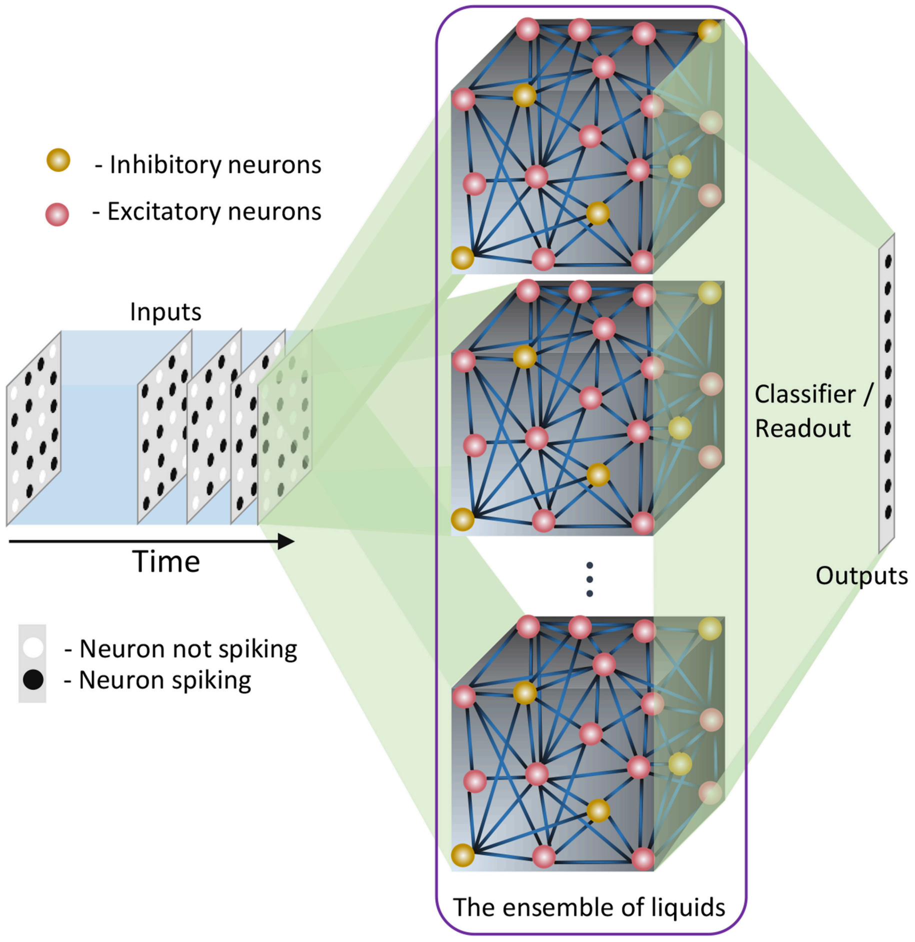


FIGURE 2 | The structure of the ensemble approach. The liquid in the standard LSM is split up to create an ensemble of smaller liquids. The input is sparsely connected to all the liquids. The output of all the liquids are concatenated to form one large liquid state vector, and connected to a single readout that is trained using supervised learning methods.

where y_i is the output vector ($Y = [y_1, ..., y_n] \in \mathbb{R}^{L \times n}$), that corresponds to the set of input features, x_i ($X = [x_1, ..., x_n] \in \mathbb{R}^{d \times n}$, each feature x_i is a column vector), $W \in \mathbb{R}^{L \times d}$ is the weight matrix that describes the linear relationship, L is the number of classes, and n is the number of data samples. The projection from LDA maximizes the separation of instances from different classes, and minimizes the dispersion of data from the same class, simultaneously, to achieve maximum class discrimination. The approximation capability is quantified by the matrix S_w called the "within class scatter matrix" that is specified by

$$S_w = \sum_{i=1}^{L} P(\omega_i)\hat{\Sigma}_i \tag{6}$$

where $P(\omega_i)$ is the probability of class ω_i, $\hat{\Sigma}_i$ is the sample covariance matrix (Park and Park, 2018) for class ω_i. The separation capability is given by the "between class scatter matrix" (S_b) that is described by

$$S_b = \sum_{i=1}^{L} P(\omega_i)(\mu_i - \mu_g)(\mu_i - \mu_g)^T \tag{7}$$

where μ_i is the sample mean vector (centroid) of class ω_i, and μ_g is the global sample mean vector. In classical LDA, the optimum weight matrix can be found by maximizing the objective function called Fisher's Discriminant Ratio (FDR) (Fukunaga and Mantock, 1983) that is computed as

$$FDR = tr(S_w^{-1}S_b) \tag{8}$$

where $tr(.)$ is the trace operation. For this work, the capability of the liquid to produce a good representation of the input data is quantified by a variant of the above ratio. The FDR is applied on the states of the liquid. However, when the data dimensionality (number of liquid neurons) is large in comparison to the sample size (n), the aforementioned scatter matrices tend to become singular (Ji and Ye, 2008) and the classical LDA cannot be applied. Hence, we use a modified discriminant ratio (DR) given by the following function:

$$DR = tr(S_b)tr(S_w)^{-1} \tag{9}$$

Note that the trace of S_w measures the closeness of each liquid state to its corresponding class mean as illustrated in **Figure 3A**, and the trace of S_b measures the distance between each class centroid and the global centroid in multidimensional space as depicted in **Figure 3B**. High $tr(S_b)$ suggests high SP (hence better) and smaller $tr(S_w)$ suggests better AP.

2.4. Experimental Setup

The performance of the ensemble of liquids approach is compared against a single liquid baseline detailed in section 2.1, with the aid of two spatial image recognition applications and two spatio-temporal speech recognition applications. The liquid was modeled in BRIAN (Goodman and Brette, 2008), a Python-based spiking neural network simulator, and the spiking activities of the neurons were recorded to calculate the liquid state vectors. The state vectors corresponding to the training input instances of each data set were then used to train a single fully-connected classification layer using the stochastic gradient descent algorithm (Robbins and Monro, 1985; Mei et al., 2018). The accuracy of the trained network was calculated on the testing data sets.

2.4.1. Data Sets Used for Illustration

The two spatio-temporal (speech) data sets used in this work are:

1. Digit sub-vocabulary of the TI46 speech corpus (Liberman et al., 1993) (TI-10)
2. TI 26-word "alphabet set"; a sub-vocabulary in the TI46 speech corpus (Liberman et al., 1993) (TI-alpha)

TI-10 consists of utterances of the words "zero" through "nine" (10 classes) by 16 speakers. There are $1,594$ instances in the training data set and $2,542$ instances in the testing data set. TI-alpha, on the other hand, has utterances of the words "A" through "Z" (26 classes). There are $4,142$ and $6,628$ instances in the training and testing data sets, respectively. For the spatial data sets (images), we used the handwritten digits from the MNIST (Deng, 2012) data set containing $60,000$ images of digits 0 through 9 in the training set and $10,000$ images in the testing set. In addition, we also created an extended MNIST data set that contains all the images from the original MNIST data set, and the same set of images transformed by rotation, shifting, and noise injection. It has $240,000$ images in the training data set and $40,000$ images in the testing data set.

2.4.2. Input Spike Generation

The first step is converting the images or the analog speech signals to spike trains to be applied as inputs to the liquid. For spatial data (images), there are p number of input spike trains fed in to the liquid, with p being the number of pixels in an image.

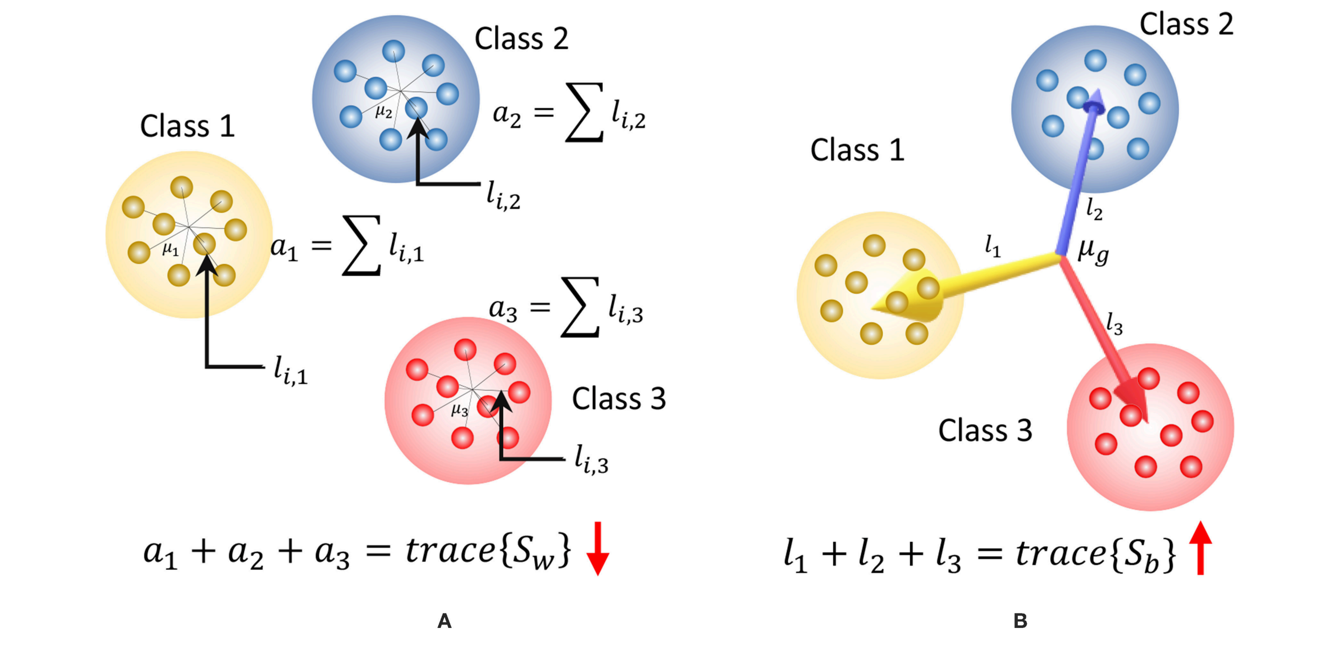


FIGURE 3 | The graphical representation of the components of the discriminant ratio (DR) for a set of two dimensional data points that belongs to three classes. **(A)** $tr(S_W)$ gives a measure of the addition of all the squared distances from the class means to each data point. This must be lower to have better approximation property. Here $l_{i,j}$ denotes the squared distance between the ith data point in class j to the class centroid, μ_j **(B)** $tr\{S_b\}$ gives a measure of the addition of the squared distances between the global mean and each class mean. High value for $tr\{S_b\}$ signals higher separation property. Here l_i denotes the squared distance from the global mean μ_g to the centroid of class i.

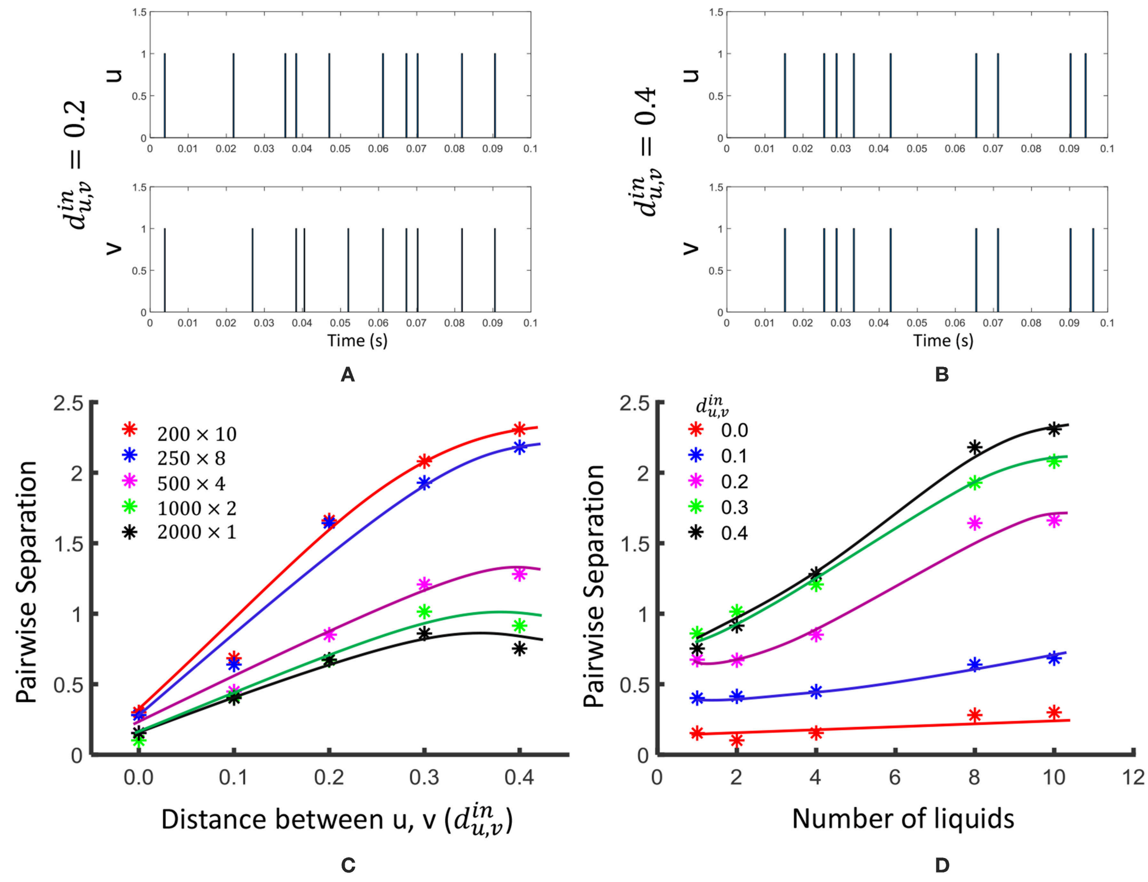


FIGURE 4 | The effect of dividing a large liquid on SP_{pw}, at different distances $d_{u,v}^{in}$ between inputs. Two input spike trains u and v are illustrated at **(A)** $d_{u,v}^{in} = 0.2$ and **(B)** $d_{u,v}^{in} = 0.4$. **(C)** The variation of pairwise separation with the distances between inputs, at different number of liquids **(D)** The variation of pairwise separation with the number of liquids, at different input distances $d_{u,v}^{in}$.

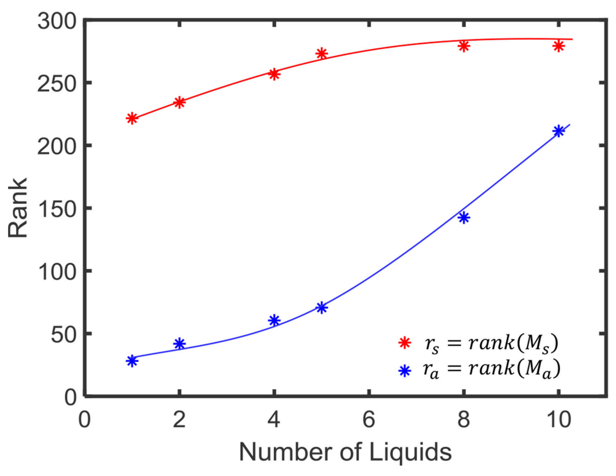


FIGURE 5 | The average rank of state matrix M_s that indicates the inter-class separability (in red) and the average rank of the matrix M_a which is an indication of the intra-class generalization capability (in blue).

The mean firing rate of each spike train is modulated depending upon the corresponding pixel intensity. Each input image pixel (ith pixel) is mapped to a Poisson distributed spike train with the mean firing rate (r_i for the ith image pixel) proportional to the corresponding pixel intensity (I_i) that is specified by

$$r_i = \frac{S_{count,i}}{T} \propto \left(\frac{I_i}{255}\right) \tag{10}$$

where $S_{count,i}$ is the number of spikes generated by the ith input neuron within a time period of T. For example, mean firing rate in this work for a white pixel (pixel intensity $I_i = 255$) is selected as 63.75 Hz. For a black pixel (pixel intensity $I_i = 0$), the mean firing rate is 0Hz. Each image is presented to the liquid for a duration of 300 ms ($= T$).

For the speech data, the audio samples available in wave format were preprocessed based on Lyon's Passive Ear model (Lyon, 1982) of the human cochlea, using Slaney's MATLAB auditory toolbox (Slaney, 1998). The model was used to convert each audio sample to temporal variation in the intensity of 39

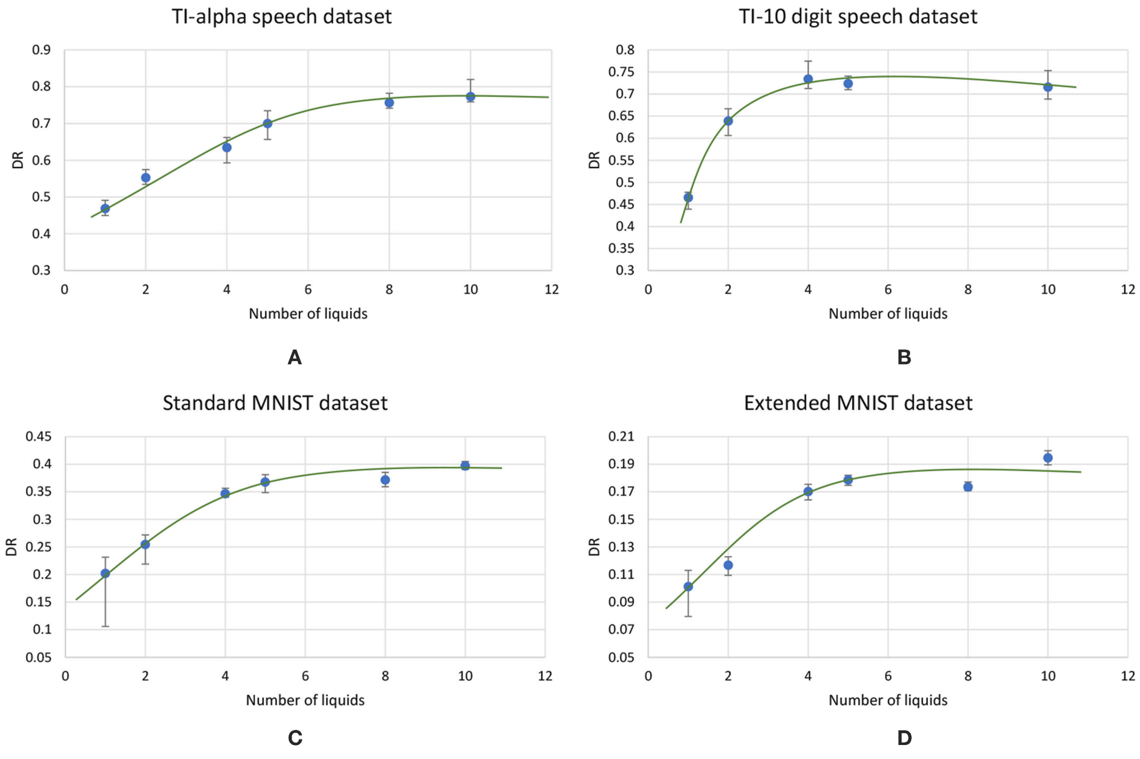


FIGURE 6 | The average (over five trials) discriminant ratio (*DR*) trends with different number of liquids in an ensemble for two speech recognition tasks; **(A)** TI-alpha dataset, **(B)** TI-10 dataset, and two image recognition tasks; **(C)** standard MNIST dataset, **(D)** extended MNIST dataset. The total number of neurons in each ensemble of liquids were kept the same. Note that all the *DR* trends increase with the number of liquids, and saturates after a certain point, that depends on the application.

frequency channels. These intensity values at each time step j ($I_{i,j}$) were then normalized and used as the instantaneous firing probability of an input neuron i ($i = \{1, 2, ..., 39\}$). The time step in this work is 0.5ms.

3. RESULTS

3.1. The Kernel Quality Improvement Due to the Ensemble Approach

In this section, we will explore the effects of dividing a large liquid, by means of standard measures for SP and AP explained in section 2.3. We involve the same general tasks suggested in Maass et al. (2005); Roy and Basu (2016), to compare the SP and AP. In order to measure the pointwise separation property, we generated 100 input spike trains $u(t)$ and $v(t)$ with different distances $d^{in}_{u,v}$ between them. The distance between two input spike trains is evaluated according to the methodology explained in Maass et al. (2005). The two spike trains were first filtered with a Gaussian kernel $e^{-(t/\tau_{in})^2}$, and then the Euclidean distance between them were measured. τ_{in} was selected as 5 *ms* (Maass et al., 2002).

$$d^{in}_{u,v} = \frac{||u(t) * e^{-(t/\tau_{in})^2} - v(t) * e^{-(t/\tau_{in})^2}||}{T} \tag{11}$$

The same 100 $u(t)$ and $v(t)$ signals were fed in to LSMs with different number of liquids ($N_{ens} = 1, 2, 4, 8, 10$), and the pairwise separation property was calculated according to Equation 3. The average SP_{pw} was evaluated over 10 different weight initialization trials and the results are shown in **Figure 4**. As the figure illustrates, the SP_{pw} improves with the distance $d^{in}_{u,v}$ between two inputs, and also with the number of liquids in the LSM.

For the linear separation property, we applied 400 randomly generated input signals $u_i(t)$ to LSMs with different number of liquids ($N_{ens} = 1, 2, 4, 8, 10$). The resultant states ($x_{u_i}(t_0)$) were used to create the matrix M_s explained in Equation 4. The average SP_{lin} ($= rank(M_s) = r_s$) was evaluated among five different sets of inputs and five different weight initializations (i.e., 25 trials altogether) and the results are finalized in **Figure 5**. As the figure illustrates, the SP_{lin} increases with the number of liquids. However, the rate of increment of SP_{lin} reduces with the increasing number of liquids.

For the generalization property, we conducted same above experiment with a different state matrix M_a. To create this matrix, we involved 400 jittered versions of the input signal $u_i(t)$, ($u^j_i(t)$) as explained in 2.3. In order to create a jittered version of $u_i(t)$, we shifted the spike times by a small delay Δt taken from a Gaussian distribution as explained in Maass et al. (2002). The average rank of the matrix M_a is shown in **Figure 4**. A lower rank of M_a (r_a) suggests better approximation of intra-class input examples. According to the figure, r_a increases with the number

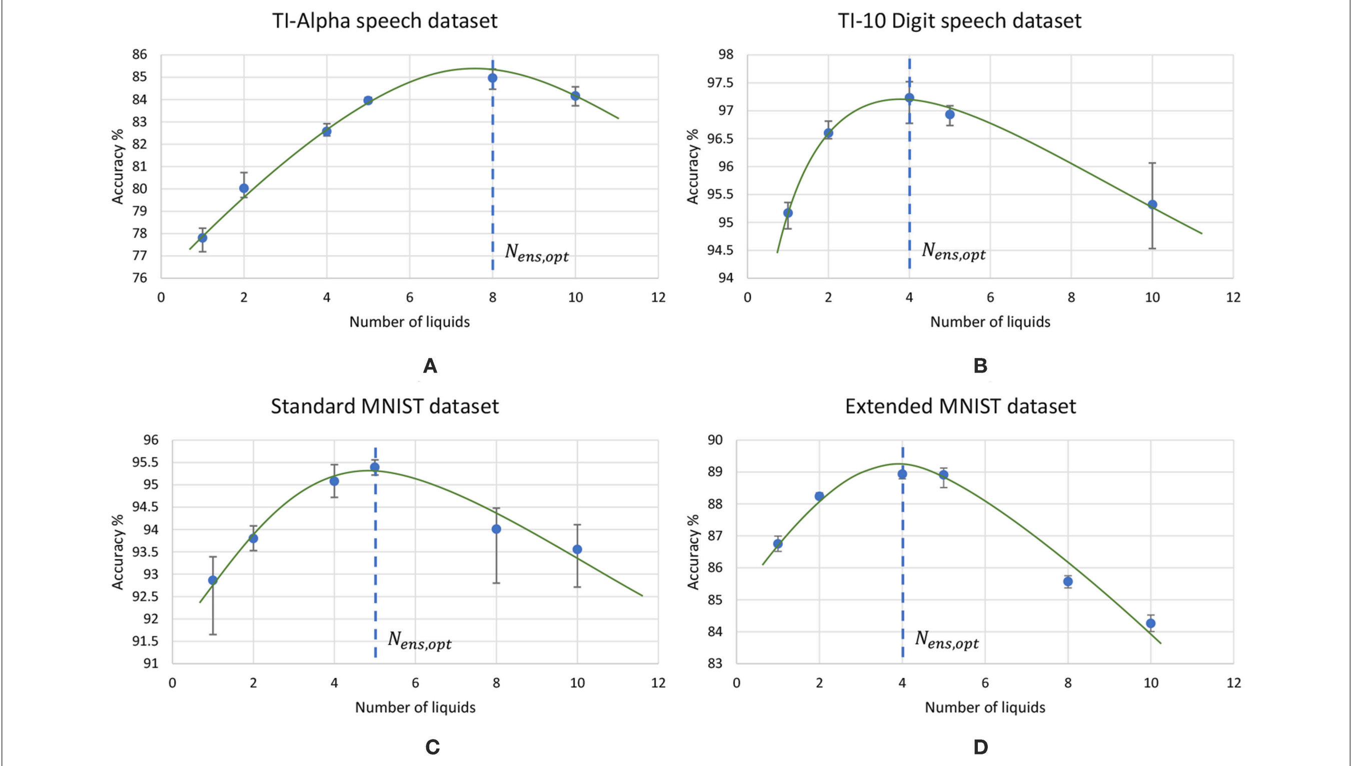


FIGURE 7 | The average (over five trials) accuracy (percentage) trends with different number of liquids in an ensemble, for two speech recognition tasks; **(A)** TI-alpha test dataset, **(B)** TI-10 test dataset, and two image recognition tasks; **(C)** standard MNIST test dataset, **(D)** extended MNIST test dataset. The total number of neurons in each ensemble of liquids were kept the same. Note that all the accuracy trends peak at a certain point, that depends on the application.

of liquids. This signals the liquid losing its ability to generalize intra-class inputs.

We observed that the SP_{pw} improves by 3.06 in the 10-liquid ensemble approach, when comparing with the single liquid baseline. The SP_{lin} improvement is 1.26×. For a similar set of experiments (for $N_{tot} = 135$), the authors in Roy and Basu (2016) explored the kernel quality of an LSM of which the reservoir connections were trained using a structural plasticity rule (Roy and Basu, 2017). The reported improvement in SP_{pw} is 1.36×, whereas the improvement in SP_{lin} is 2.05× when compared with a randomly generated traditional LSM. It is noteworthy that when training using structural plasticity, the inter-class separation capability can be improved, with respect to a traditional liquid with random connections. Without involving such complex learning techniques, one can obtain improved separation by simply dividing a liquid as shown in our work. However, note that such reservoir connection learning methods can simultaneously preserve the ability of the LSM to approximate intra-class examples, which is not attainable by the ensemble approach, at higher number of liquids. As explained in section 2.3, the ability of a liquid to produce a better representation of an input is a measure of both SP and AP. In the next section, we will explore this combined measure of SP and AP defined as *DR* in section 2.3, on real world spatio-temporal data classification tasks.

3.2. Impact of the Ensemble Approach on Accuracy of Different Applications

Using the experimental setup explained in the previous section 2.4, we initially simulated our baseline single liquid LSM (section 2.1) for the four data sets. We used 500, 2,000, 1,000, and 1,000 neurons in total within the liquid for the TI-10, TI-alpha, standard MNIST, and extended MNIST pattern recognition tasks, respectively. We refined the percentage connectivity for each task as shown in **Table 1**. The classifier was trained using the liquid states corresponding to the training examples, and the classification accuracy of the trained network was obtained for unseen instances from the test data set. For each application, we then created an ensemble of liquids with $\frac{N_{tot}}{N_{ens}}$ number of neurons in each small liquid. For all the four applications, we evaluated the SP and AP for different number of liquids in the ensemble ($N_{ens} = 1, 2, 4, 5, 8, 10$) and quantified how good is the input representation of the ensemble of liquids using *DR* (explained in section 2.3). **Figure 6** shows that the *DR* increases up to a certain number of liquids in the ensemble and then saturates for the four different applications we have considered. This signals that the ensemble of liquids, in principle, gives a better representation of the input with increasing number of liquids until a certain point. In order to verify whether this improvement in the *DR* actually implies an improvement in classification accuracy, we evaluated the LSM accuracy for different number of

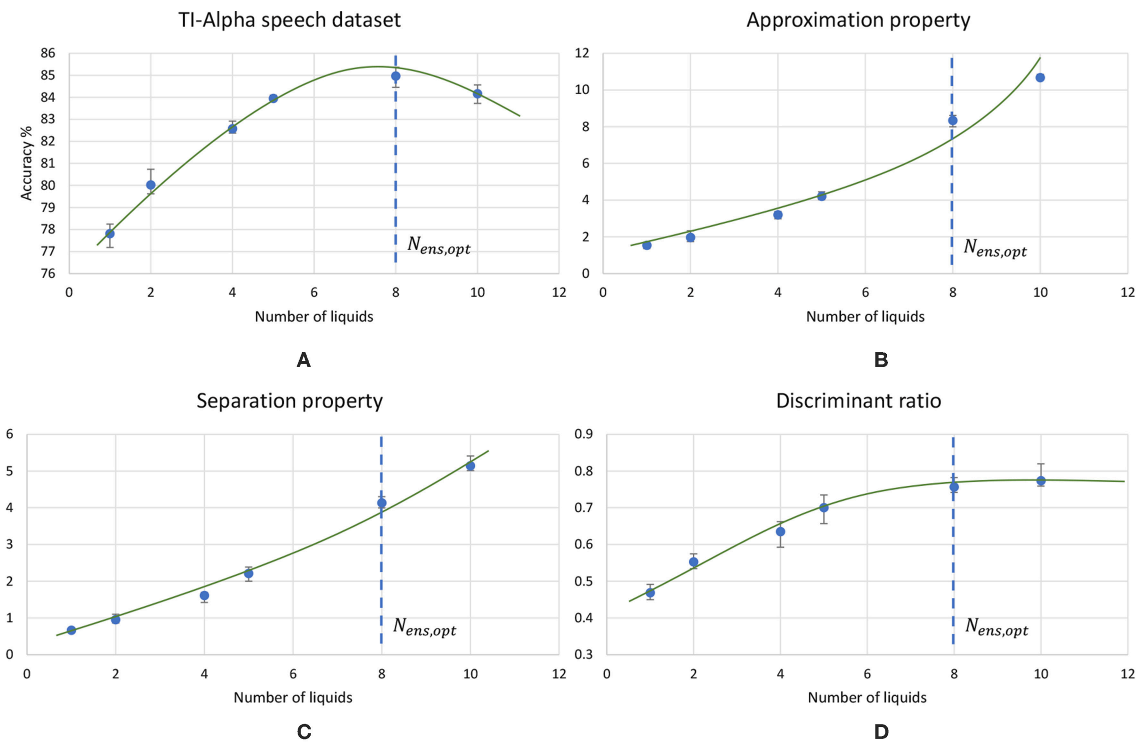


FIGURE 8 | The trends of different measures associated with LSMs, with the increasing number of liquids. **(A)** Accuracy, **(B)** Approximation property (AP), **(C)** Separation property (SP) and **(D)** Discriminant ratio (*DR*). The LSM is trained for TI-alpha speech recognition application. Both AP and SP continuously increases with the number of liquids. Note that the increment in AP is more significant than that of SP for larger number of liquids.

liquids ($N_{ens} = 1, 2, 4, 5, 8, 10$) for the four different classification applications. **Figure 7** shows that the accuracy indeed improves with the number of liquids until a certain point. Let us denote this point as the "peak accuracy point" and the corresponding number of liquids for that point as the "optimum number of liquids" ($N_{ens,opt}$). We noticed that the $N_{ens,opt}$ is a function of the application, and that increasing N_{ens} beyond $N_{ens,opt}$ actually results in accuracy loss. When comparing **Figures 4**, **7**, it is evident that the point at which the *DR* saturates is the same as $N_{ens,opt}$. This explains that dividing a large liquid into multiple smaller liquids enhances the class discrimination capability of the liquid, leading to improved classification accuracy. However, note that after the $N_{ens,opt}$ point, the *DR* saturates whereas the accuracy degrades. The *DR* does not offer a direct mapping of the accuracy of an LSM. However, it could still be utilized as a measure of identifying the point at which the accuracy starts to drop ($N_{ens,opt}$). This is the same point at which the *DR* stops improving.

Figure 8 plots the variation of the individual *DR* components; separation $\left(SP{=}tr(S_b)\right)$ and approximation $\left(AP{=}tr(S_w)\right)$ with the number of liquids for the TI-alpha speech recognition task. **Figure 8** shows that SP improves continuously with the number of liquids. Improved separation suggests larger dispersion among the centroids of the liquid states corresponding to instances from different classes, which renders the input representations provided by the liquid easier to classify. This is illustrated in the cartoon in **Figure 9A** for a set of two-dimensional data points from two classes, wherein higher SP while maintaining the same AP results in enhanced class discrimination capability. At the same time, **Figure 8** indicates that AP also increases with the number of liquids, implying that larger number of liquids leads to higher dispersion between projected inputs from the same class. Higher AP for a given SP is not desirable since it could potentially lead to overlap among instances belonging to different classes as depicted in **Figure 9B**, thereby degrading the class discrimination capability. Since both SP and AP increases, the ratio *DR* gives a better measure about the overall effect of the proposed ensemble approach on the classification accuracy of the LSM rather than the individual components *per se*. As shown in **Figure 8**, the *DR* increases until a certain number of liquids, signaling the dominance of the improvement in SP over the degradation in AP as graphically illustrated in **Figure 9C**. In contrast, as the number of liquids is increased beyond $N_{ens,opt}$, *DR* saturates since the increment in SP is no longer sufficient to compensate for the degradation in AP as shown in **Figure 8**. When the dispersion between classes (due to increment in SP) is not sufficient to compensate for the dispersion occurring for instances within the same class (due to AP degradation), there can be overlaps among class boundaries as depicted in **Figure 9D**, leading to accuracy loss as experimentally validated in **Figure 7** across different applications.

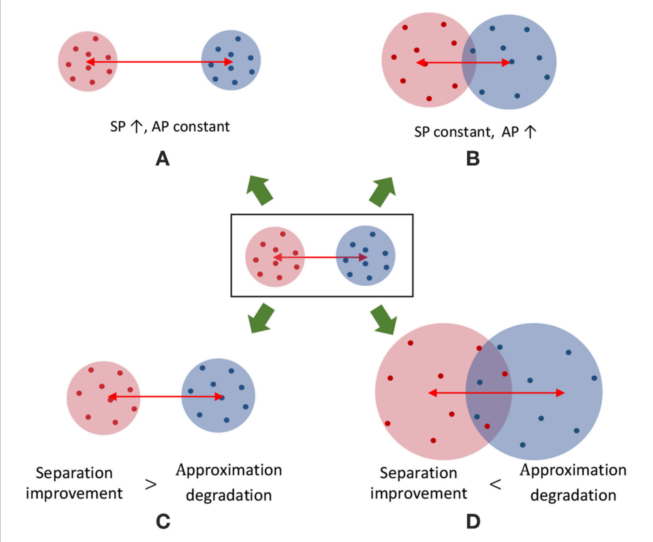


FIGURE 9 | A cartoon that shows the distribution of two dimensional data points that belong to two classes under different conditions. **(A)** A case with increased SP while maintaining the same AP. **(B)** A case where the AP is increased while maintaining the same SP. Note that the class boundaries can get overlapped leading to classification errors. Hence, increased AP is not desirable. **(C,D)** shows two scenarios where both SP and AP increased from the baseline distribution of data points (figure in the middle). **(C)** The improvement in SP is larger than the degradation in AP. **(D)** The improvement in SP is not sufficient to compensate for the AP degradation, leading to overlapped class boundaries.

In order to graphically view the variation in SP and AP with the number of liquids for the applications considered in this work, we used Principal Component Analysis (PCA) to plot the high-dimensional liquid states in a low-dimensional space. Generally, the first few principal components preserves the most variance in a given high-dimensional data set. Hence, the same object in multi-dimensional space can be visualized in low-dimensional space with insignificant changes. To create such a low-dimensional projection of the liquid state vectors for different input patterns, we reduced their dimension using PCA and plotted the two most significant Principal Components (PCs) corresponding to the two largest eigenvalues. **Figure 10** plots the 800-dimensional liquid state vectors, projected to the two-dimensional space using the first two PCs, for $1,000$ randomly picked input patterns from three classes in the MNIST data set. **Figure 10** clearly illustrates why the accuracy improves till the $N_{ens,opt}$ point and degrades beyond that as explained below. The single liquid case shows concentrated (low AP), but overlapped data (low SP). This is where the AP is the lowest due to the concentrated data points. As the number of liquids increases, the classes become clearly separated. Note that the points belonging to the same class also moves away from their respective centroids due to the increased AP. This ultimately results in the aforementioned overlapping between the classes for number of liquids larger than $N_{ens,opt}$, which gives rise to more misclassifications.

3.3. Benefits of the Ensemble Approach

The ensemble of liquids approach creates smaller liquids where the dynamics of one network does not affect another. When evaluating the spike propagation within the liquids, these smaller liquids can be run independently and in parallel. Since the evaluation time is a higher order polynomial function of the number of neurons, computing few smaller liquids in parallel instead of computing one large liquid is beneficial in terms of reducing the inference time. Note that the evaluation of a large liquid can also be parallelized. The liquid dynamics vary temporally, and for digital simulations, it can be divided in to multiple time steps. Each evaluated neuron state in the liquid at one time step is temporally correlated to that of the next time step. Therefore, the liquid evaluation process cannot be temporally divided for parallelizing the operation. Furthermore, since all the neurons are connected to each other (with a given sparsity), the dynamics of one neuron is dependent upon that of other neurons connected to it. Therefore, "fully independent" simulations are also not possible at the neuron level. However, the matrix-vector manipulations involved in each time step *can* be parallelized. Simply put, in finding the pre-synaptic currents of the neurons, the matrix-vector multiplication between the spiking activity and the weight matrix must be evaluated as shown below (with respect to excitatory neurons for example).

$$\Delta g_e(t_i) = WS(t_i) \tag{12}$$

where $\Delta g_e(t_i)$ is the instantaneous jump of conductance at time t_i (refer to Equation 2), $S(t_i)$ is the spiking activity vector of N number of neurons in the liquid at time t_i, and $W \in \mathbb{R}^{N\times N}$ is the connection matrix that defines the liquid. Consider dividing the above process in to multiple processing cores. The division of the operation in to two cores using row-wise striped matrix decomposition requires the matrix W to be divided in to two parts (**Figure 11A**). During each simulation time step (t_i), each core evaluates membrane potentials $S_1(t_{i+1}) = \left[s_1(t_{i+1}), ..., s_{N/2}(t_{i+1})\right]$ and $S_2(t_{i+1}) = \left[s_{N/2+1}(t_{i+1}), ..., s_N(t_{i+1})\right]$. For the next time step, these S_1 and S_2 must be concatenated and requires communication between cores. In contrast, a concatenation is not required until the end of the total simulation duration (T) in our ensemble approach (**Figure 11B**). Due to the lack of communication overhead between processors, the ensemble approach is faster than a parallelized version of the single liquid baseline among N_{ens} number of processors. In fact, due to the aforementioned communication overheads, efficient parallel processing can be hindered even in Graphical processing units (GPUs)(Kasap and van Opstal, 2018). However, in any method of evaluating the liquid dynamics, note that the ensemble approach has less number of connections than a single liquid baseline. Therefore, the ensemble approach has reduced amount of computation leading to lower evaluation time. Different studies have shown designing hardware accelerators for spiking neural network platforms (Wang et al., 2014, 2017; Du et al., 2017). In the context of reducing the design complexity, above methods could potentially benefit from the low connection complexity, and

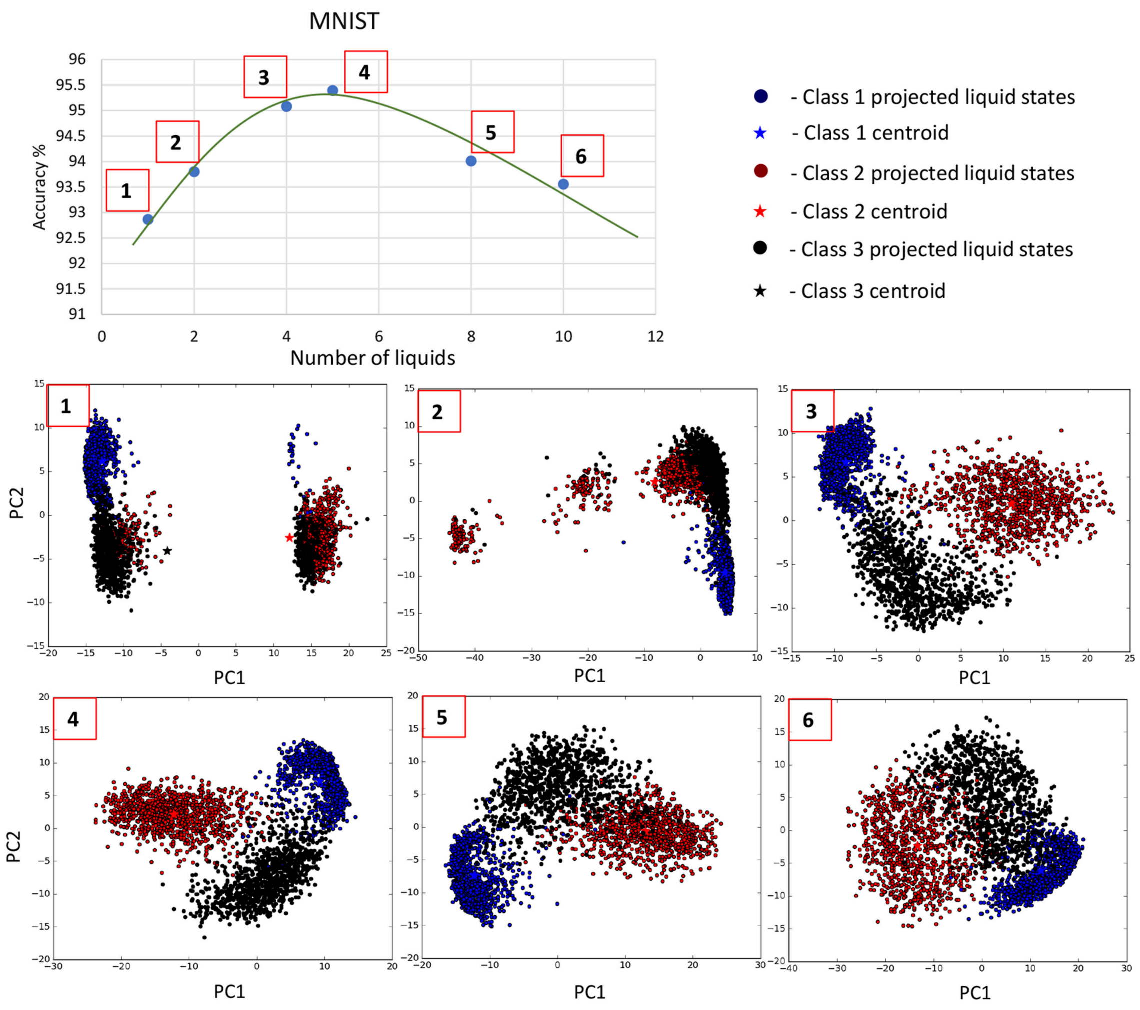


FIGURE 10 | The distribution of the liquid state vectors, as a projection to the first two principal components PC1 and PC2, for different number of liquids. The liquid state vectors (represented as dots) correspond to three classes in the MNIST image data set. Each class has randomly picked 1,000 liquid state vectors. Distributions related to point $\boxed{3}$ and $\boxed{4}$ show less overlapping between classes, and the data points are more concentrated at the class mean points in contrast to $\boxed{6}$, which has significant overlapping that caused the accuracy degradation.

"embarrassingly parallel" nature (Herlihy and Nir, 2011) of our ensemble approach.

The inference time is the addition of the liquid evaluation time and the classifier evaluation time. The liquid evaluation time was calculated by giving 100 input instances to the LSM model solver and estimating the average liquid computation time per input. The classifier evaluation time is significantly lower than the liquid computation time ($\sim \times 10$). Note that the classifier training time is similar in the baseline (single liquid LSM) and the ensemble approach, since there are equal number of neurons in the liquid and the number of trained weights are the same.

Once an LSM is trained, the connections within the liquid and the classifier weights must be stored. LSMs with large liquids require more space. In the ensemble approach, the number of connections within the liquid are significantly lower than the single liquid baseline. For example, assume dividing a liquid with N_{tot} number of neurons in to N_{ens} number of smaller liquids with $\frac{N_{tot}}{N_{ens}}$ amount of neurons in each of them. The number of connections available within the liquid for the single liquid baseline is $\sim N_{tot}^2$ whereas the number of connections in the multi-liquid case is $\sim (\frac{N_{tot}}{N_{ens}})^2 N_{ens} = \frac{N_{tot}^2}{N_{ens}}$. This shows that the number of connections reduces by a factor of N_{ens} when dividing a large liquid into N_{ens} smaller liquids, given that the percentage connectivity stays the same. **Figures 12A,B** illustrate how the memory requirement varies for different number of liquids for the MNIST image recognition and TI-alpha speech recognition applications, respectively. When the optimum accuracy point for the ensemble approach is considered, we witnessed 87% reduction in the amount of memory, 55% reduction in inference time, and a 7.3% improvement in accuracy simultaneously, for the TI-alpha speech recognition application. For the MNIST handwritten digit recognition application, we witnessed 78% reduction in the amount of memory, 72% reduction in the inference time, and 3.9% improvement in classification accuracy.

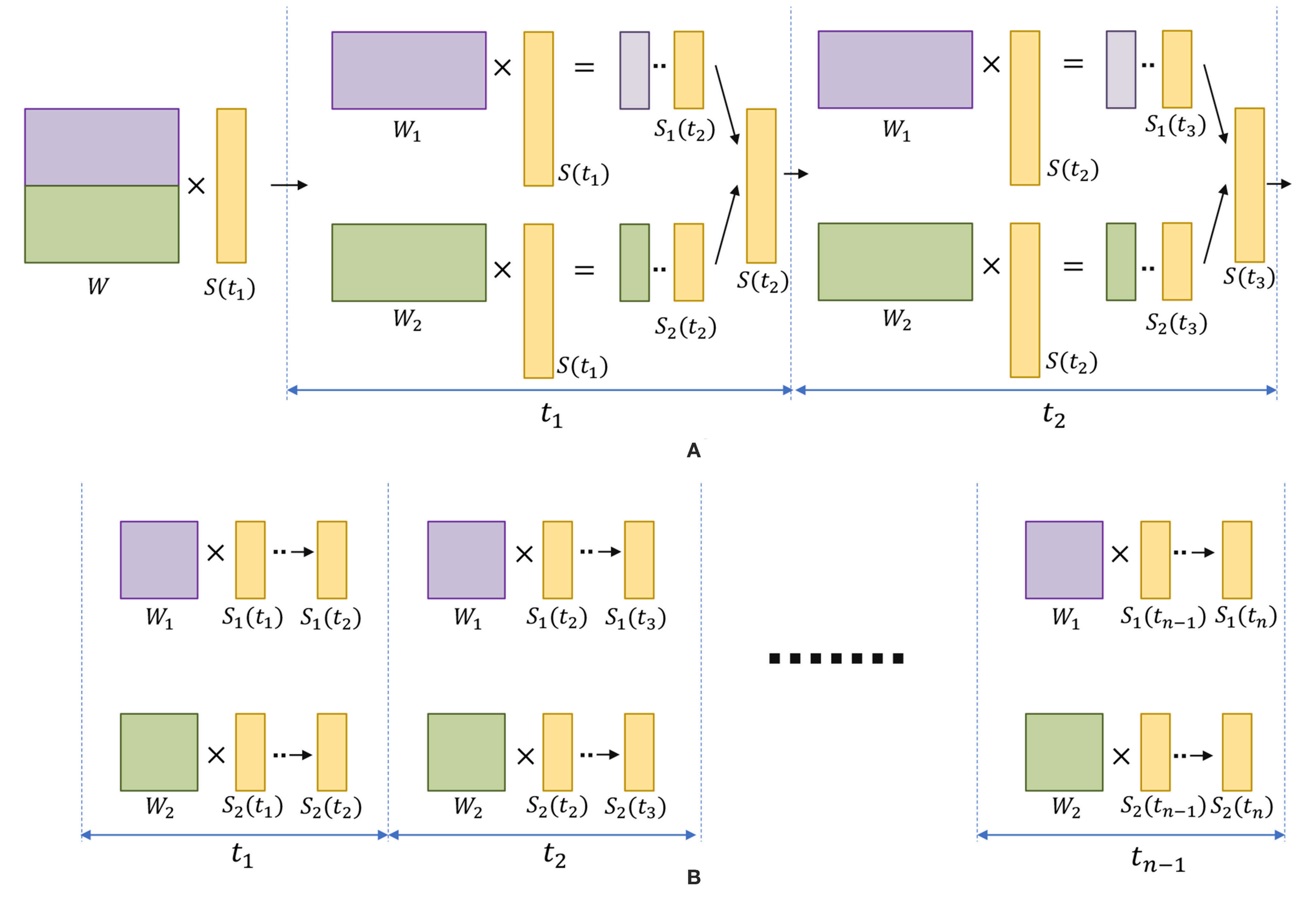


FIGURE 11 | (A) The division of matrix-vector multiplication using row-wise striped matrix decomposition, for the single liquid baseline LSM. Note that during each time step, the generated S_1 and S_2 vectors need to be concatenated to form the S vector (represents the spiking activity of the liquid), which requires communication between cores. **(B)** The "embarrassingly parallel" nature, and the reduced amount of operations in the ensemble approach allows two small liquids to run in parallel as two independent tasks, until the end of the last simulation time step t_n.

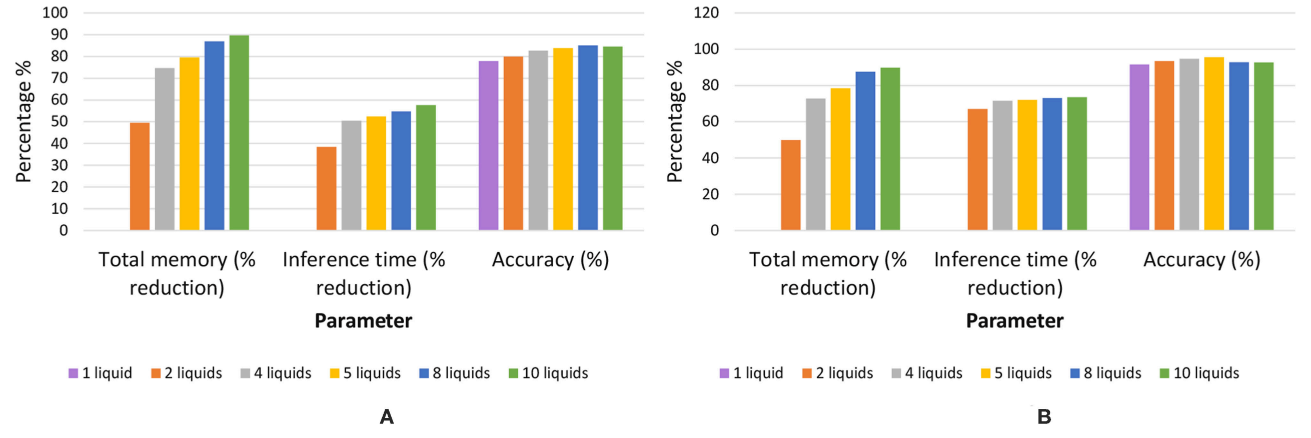


FIGURE 12 | The total memory reduction (%), inference time reduction (%) with respect to the baseline, and accuracy for different number of liquids in the ensemble. Two applications were considered; **(A)** temporal data classification problem (TI-alpha) **(B)** spatial data classification problem (MNIST).

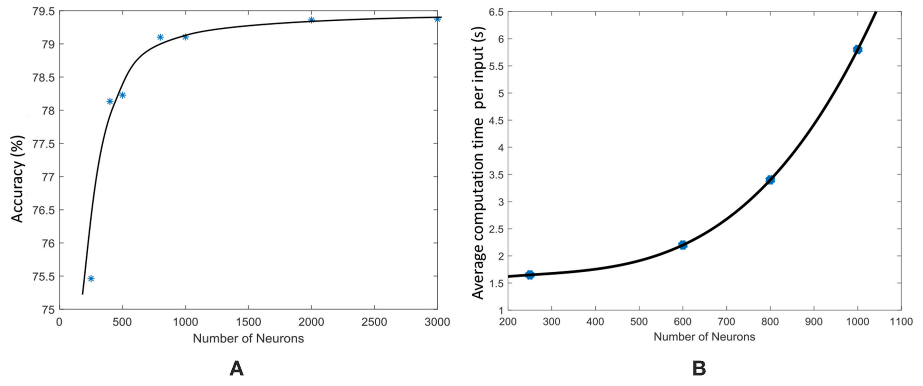


FIGURE 13 | (A) The accuracy of an LSM with a single liquid, measured at different number of neurons, for a speech recognition application (TI-alpha). **(B)** The average liquid evaluation time of an LSM measured at different number of neurons.

3.4. Conventional Methods of Improving the Accuracy vs. the Ensemble Approach

The simple structure and training of LSMs, come with an accuracy trade-off, when compared with other non-reservoir computing techniques such as LSTM networks (Bellec et al., 2018). Different mechanisms have been studied in the literature such as training the connections in the reservoir (Xue et al., 2016), using expensive learning rules (for example, backpropagation through time Bellec et al., 2018), and selecting complex architectures (Wang and Li, 2016), in order to improve the accuracy of liquid state machines. However, these methods will increase the complexity of the LSM resulting in poor performance with respect to latency, despite the higher accuracy. Furthermore, a liquid can be considered as a universal computational medium. A single liquid with multiple trained readouts can be used for multiple applications(Wang et al., 2015). Above methods such as training the connections within the liquid will make the LSM restricted to one application. In this section, we will explain two basic methods of improving accuracy, while leaving the structural and training simplicity of LSMs intact, and compare the results with the ensemble approach.

3.4.1. Increasing the Number of Neurons in the Liquid

As explained in Maass et al. (2003), sufficiently large and complex liquids possess large computational power. Increased number of neurons in the liquid will result in increased number of variables for the classifier. Based on “multiple linear regression” methods of predicting a function, increased number of predictor variables (in this case the number of neurons), will result in better prediction (Krzywinski and Altman, 2015; Wijesinghe et al., 2017). Therefore, increasing the number of neurons will improve the prediction accuracy of the LSM. Note however that using enormous number of predictor variables/neurons will make the network suffer from overfitting. **Figure 13A** shows how the accuracy of an LSM varies with the number of neurons in the reservoir for the TI-alpha speech recognition task. As **Figure 13A** illustrates, the accuracy initially increases with the number of neurons and then saturates after a certain point. Increased number of neurons implies increased connections within the liquid, given that the percentage connectivity stays the same. The number of connections within the liquid shows a square relationship $\sim \nu N_{tot}^2$ with the number of neurons N_{tot}, where ν is the global percentage connectivity. Due to this, evaluation time of the liquid increases exponentially as shown in **Figure 13B**. Therefore, when the number of neurons are already high, the accuracy improvement we obtain by further increasing the number of neurons is not worth the resultant performance and storage requirement penalty. Furthermore, the accuracy saturates around $\sim$ 79.2% for the TI-alpha application (for $N_{tot} \geq 800$). Note that we have also adjusted the percentage connectivity at each point in the graph, to get the best accuracy for a given number of neurons. However, the ensemble approach for $N_{tot} = 1000$ and $N_{ens} = 4$ gives $\sim$ 83% accuracy, which is larger than the accuracy achievable by increasing the number of neurons in a single liquid.

3.4.2. Percentage Connectivity Within the Liquid

The percentage connectivity within the LSM is an important measure of the spiking activity of a liquid. The spiking activity of the liquid could show two negative behaviors which could drastically reduce the accuracy of the network, *viz*. pathological synchrony and over-stratification (Norton and Ventura, 2006). Pathological synchrony occurs when the neurons get caught in infinite positive feedback loops resulting in heavy continuous spiking activity. Over-stratification can be defined as the opposite extreme of the above. Here, the neurons do not propagate an input signal properly, resulting in reduced spiking activity. Both the above behaviors result in similar outcomes for input instances of different classes (hence poor separation between classes), making classification tasks hard. We noticed that lower connectivity ($P_{In\rightarrow E}$) results in over-stratification (**Figure 14A**) whereas higher connectivity results in pathological synchrony (**Figure 14B**).

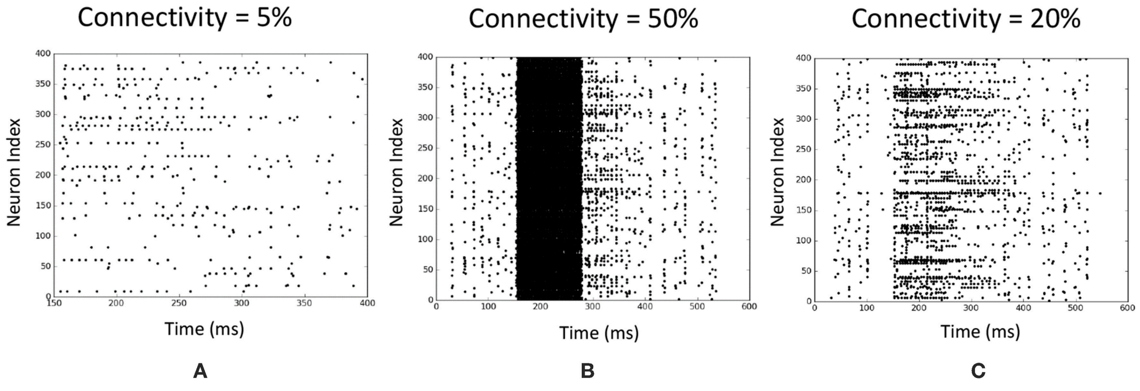


FIGURE 14 | Illustration of two negative behaviors of an LSM at different input to liquid percentage connectivity values. Each raster plot shows the spiking activity of the liquid neurons over time. The application is a speech recognition task (TI-alpha). **(A)** Over-stratification at low percentage connectivity. **(B)** Pathological synchrony at higher percentage connectivity. **(C)** An instance that shows clear differences between spiking activity of the liquid neurons in contrast to **(A,B)**.

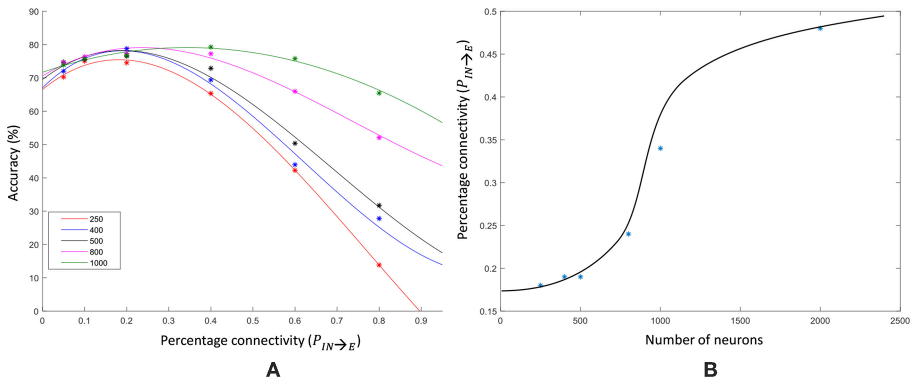


FIGURE 15 | (A) The accuracy trend with varying input–liquid percentage connectivity, for different number of liquid neurons. The experiment is done on a single liquid LSM conducting a speech classification task (TI-alpha). **(B)** The percentage connectivity that gives the best accuracy at different number of neurons.

We changed the percentage connectivity between different combinations of pre- and post-neuron types ($E-E, I-E, E-I$) till we obtain good accuracy avoiding pathological synchrony and over-stratification (**Figure 14C**). After that, we refined the input-liquid connectivity for further accuracy improvement. **Figure 15A** shows how the accuracy changes with the percentage connectivity of the input to liquid connections. Liquids with different number of neurons have different optimum connectivity values as shown in **Figure 15B**. The application is recognizing spoken letters in TI-alpha speech corpus. The maximum accuracy achievable by changing the percentage connectivity ($P_{IN \to E}$) for $N_{tot} = 1,000$ is $\sim$ 79% (refer to the green colored trend in **Figure 15A**). This is smaller than that achievable ($\sim$ 83%) by our ensemble approach with $N_{tot} = 1,000$ and $N_{ens} = 4$.

Furthermore, we simultaneously changed the $P_{E \to E}$ and $P_{IN \to E}$ percentage connectivity values of LSMs with different number of liquids, and evaluated the accuracy. Four $P_{IN \to E}$ values $(0.1, 0.2, 0.4, 0.6)$ and three $P_{E \to E}$ values $(0.2, 0.4, 0.6)$ were selected for the experiment. The summarized results are illustrated in the 3D plot in **Figure 16**. The color code of the figure gives the accuracy of a particular combination of connectivity values. Across all LSM configurations with different number of liquids, we witnessed that higher $P_{IN \to E}$ and higher $P_{E \to E}$ results in accuracy degradation. Sparser connectivity gives better results. As the figure illustrates, at sparser connectivity values, a single liquid LSM offers lower accuracy than an LSM with N_{ens} liquids (refer to the upper left corner of the 3D plots). The "maximum capacity" of each LSM configuration (for a given number of liquids) is plotted in **Figure 17A**. The "maximum capacity" is the best accuracy attainable from a particular liquid configuration, after optimizing the percentage connectivity values (in the selected range). As **Figure 17A** illustrates, maximum accuracy obtained from the single liquid configuration is smaller than that of other configurations. We also plotted the average accuracy of a given LSM configuration across all percentage connectivity values (**Figure 17B**). The average accuracy to some extent could be thought of as the outcome one would witness in a given LSM configuration for an arbitrarily selected connectivity value (within the specified sparse

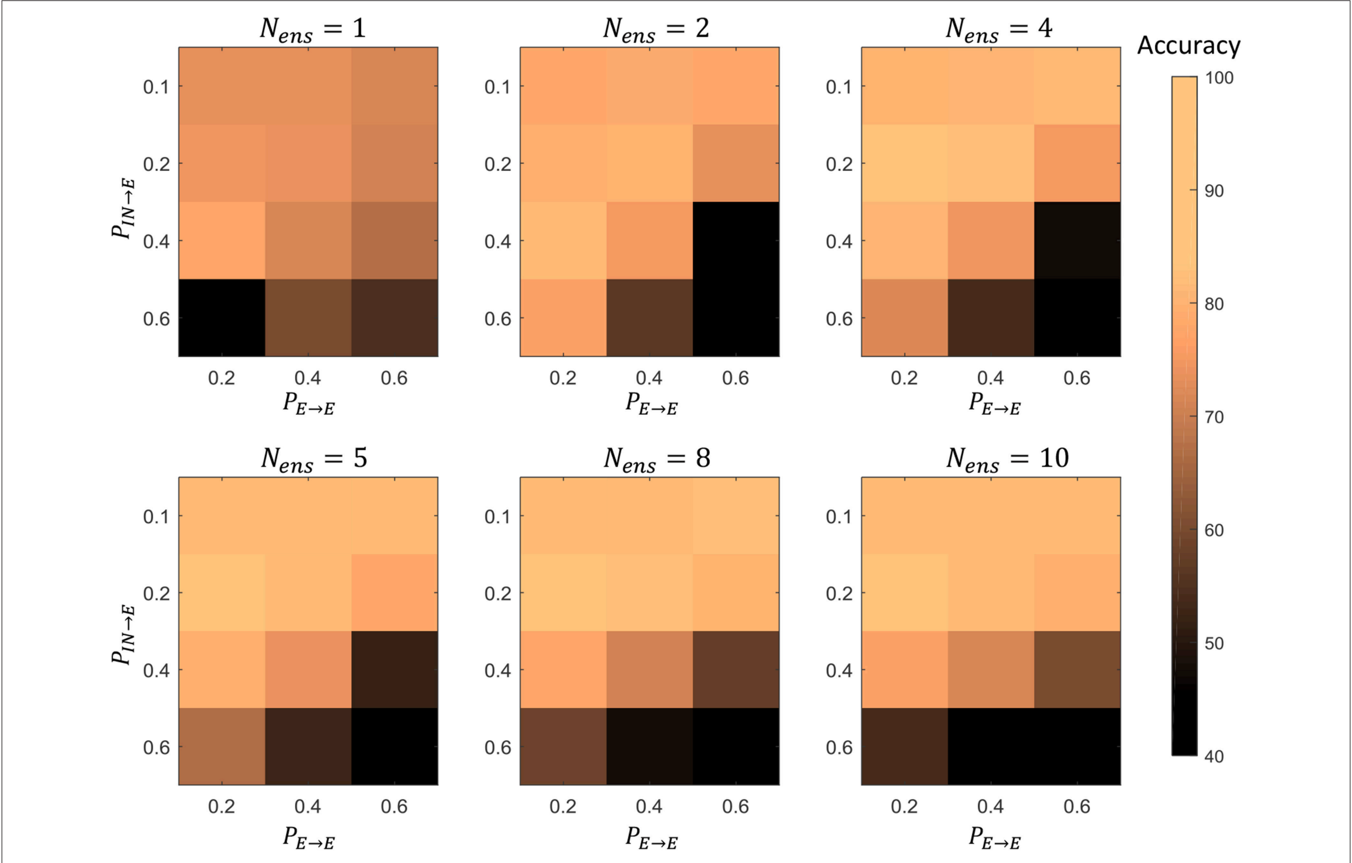


FIGURE 16 | The accuracy of LSMs with $N_{ens} = 1, 2, 4, 5, 8, 10$ at different percentage connectivity ($P_{E\to E}$ and $P_{IN\to E}$) values, for the TI46-alpha classification task.

connectivity region of the experiment). The average accuracy of the single liquid LSM configuration is lower than that of multiple liquids.

In section 3.3, we explored the benefits of the ensemble approach due to reduced number of connections in the liquid. A single liquid LSM configuration has N_{ens} times more number of connections as the LSM with N_{ens} number of liquids as explained in section 3.3. In order to view if a single liquid with sparser connectivity offers better accuracy than an LSM with N_{ens} number of liquids and higher percentage connectivity, we conducted an experiment. In other words, the goal of the experiment is to view the accuracy of two LSM configurations with same number of connections. The dominant component of the number of connections in an LSM is the connections between the excitatory neurons. Therefore, we varied the $P_{E\to E}$ for two LSM configurations ($N_{ens} = 1$ and $N_{ens} = 4$) and observed the accuracy for the TI46-alpha application. **Figure 18** illustrates that for $P_{E\to E} < 0.57$, the multiple liquid configuration ($N_{ens} = 4$) provides better accuracy, suggesting that the ensemble approach gives better results even under same number of connections in comparison to the single liquid baseline. For example, the ensemble approach gives $\sim$ 83% accuracy at $P_{E\to E} = 0.4$ and for the same number of connections, (i.e., at $P_{E\to E} = 0.1$), the single liquid LSM configuration gives lower accuracy ($\sim$ 76%). However, for $P_{E\to E} > 0.57$, single liquid LSM seems to perform better. Hence we conclude that at higher degrees of sparsity, the ensemble approach performs better than a single liquid baseline with the same number of connections.

Apart from the percentage connectivity, different connectivity patterns within the liquid were also considered in literature. For example, a probabilistic local connectivity within the liquid, inspired by the connectivity in biological neurons is suggested in Maass et al. (2003). We conducted an experiment with different sets of parameters (refer to the **Supplementary Material**) for the probabilistic local connectivity model. Our results indicate that, the highest accuracy achieved (for the ranges of parameters we have considered) with the probabilistic local connectivity model (an LSM with 1,008 neurons arranged in a liquid column $6 \times 6 \times 28$, gave a maximum accuracy of $\sim$ 78%, for the TI-alpha speech recognition application) is lower than that attainable from our proposed ensemble approach (4 ensembles with 250 neurons in each, resulted in an accuracy of 83%, for the same TI-alpha application). More information on our analysis is included in the **Supplementary Material**.

3.5. Limitations of the Ensemble Approach

In this section, we analyze whether dividing a liquid with any number of neurons (N_{tot}) would result in similar accuracy

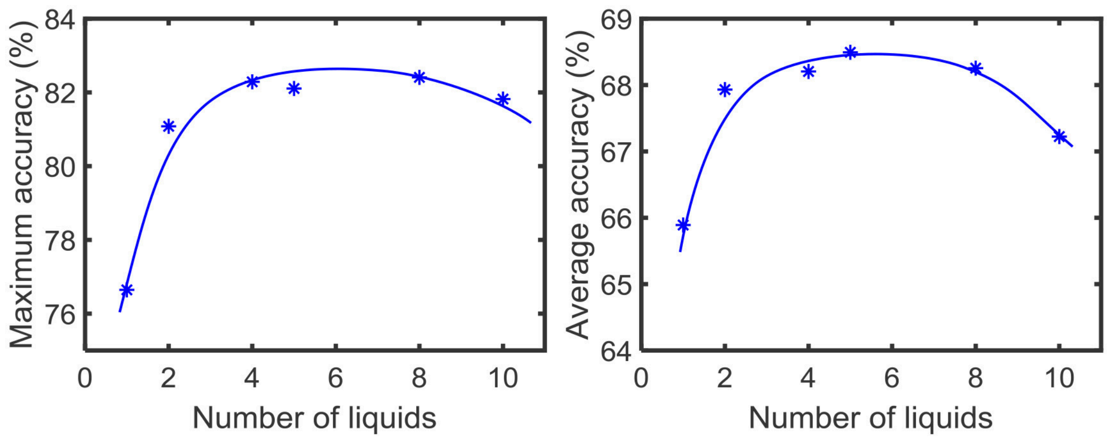


FIGURE 17 | (A) The maximum accuracy among all the LSM configurations with different $P_{IN\rightarrow E}$ and $P_{E\rightarrow E}$ **(B)** The average accuracy across all the LSM configurations with different $P_{IN\rightarrow E}$ and $P_{E\rightarrow E}$.

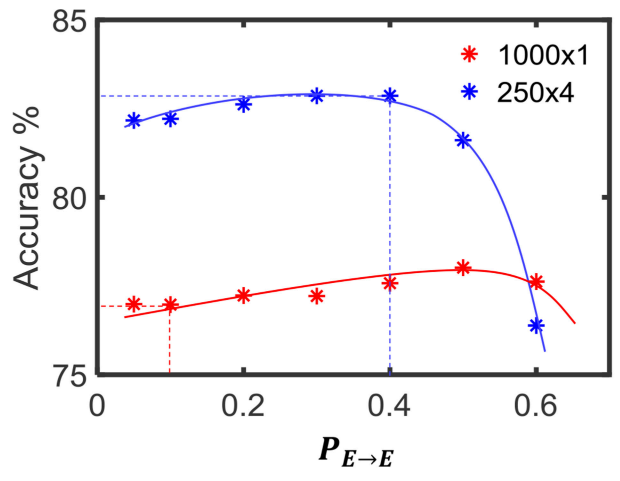


FIGURE 18 | The accuracy variation of the single liquid LSM baseline and ensemble approach ($N_{ens} = 4$) at different percentage connectivity values ($P_{E\rightarrow E}$). The accuracy of the ensemble approach at $P_{E\rightarrow E} = 0.4$ is higher than the accuracy of the single liquid LSM at $P_{E\rightarrow E} = 0.1$. Note that the number of connections in both the cases considered are the same. The accuracy was evaluated on the TI46-alpha classification task.

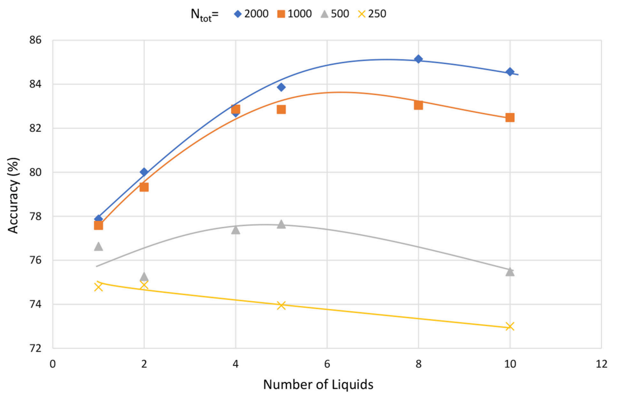


FIGURE 19 | The accuracy varying with the number of liquids in the ensemble approach, for different total number of neurons (N_{tot}). The LSM classifies speech data in the TI-alpha dataset.

improvements. To this effect, we created ensembles of liquids with different total number of neurons (N_{tot}). As **Figure 19** illustrates, liquids with large number of neurons show clear sign of accuracy improvement when divided into smaller liquids. However, when the number of neurons is smaller, dividing the liquid may result in decreased accuracy. For example, note that the accuracy reduces continuously when a liquid with 250 neurons is divided. This result is similar to the observation in Srinivasan et al. (2018), where the authors have shown that the input and liquid subdivision is beneficial for LSMs with large number of neurons. Similarly, here the ensemble approach makes sense only for LSMs with large number of neurons in them. In conclusion, we state the following with respect to the applicability of the ensemble approach for LSMs. In order to improve the accuracy of an LSM, the number of neurons can be increased. However, beyond a certain point, accuracy does not improve further. In such a case, the ensemble approach can be utilized to further increase the accuracy. Such accuracy improvements are not attainable by means of other simple methods that preserve the structural and training simplicity of the standard LSM, such as changing the connectivity.

3.6. Multiple Liquid-Multiple Readouts (MLMR) Approach

When moving from the single liquid approach to the ensemble of liquids approach, any benefit in terms of classifier training time was not observed. This is due to the fact that the number of total liquid neurons is the same, and we are using a single classifier. In this section, we analyze, if including a readout at the end of each small liquid is beneficial than having a single readout for all

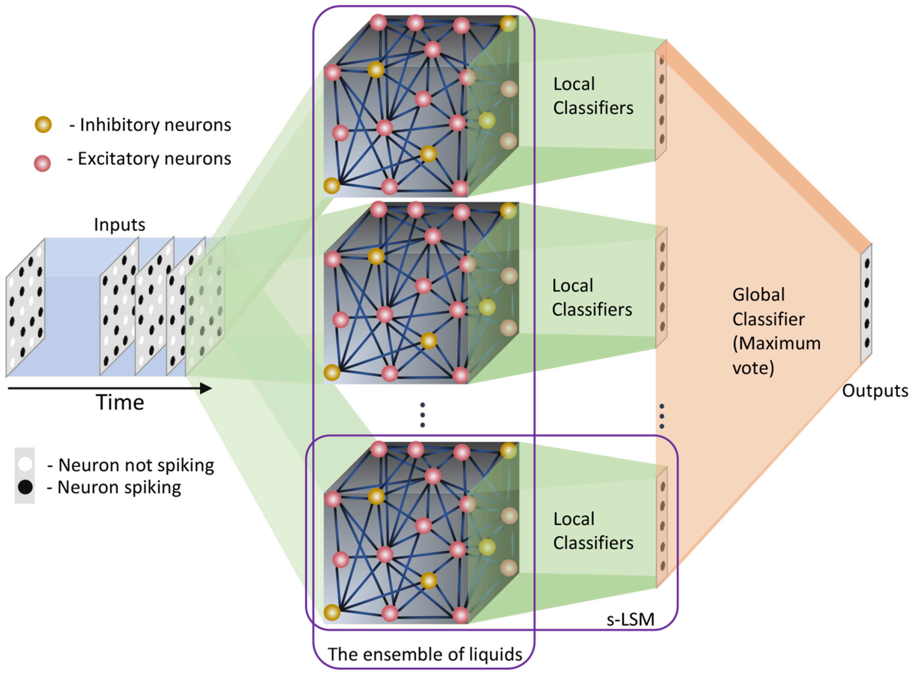


FIGURE 20 | The structure of the multiple liquid-multiple readout (MLMR) approach. There are multiple small liquids, and individual classifiers at the end of each liquid. These are defined as small LSMs or s-LSMs. Final outcome (global output) is calculated by considering the maximum vote among all the local classified outputs from the s-LSMs (local outputs).

the liquids. The basic structure of this multiple liquid-multiple readouts (MLMR) approach is shown in **Figure 20**.

In contrast to our previous approach, this structure could be viewed as a collection of small LSMs (s-LSMs). Each s-LSM is trained individually, and the final classification is done by considering either the maximum outcome, or the majority vote among all the local classifiers. During training, we do not use all the training data points for each local s-LSM classifier. Instead, we divide the training space among the ensemble of s-LSMs based on the following two criteria:

1. Random training space division (RD)
2. Clustered training space division (CD)

In random training space division (RD) method, we randomly divide the training data space among the ensemble of s-LSMs, and feed them to obtain the corresponding liquid state vectors at the output of each liquid. These state vectors were then used to train the local classifiers attached to each s-LSM in the ensemble using gradient descent error backpropagation. For example, if there are N_{ens} number of s-LSMs and N_{train} number of examples in the training set, each s-LSM will be trained with $\frac{N_{train}}{N_{ens}}$ number of randomly picked training examples. On the other hand, in the clustered input space division (CD) method, we divide the training instances into certain clusters depending upon their features, (for instance, we have selected "original," "rotated," "shifted," and "noisy" images from the extended MNIST dataset as clusters) and used them to train each readout. Here, an s-LSM has specific knowledge about the cluster of examples that it is trained with, and zero knowledge about other clusters. Therefore, an s-LSM may not correctly identify an input that belongs to a different cluster, apart from what it was trained with, leading to large accuracy degradation at the global classifier. For example, if a rotated image of digit "1" is given as the input, the s-LSM that was trained with rotated images will correctly recognize the given image. i.e., the output neuron−1 gives the highest outcome (there are 10 output neurons and they are indexed as neuron−0 through neuron−9 as shown in **Figure 21**). Other s-LSMs may not recognize this input correctly, potentially leading to another neuron apart from neuron−1 to give a high value at the outputs of their corresponding classifiers. When getting the final outcome using "maximum output" method, the neuron that gives the highest value over all the s-LSMs may not be neuron−1. Instead, it could be some different neuron from an s-LSM that was not trained with rotated images. To address this issue, we use an "inhibition" criterion to suppress the s-LSMs from giving high outputs for cluster types that they are not trained with. Initially we divide the training space into clusters along with their standard target vectors (vectors of which the length is equal to the number of classes L. If the input belongs to the ith class, then the ith element in the vector will be "1" and the other elements will be "0." Refer to **Figure 21**). Then, we randomly select 10% of the training instances from each cluster (foreign instances), and add them to the training space of all other clusters. The target vectors of the foreign instances are forced to have all their elements equal to $1/L$ (L is the number of classes) and we name this target vector as "inhibitory label vector". This will force each s-LSM outcome

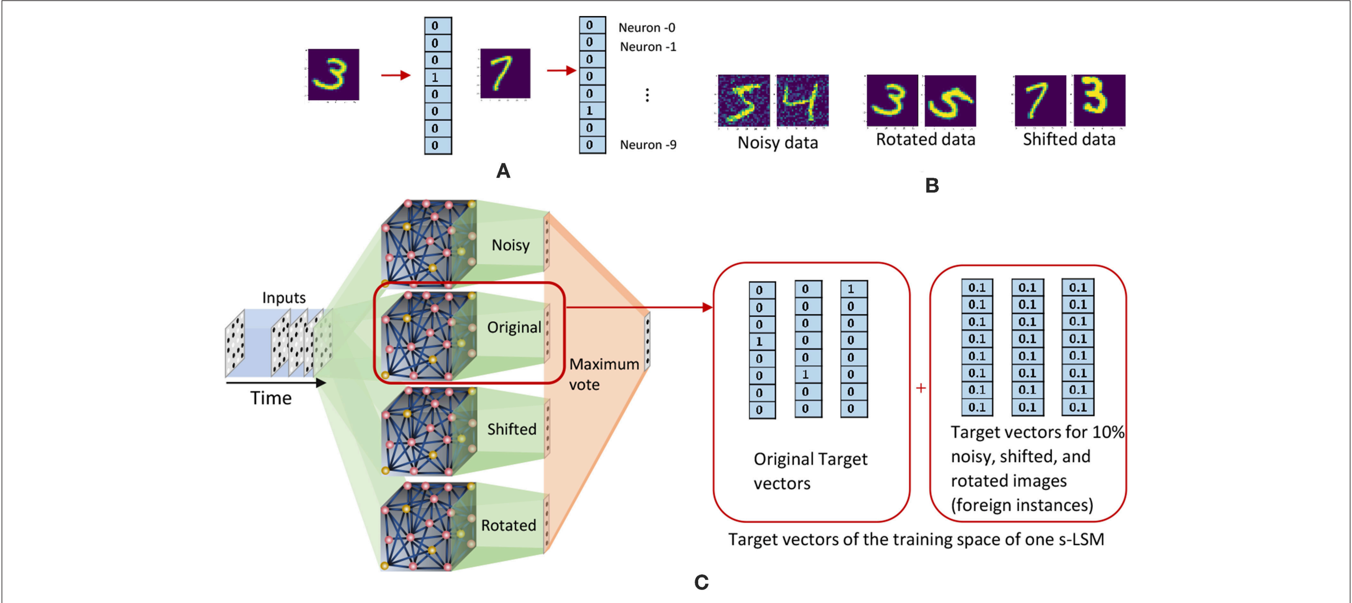


FIGURE 21 | (A) Target vectors that correspond to images in the extended - MNIST data set. **(B)** Examples from the clustered training data space of the extended MNIST dataset. **(C)** The clustered training space division method. Each s-LSM is trained with a particular cluster of images, and an additional 10% of the images in the other clusters (foreign data). The target vectors of the foreign data are modified to have each value equal to 0.1.

TABLE 3 | Accuracy of different ensemble approaches.

Approach (total number of neurons= 1, 000)	Accuracy (%)
Single liquid, single readout (baseline)	86.9
4 ensembles, single readout (MLSR)	89.0
4 ensembles, 4 readouts, (MLMR) random training space division (RD)	82.5
4 ensembles, 4 readouts, (MLMR) clustered training space division (CD)	83.1

to be low, when the presented input does not belong to the cluster with which the s-LSM was trained. This method is explained graphically in **Figure 21**, by means of an example.

We used the handwritten digit recognition application with the extended MNIST data set, to check the accuracy, performance, and training time of the aforementioned methods. The training data set was divided into 4 clusters; original MNIST images, noisy images, rotated images and shifted images. Total number of neurons were 1, 000 and each s-LSM has 250 neurons. The connectivity is set as indicated in **Table 1**. **Table 3** reports the accuracy of the above explained two training space division methods (RD and CD) along with the accuracy of the baseline (single liquid with 1, 000 neurons). The accuracy of the two methods (RD $\rightarrow$ 82.5% and CD $\rightarrow$ 83.1%) are inferior to that of the baseline (86.9%).

When comparing with RD method, CD gives better accuracy for the same number of neurons. The reason for this can be explained as follows. The clusters in the training dataset can have different overlapping/non-overlapping distributions. For instance, three clusters ("noisy," "shifted," and "rotated") in the considered example in this work follow three different distributions as shown in **Figure 22A**. The figure elaborates the t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008) of the high dimensional images that belong to the aforementioned three clusters, for better visualization in the lower dimensional space (2D). Due to this separate distributions, examples that belong to the same class but in different clusters may not spatially stay together in the higher dimensional space. For example, **Figure 22B** shows the data points that correspond to digit "0" and digit "1" in different clusters, and neither the data points of digit "0" nor "1" stay together. Let us consider the RD method, and how it tries to classify the aforementioned digit "0" and digit "1." If there are N_{tot} amount of training examples and L classes, the number of examples that belongs to class i each classifier sees is $N_{ex,i} = \frac{N_{tot}}{L \times N_{ens}}$. The $N_{ex,i}$ number of examples a classifier in RD method sees belongs to N_{ens} number of clusters and they are distributed all over as shown in **Figure 22B**. According to the figure, the two classes are not linearly separable. Therefore, the RD method leads to more misclassifications as elaborated in **Figure 22A**. In contrast, a classifier trained for "shifted" data cluster in CD method fits to a decision boundary that classifies digit "0" and digit "1" that *onlybelongs* to "shifted" data cluster. Owing to the proposed inhibition criterion, the classifier trained for "shifted" examples in CD tries to put the data points that belong to other clusters into a single category. As the **Figure 22B** illustrates, the classes: "digit 0," "digit 1," and "foreign" are more linearly separable by CD method than the RD method, and this leads to higher accuracy in RD method.

Selecting more foreign examples would result in the classifier to concentrate more on fitting the foreign data into "nhibitory label vectors," instead of classifying data in the corresponding

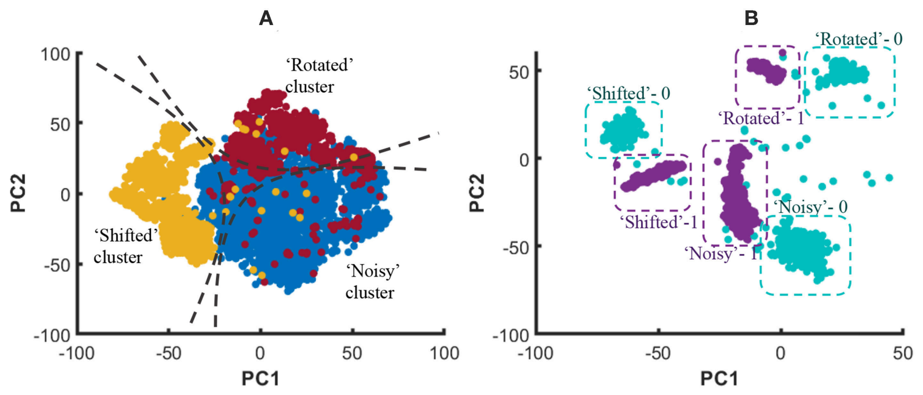


FIGURE 22 | (A) The t-Distributed Stochastic Neighbor Embedding (t-SNE) of the high dimensional input data points, in 2D space for better visualization. The distribution of data points that belong to three clusters ("Noisy," "Shifted," and "Rotated") stay spatially separated. **(B)** The distribution of data points of digit "1" and digit "0" that belongs to three clusters.

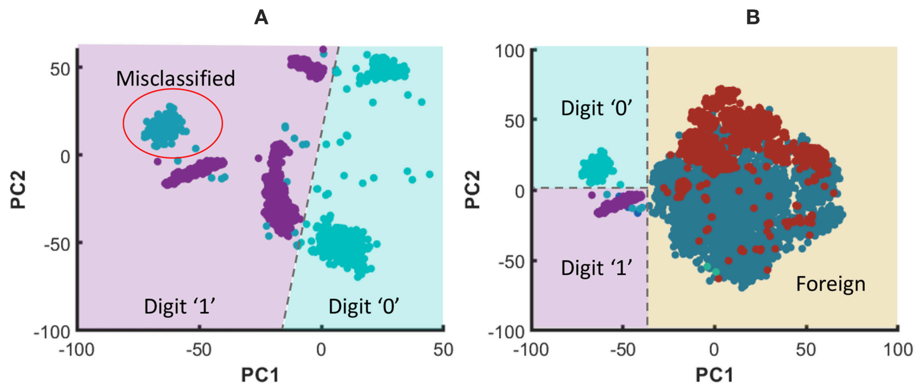


FIGURE 23 | (A) The graphical representation of the RD method trying to classify digit "0" and digit "1" that belong to three clusters using a linear classifier. Note that the digit "0" that belong to the shifted cluster is misclassified as digit "1". **(B)** The graphical representation of the CD method trying to classify digit "0" and digit "1". The particular classifier shown has learned to correctly classify digit "0" and digit "1" that belong to "shifted" cluster. Furthermore, it recognizes the data points that belong to foreign clusters due to the proposed inhibition criterion. The dashed lines show the classifier decision boundaries.

cluster. Consider adding an $x_f\%$ of foreign instances per cluster. This would result in adding $\frac{(N_{ens}-1)x_f\%}{(N_{ens}-1)x_f\%+1}$ overall percentage of extra data that does not belong to the cluster to which the classifier must be trained. The training space of the classifier that must be trained for "shifted" images consists of the following "sets": (1) shifted digit "0," (2) shifted digit "1,"..., 10)shifted digit "9," randomly selected images from (11) "noisy" cluster, (12) "original" cluster, (13) "rotated" cluster. We selected a percentage, that will pick approximately equal number of data points from each of the aforementioned 13 sets. In our particular example, to make $\sim 7.7\%(= 100/13)$ of the training space to be attached to each of the above "sets", $x_f\%$ needs to be selected as 10%. The total percentage of the foreign instances are 23% $\left(= \frac{(N_{ens}-1)x_f\%}{(N_{ens}-1)x_f\%+1}\right)$ per cluster.

In order to see if there is any benefit in the MLMR approach when achieving a "given" accuracy, we reduced N_{tot} in the baseline to match the accuracy of both the RD and CD methods. The memory requirement, inference time, and training time were calculated for two scenarios. First, N_{tot} in the baseline was selected such that both the baseline and the RD method have the same accuracy (82.5%). Second, N_{tot} in the baseline was selected such that it matches the accuracy of the CD method (83.1%). In each of the above scenarios, the obtained memory requirement, inference time, and training time values were normalized with respect to the baseline. These normalized values for the two cases are shown in a single graph in **Figure 24**. The CD method is better in terms of memory requirement and inference time, in comparison to the single liquid baseline and RD method. We calculated the total number of MAC (multiply and accumulate) operations during training to estimate the training time (it is a function of the number of neurons in a liquid, number of output neurons, and number of training examples). Lowest training time was achieved in the RD method. The CD method offers 56% reduction in memory and 45% reduction in inference time, with respect to the baseline. For a $1,000$ total number of neurons, the

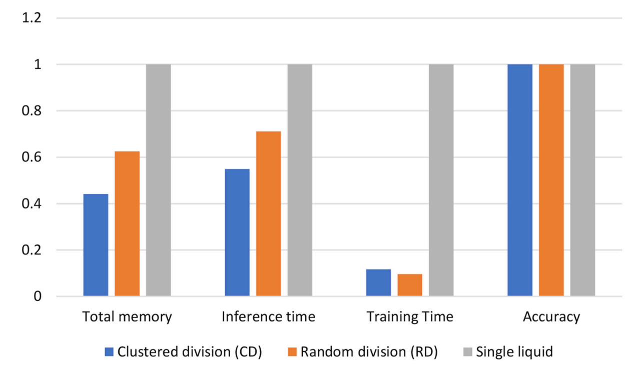


FIGURE 24 | Normalized total memory requirement, inference time, and training time of the clustered training space division method (CD), random training space division method (RD), and the single liquid baseline. The results are under iso-accuracy conditions.

4 ensemble case with a single classifier (studied in section 3.2. Let us denote this method as multiple liquids, single readout or MLSR approach) resulted in 78% reduction in memory usage and 72% reduction in inference time along with 2.1% accuracy improvement (hence better than both CD and RD methods under the memory usage and inference time metrics). However, in terms of training time, the MLSR approach did not show any improvement, whereas the MLMR showed 88% reduction with respect to the baseline.

4. CONCLUSION

We have presented an ensemble approach for Liquid State Machines (LSMs) that enhances separation and approximation properties, leading to accuracy improvements. The separation property in LSMs measures the dispersion between projected liquid states from different classes, whereas the approximation property indicates the concentration of the liquid states that belong to the same class. The ratio between SP and AP (*DR*) is a measure of the class discrimination. We witnessed that the *DR* increases when a large liquid is divided into multiple smaller independent liquids across four speech and image recognition tasks. We observed the existence of an optimal number of liquids ($N_{ens,opt}$) until which the *DR* increases and saturates thereafter. Owing to the improvement in the *DR* in our proposed ensemble approach, we noticed an LSM accuracy enhancement with increasing number of liquids. The accuracy peaked at the same $N_{ens,opt}$ point at which each *DR* saturated, for different recognition tasks. This validated the existence of an optimal number of liquids which gives the best accuracy for the LSM, and this point is highly dependent upon the application and the total number of liquid neurons.

There is plethora of complex approaches that concentrate on improving the accuracy of LSMs, including learning the liquid connections (Wang and Li, 2016; Xue et al., 2016). In contrast to such works, our proposed approach does not change the simple structure and training methods of LSMs. Furthermore, the ensemble approach gives better accuracy when compared with other simple mechanisms of improving the LSM accuracy such as increasing the number of neurons, changing the percentage connectivity, and utilizing the probabilistic local connectivity models. Apart from providing improved accuracy, the proposed ensemble approach comes with other benefits including lower memory requirement and lower inference time. We have shown that creating an ensemble of liquids leads to lower inter-connections in comparison to a single liquid with the same number of neurons. Furthermore, the liquid evaluation can potentially be parallelized in the ensemble approach due to the existence of small independent liquids. This results in reduced LSM inference time. The accuracy improvement with increasing number of liquids in the ensemble becomes less evident when the total number of neurons is small. In fact, creating an ensemble of liquids with a small number of neurons will rather reduce the accuracy. Hence the ensemble approach makes sense for LSMs with large number of neurons (Srinivasan et al., 2018).

Since there is no benefit in terms of training time between a single-liquid LSM and the proposed ensemble approach (MLSR), we investigated the MLMR approach where a classifier is added to each small liquid in the ensemble. By dividing the training example space to train each small LSM, we were able to attain significant benefits in terms of training time, when compared with MLSR approach. There are multiple classifiers that were trained independently in the MLMR approach, and the final output is the maximum vote of all the local classifiers. The set of multiple liquid-classifier units are in fact a collection of small LSMs (noted as s-LSMs). Despite the performance benefits during training, we noticed an accuracy degradation in the MLMR approach, when compared with both the MLSR approach and the single-liquid baseline LSM with equal number of liquid neurons. The reason for this can be explained as follows. The classifiers in each s-LSM are smaller than that of the baseline and the MLSR approaches. A large classifier (as in the baseline and MLSR approach) has more number of parameters and is capable of fitting in to an unknown function better than a small classifier (Krzywinski and Altman, 2015), leading to improved accuracy.

DATA AVAILABILITY

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

AUTHOR CONTRIBUTIONS

PW performed the simulations. All the authors contributed in developing the concepts, generating experiments, and writing the manuscript.

ACKNOWLEDGMENTS

The authors would like to thank C. Liyanagedera of Purdue University for carefully reading the manuscript and giving feedback. This work was supported in part by the Center for Brain Inspired Computing (C-BRIC), one of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, by the Semiconductor Research Corporation, the National Science Foundation, Intel Corporation, the DoD Vannevar Bush Fellowship, and by the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-16-3-0001.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2019.00504/full#supplementary-material

REFERENCES

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* 50, 303–304. doi: 10.1016/S0361-9230(99)00161-6

Anastassiou, C. A., Perin, R., Markram, H., and Koch, C. (2011). Ephaptic coupling of cortical neurons. *Nat. Neurosci.* 14, 217. doi: 10.1038/nn.2727

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. *arXiv preprint arXiv:1803.09574*. Available online at: http://papers.nips.cc/paper/7359-long-short-term-memory-and-learning-to-learn-in-networks-of-spiking-neurons.pdf

Cruz-Albrecht, J. M., Yung, M. W., and Srinivasa, N. (2012). Energy-efficient neuron, synapse and stdp integrated circuits. *IEEE Trans. Biomed. Circ. Syst.* 6, 246–256. doi: 10.1109/TBCAS.2011.2174152

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Proc. Mag.* 29, 141–142. doi: 10.1109/MSP.2012.2211477

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9:99. doi: 10.3389/fncom.2015.00099

Du, C., Cai, F., Zidan, M. A., Ma, W., Lee, S. H., and Lu, W. D. (2017). Reservoir computing using dynamic memristors for temporal information processing. *Nat. Commun.* 8:2204. doi: 10.1038/s41467-017-02337-y

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugen.* 7, 179–188. doi: 10.1111/j.1469-1809.1936.tb02137.x

Fukunaga, K., and Mantock, J. M. (1983). Nonparametric discriminant analysis. *IEEE Trans. Pattern Anal. Mach. Intel.* 6, 671–678. doi: 10.1109/TPAMI.1983.4767461

Goodman, D. F., and Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Front. Neuroinf.* 2:5. doi: 10.3389/neuro.11.005.2008

Goodman, E. and Ventura, D. (2006). "Spatiotemporal pattern recognition via liquid state machines," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on* (Vancouver, BC: IEEE), 3848–3853.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Grzyb, B. J., Chinellato, E., Wojcik, G. M., and Kaminski, W. A. (2009). "Facial expression recognition based on liquid state machines built of alternative neuron models," in *2009 International Joint Conference on Neural Networks* (Atlanta, GA).

Herlihy, M., and Nir, S. (2011). *The Art of Multiprocessor Programming*. Burlington, MA: Morgan Kaufmann.

Hourdakis, E., and Trahanias, P. (2013). Use of the separation property to derive liquid state machines with enhanced classification performance. *Neurocomputing* 107, 40–48. doi: 10.1016/j.neucom.2012.07.032

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Comput.* 3, 79–87. doi: 10.1162/neco.1991.3.1.79

Jaeger, H. (2007). Echo state network. *Scholarpedia* 2:2330. doi: 10.4249/scholarpedia.2330

Ji, S., and Ye, J. (2008). Generalized linear discriminant analysis: a unified framework and efficient model selection. *IEEE Trans. Neural Netw.* 19, 1768–1782. doi: 10.1109/TNN.2008.2002078

Jones, B., Stekel, D., Rowe, J., and Fernando, C. (2007). "Is there a liquid state machine in the bacterium escherichia coli?" in *Artificial Life, 2007. ALIFE'07. IEEE Symposium on* (Honolulu, HI: IEEE), 187–191.

Kaiser, J., Stal, R., Subramoney, A., Roennau, A., and Dillmann, R. (2017). Scaling up liquid state machines to predict over address events from dynamic vision sensors. *Bioinspiration Biomimetics* 12, 055001. doi: 10.1088/1748-3190/aa7663

Kasap, B., and van Opstal, A. J. (2018). Dynamic parallelism for synaptic updating in gpu-accelerated spiking neural network simulations. *Neurocomputing* 302, 55–65. doi: 10.1016/j.neucom.2018.04.007

Kötter, R. (2012). *Neuroscience Databases: A Practical Guide*. New York, NY: Springer Science & Business Media.

Krzywinski, M., and Altman, N. (2015). Points of significance: multiple linear regression. *Nat. Methods* 12, 1103–1104. doi: 10.1038/nmeth.3665

Liberman, M., Amsler, R., Church, K., Fox, E., Hafner, C., Klavans, J., et al. (1993). *Ti 46-Word*. Philadelphia, (PA): Linguistic Data Consortium.

Liyanagedera, C. M., Wijesinghe, P., Jaiswal, A., and Roy, K. (2017). "Image segmentation with stochastic magnetic tunnel junctions and spiking neurons," in *2017 International Joint Conference on Neural Networks (IJCNN)* (Anchorage, AK: IEEE), 2460–2468.

Lyon, R. (1982). "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.* Vol. 7, (Paris: IEEE), 1282–1285.

Maass, W., Legenstein, R. A., and Bertschinger, N. (2005). "Methods for estimating the computational power and generalization capability of neural microcircuits," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 865–872.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

Maass, W., Natschläger, T., and Markram, H. (2003). "A model for real-time computation in generic neural microcircuits," in *Advances in Neural Information Processing Systems* (Cambridge, MA), 229–236.

Maass, W., Natschläger, T., and Markram, H. (2004). Computational models for generic cortical microcircuits. *Comput. Neurosci.* 18:575. doi: 10.1201/9780203494462.ch18

Maaten, L. v. d., and Hinton, G. (2008). Visualizing data using t-sne. *J. Mach. Learn. Res.* 9, 2579–2605. Available online at: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

Mei, S., Montanari, A., and Nguyen, P.-M. (2018). A mean field view of the landscape of two-layers neural networks (Washington, DC). *arXiv preprint arXiv:1804.06561*. doi: 10.1073/pnas.1806579115

Neftci, E. O., Pedroni, B. U., Joshi, S., Al-Shedivat, M., and Cauwenberghs, G. (2016). Stochastic synapses enable efficient brain-inspired learning machines. *Front. Neurosci.* 10:241. doi: 10.3389/fnins.2016.00241

Nikolić, D., Häusler, S., Singer, W., and Maass, W. (2009). Distributed fading memory for stimulus properties in the primary visual cortex. *PLoS Biol.* 7:e1000260. doi: 10.1371/journal.pbio.1000260

Norton, D., and Ventura, D. (2006). "Preparing more effective liquid state machines using hebbian learning," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on* (Vancouver, BC: IEEE), 4243–4248.

Panda, P., and Roy, K. (2017). Learning to generate sequences with combination of hebbian and non-hebbian plasticity in recurrent spiking neural networks. *Front. Neurosci.* 11:693. doi: 10.3389/fnins.2017.00693

Panda, P., and Srinivasa, N. (2018). Learning to recognize actions from limited training examples using a recurrent spiking neural model. *Front. Neurosci.* 12:126. doi: 10.3389/fnins.2018.00126

Park, K. I., and Park (2018). *Fundamentals of Probability and Stochastic Processes With Applications to Communications*. Gewerbestrasse, Cham: Springer.

Robbins, H., and Monro, S. (1985). "A stochastic approximation method," in *Herbert Robbins Selected Papers* (Ann Arbor, MI: Springer), 102–109.

Roy, S., and Basu, A. (2016). An online structural plasticity rule for generating better reservoirs. *Neural Comput.* 28, 2557–2584. doi: 10.1162/NECO_a_00886

Roy, S., and Basu, A. (2017). An online unsupervised structural plasticity algorithm for spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 900–910. doi: 10.1109/TNNLS.2016.2582517

Roy, S., Basu, A., and Hussain, S. (2013). "Hardware efficient, neuromorphic dendritically enhanced readout for liquid state machines," in *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (Rotterdam: IEEE), 302–305.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* 323, 533. doi: 10.1038/323533a0

Shim, Y., Philippides, A., Staras, K., and Husbands, P. (2016). Unsupervised learning in an ensemble of spiking neural networks mediated by itdp. *PLoS Comput. Biol.* 12:e1005137. doi: 10.1371/journal.pcbi.1005137

Slaney, M. (1998). *Auditory Toolbox*. Technical Repor, Interval Research Corporation, Vol. 10, 1998.

Srinivasan, G., Panda, P., and Roy, K. (2018). Spilinc: spiking liquid-ensemble computing for unsupervised speech and image recognition. *Front. Neurosci.* 12:524. doi: 10.3389/fnins.2018.00524

Toledo-Suárez, C., Duarte, R., and Morrison, A. (2014). Liquid computing on and off the edge of chaos with a striatal microcircuit. *Front. Comput. Neurosci.* 8:130. doi: 10.3389/fncom.2014.00130

Urbain, G., Degrave, J., Carette, B., Dambre, J., and Wyffels, F. (2017). Morphological properties of mass–spring networks for optimal locomotion learning. *Front. Neurorob.* 11:16. doi: 10.3389/fnbot.2017.00016

Verstraeten, D., Schrauwen, B., and Stroobandt, D. (2005). Isolated word recognition using a liquid state machine. *Inf. Proc. Lett.* 95, 521–528. doi: 10.1016/j.ipl.2005.05.019

Waldrop, M. M. (2012). Brain in a box: Henry markram wants 1 billion [euro] to model the entire human brain. sceptics don't think he should get it. *Nature* 482, 456–459. doi: 10.1038/482456a

Wang, Q., Jin, Y., and Li, P. (2015). "General-purpose lsm learning processor architecture and theoretically guided design space exploration," in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (Atlanta, GA: IEEE), 1–4.

Wang, Q., Kim, Y., and Li, P. (2014). "Architectural design exploration for neuromorphic processors with memristive synapses," in *14th IEEE International Conference on Nanotechnology* (Toronto, ON: IEEE), 962–966.

Wang, Q., and Li, P. (2016). "D-lsm: Deep liquid state machine with unsupervised recurrent reservoir tuning," in *Pattern Recognition (ICPR), 2016 23rd International Conference on* (Cancun: IEEE), 2652–2657.

Wang, Q., Li, Y., Shao, B., Dey, S., and Li, P. (2017). Energy efficient parallel neuromorphic architectures with approximate arithmetic on fpga. *Neurocomputing* 221, 146–158. doi: 10.1016/j.neucom.2016.09.071

Wehr, M., and Zador, A. M. (2003). Balanced inhibition underlies tuning and sharpens spike timing in auditory cortex. *Nature* 426, 442. doi: 10.1038/nature02116

Wijesinghe, P., Ankit, A., Sengupta, A., and Roy, K. (2018). An all-memristor deep spiking neural computing system: a step toward realizing the low-power stochastic brain. *IEEE Trans. Emerg. Top. Comput. Intel.* 2, 345–358. doi: 10.1109/TETCI.2018.2829924

Wijesinghe, P., Liyanagedera, C. M., and Roy, K. (2017). "Fast, low power evaluation of elementary functions using radial basis function networks," in *Proceedings of the Conference on Design, Automation & Test in Europe* (Lausanne: European Design and Automation Association), 208–213.

Wu, J., Chua, Y., Zhang, M., Li, H., and Tan, K. C. (2018). A spiking neural network framework for robust sound classification. *Front. Neurosci.* 12:836. doi: 10.3389/fnins.2018.00836

Xie, Z. (2017). Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*.

Xue, F., Guan, H., and Li, X. (2016). "Improving liquid state machine with hybrid plasticity," in *Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016 IEEE* (Xi'an: IEEE), 1955–1959.

Yaniv, L., and Yossi, M. (2018). Google duplex: an ai system for accomplishing real-world tasks over the phone. *Google AI Blog* (2018).

Yao, W., Zeng, Z., Lian, C., and Tang, H. (2013). "Ensembles of echo state networks for time series prediction," in *Advanced Computational Intelligence (ICACI), 2013 Sixth International Conference on* (Hangzhou: IEEE), 299–304.

Zhang, W., and Li, P. (2019). Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks. *Front. Neurosci.* 13:31. doi: 10.3389/fnins.2019.00031

Zhang, Y., Li, P., Jin, Y., and Choe, Y. (2015). A digital liquid state machine with biologically inspired learning and its application to speech recognition. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 2635–2649. doi: 10.1109/TNNLS.2015.2388544

Conflict of Interest Statement: 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.

ORIGINAL RESEARCH
published: 21 August 2019
doi: 10.3389/fnins.2019.00827

A Theory for Sparse Event-Based Closed Loop Control

Pierre Daye[1]*, *Sio-Hoi Ieng*[2,3,4] and *Ryad Benosman*[2,3,4,5,6]

[1] StreetLab - Institut de la Vision, Paris, France, [2] INSERM UMRI S 968, Institut de la Vision, Paris, France, [3] Sorbonne Universités, UPMC Univ Paris 06, UMR S 968, Institut de la Vision, Paris, France, [4] CNRS, UMR 7210, Institut de la Vision, Paris, France, [5] University of Pittsburgh Medical Center, Biomedical Science Tower 3, Pittsburgh, PA, United States, [6] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, United States

Most dynamic systems are controlled by discrete time controllers. One of the main challenges faced during the design of a digital control law is the selection of the appropriate sampling time. A small sampling time will increase the accuracy of the controlled output at the expense of heavy computations. In contrast, a large sampling time will decrease the computational power needed to update the control law at the expense of a smaller stability region. In addition, once the setpoint is reached, the controlled input is still updated, making the overall controlled system not energetically efficient. To be more efficient, one can update the control law based on a significant fixed change of the controlled signal (send-on-delta or event-based controller). Like for time-based discretization, the amplitude of the significant change must be chosen carefully to avoid oscillations around the setpoint (e.g., if the setpoint is in between two samples) or an unnecessary increase of the samples number needed to reach the setpoint with a given accuracy. This paper proposes a novel non-linear event-based discretization method based on inter-events duration. We demonstrate that our new method reaches an arbitrary accuracy independently of the setpoint amplitude without increasing the network data transmission bandwidth. The method decreases the overall number of samples needed to estimate the states of a dynamical system and the update rate of an actuator, making it more energetically efficient.

Keywords: dynamic systems, feedback control, control theory, event-based signal processing, level crossing sampling

OPEN ACCESS

Edited by:
Gert Cauwenberghs,
University of California, San Diego,
United States

Reviewed by:
Yulia Sandamirskaya,
Institute of Neuroinformatics, ETH
Zürich, Switzerland
Nabil Imam,
Cornell University, United States

****Correspondence:***
Pierre Daye
pierre.daye@gmail.com

Specialty section:
This article was submitted to
Neuromorphic Engineering,
a section of the journal
Frontiers in Neuroscience

Received: *15 February 2019*
Accepted: *24 July 2019*
Published: *21 August 2019*

Citation:
Daye P, Ieng S-H and Benosman R
(2019) A Theory for Sparse
Event-Based Closed Loop Control.
Front. Neurosci. 13:827.
doi: 10.3389/fnins.2019.00827

1. INTRODUCTION

With ever faster and ever cheaper digital computers, the control of dynamic systems has shifted from analog to digital controllers. Critically, the majority of discrete-time control laws assume that the sampling rate of the discretization process is constant. There is currently a discrete-time equivalent for the vast majority of continuous-time control theory principles, from the continuous proportional-integral mechanical "governors" of Maxwell (1867) to the more recent optimal control theories based on Pontryagin's maximum principle (Pontryagin et al., 1962).

The selection of the appropriate sampling rate depends both on the open-loop system dynamics and on the desired dynamics of the controlled system. A system with fast dynamics needs a high sampling rate to ensure the stability of the controlled system at the expense of higher computational power. Moreover, the controlled input of a dynamical system is traditionally updated at each time step independently of the error amplitude. When a controlled system is in a stable configuration at the setpoint, there is obviously no need to sample the data, update the controller and the actuator. Indeed, doing so is not energetically efficient.

In the early 60's, this lack of efficiency combined with lower computational power spurred the development of adaptive discrete-time sampling methods (Dorf et al., 1962; Tomovic and Bekey, 1966). In these methods, an event is sent once the sampled signal increases or decreases by a certain delta (send-on-delta/event-triggered discretization schemes). When the signal doesn't change, there are intrinsically no more updates. More than three decades later, there was a resurgent gain of interest for these discretization methods with aperiodic sampling time leading to new control mechanisms (e.g., Arzén, 1999; Bernhardsson and Aström, 1999; Heemels et al., 2012) for send-on-delta, (Miskowicz, 2005, 2007) for area/integral thresholds, and their analyses, e.g., the effect of noise on the send-on-delta mechanism (Astrom and Bernhardsson, 2002; Cervin and Astrom, 2007). In Hetel et al. (2017), a survey on stability studies of aperiodic sampling systems is provided.

Event-based control mechanisms have also drawbacks. For example, to the best of our knowledge, all current event-based control schemes transmit the signal value within an event. This limits the system dynamic range (ratio between the largest and the smallest value that a signal within a control system can assume) because the data transfer must have the same representation as the system data. Therefore, if one wants to control a system with a high accuracy on a wide range of value, such a control system would need a high bandwidth network to transmit the values of the signal.

In this paper, we propose a new framework and theory for event-based control that are circumventing the problem mentioned above while preserving the benefit of the event-based approach. We formalize a generalized discretization method that stems from the principle of neuromorphic event-based cameras (Posch et al., 2008, 2011) for analog signals. In contrast to traditional frame-based cameras where a clock synchronizes the acquisition of each pixel and the pixels' value is transmitted directly in an image, in event-based cameras each pixel is independent. When a pixel detects a light intensity change of a certain magnitude, it signals the change emitting an event. This event carries information about the time of the change, the position of the pixel and if the light intensity increased or decreased. The principal contribution of this work is to generalize the level-crossing sampling representation in the context of control. We focus on exploiting the benefit provided by a more efficient information coding to reduce computation resources through the use of the duration between two events rather than the value of a signal to update the control law. Therefore, an event can be represented with fewer bits than the input signal and this increases the dynamic range of the control system.

First, we describe a general class of event-based discretization functions and the associated reconstruction process in section 2.1. Then, in section 2.2 we show how uncertainties on the event timing as well as on the initial value of the signal affect the accuracy of the reconstructed signal. In section 2.3, we use a logarithm as discretization function and we show how we tackle the issue of the Zeno phenomenon (Heymann et al., 2005; Lampersky and Ames, 2013). Section 3 presents control results based on the logarithmic discretization. Finally, section 4 concludes the paper.

2. MATERIALS AND METHODS

2.1. Non-linear Event-Based Discretization

Figure 1 presents the general principle of a linear (**Figure 1A**) and a non-linear (**Figure 1B**) event discretization applied to an error signal $\epsilon(t)$. Using linear event-based discretization, if the setpoint lies between two intervals then no event will be generated (no event will be generated after t_4 as the system is stable between two samples). To overcome this problem, the simplest method is to compute the step size such that the setpoint is an integer multiple of the step size. It is worth noting that when the setpoint lies between two samples, it corresponds to a bias β in the linear function presented in **Figure 1A**. In the non-linear case presented in **Figure 1B**, the step size needed to generate events decreases as the controlled signal approaches the setpoint. Therefore, one can stop the event generation when the absolute error is smaller than a predefined threshold.

2.1.1. Event Generator Function

Given a generic finite dimension, non-linear continuous-time system S to control, we assume that a control law C has been designed to ensure that the output y of the system converge toward a setpoint r, stabilizing (at least locally) the controlled system. Both signals r and y can be multidimensional. Using this representation, the error signal ϵ is equal to the difference between the system output y and the setpoint r. **Figure 2** is a schematic summarizing this control configuration.Then, $\mathbb{B}$, a subset of $\mathbb{R}^+$ computed from the control law C, can be defined as the basin of attraction around the equilibrium $\epsilon = 0$ such that:

$$\forall \epsilon \in \mathbb{B} \subset \mathbb{R}^*, \lim_{t \to +\infty} \epsilon(t) = 0, \tag{1}$$

$$\lim_{t \to +\infty} \frac{d\epsilon(t)}{dt} = 0. \tag{2}$$

Let h be the "event generator" continuous function that computes the time of the next event t_{i+1} from the value of ϵ at t_i.

$$\begin{aligned} h : \mathbb{B} &\to \mathbb{R}^+ \\ \epsilon(t_i) &\mapsto h(\epsilon(t_i)) = t_{i+1} \end{aligned} \tag{3}$$

The key principle of our event-based discretization method is that the duration between two events generated by h must tend to zero when the error tends to zero. Mathematically, this can be written has a limit:

$$\lim_{t \to +\infty} [(h \circ \epsilon)(t) - t] = 0. \tag{4}$$

Equation (4) can be equivalently rewritten into:

$$\lim_{t \to +\infty} (h \circ \epsilon)(t) = \lim_{\epsilon \to 0} h(\epsilon) = \lim_{t \to +\infty} t \tag{5}$$

$$\Rightarrow \lim_{\epsilon \to 0} h(\epsilon) = \infty. \tag{6}$$

Equation (6) is the first requirement that must be met by h if (4) is true.

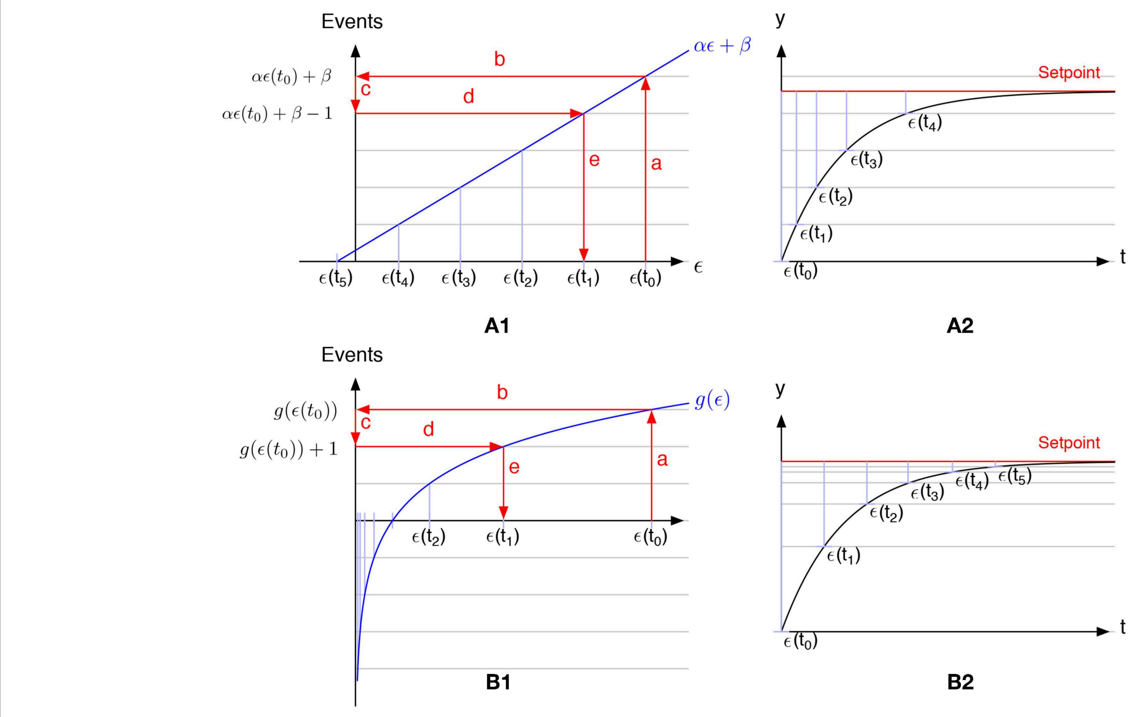


FIGURE 1 | Comparison of linear **(A1)** vs. non-linear **(B1)** event-based discretization. **(A2,B2)** Represent the time course of the controlled system (black lines) and the setpoint (red lines) for the linear and the non-linear discretization. Gray lines represents the levels at which an event will be triggered. Starting from an initial condition $\epsilon(t_0)$ (step a), one can evaluate $g(\epsilon(t_0))$ (step b). Then remove 1 from the value (step c) to evaluate when the next event will be generated $g(\epsilon(t_1)) = g(\epsilon(t_0)) - 1$. Finally, compute the inverse of g for this value and extract the error $\epsilon(t_1)$ at the next event (steps d and e). These steps can be repeated to extract $\epsilon(t_2)$ from $\epsilon(t_1)$, $\epsilon(t_3)$ from $\epsilon(t_2)$ and so on. As clearly shown in the figure, the limit of this recursive process when the error approaches zero is zero in the non-linear discretization but not in the linear one (because of the bias β in the function).

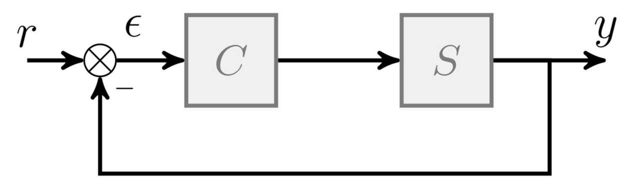


FIGURE 2 | Generic closed loop control system. r represents the setpoint, y the system output, ϵ the error between the setpoint and the system output, C the controller, and S the controlled system.

2.1.2. Events Generation

As the class of functions that satisfy (4) is very broad, this section presents a method to build these functions. One can compute a time series $\{t_i\}$ using a monotonically increasing function g applied to $\epsilon(t)$ such that an event will be triggered when the difference of g between two events is equal to one:

$$\begin{aligned}&\forall i \in \mathbb{N}^+\\&g : \mathbb{R}^+ \to \mathbb{R}\\&\left\{t_i \middle| \; g(\epsilon(t_{i-1})) - g(\epsilon(t_i)) = 1\right\}.\end{aligned} \quad (7)$$

For the rest of the paper, we postulate that g is monotonically increasing. However, it must be noted that if g is monotonically decreasing, the same reasoning can be applied and the time series becomes:

$$\left\{t_i \middle| \; g(\epsilon(t_i)) - g(\epsilon(t_{i-1})) = 1\right\}. \quad (8)$$

As g^{-1} exists (it is monotonically increasing), one can predict the value of the error at which the next event will be triggered:

$$\epsilon(t_i) = g^{-1}\left((g \circ \epsilon)(t_{i-1}) - 1\right). \quad (9)$$

This discretization process is presented in **Figure 1**.

In addition, ϵ^{-1} exists by definition of Equation (9). Therefore, one can write the function h defined in (4) as:

$$\begin{aligned}t_i &= \epsilon^{-1}(g^{-1}\left((g \circ \epsilon)(t_{i-1}) - 1)\right) && (10)\\&= h \circ \epsilon(t_{i-1}). && (11)\end{aligned}$$

From this relationship, a supplementary condition on g can be extracted from (6):

$$\lim_{t\to+\infty}(g \circ \epsilon)(t) = \lim_{t\to+\infty}(g \circ \epsilon)(t) - 1. \quad (12)$$

As g is a function of reals, only $\pm\infty$ is a solution of (12).

2.1.3. Signal Reconstruction

Using the proposed discretization, one can reconstruct the time course of the original error signal using the events:

$$\epsilon(t_N) = \epsilon(t_0) + \sum_{i=1}^{N}[g^{-1}\left((g\circ\epsilon)(t_{i-1}) - 1\right) - \epsilon(t_{i-1})]H(t - t_i) \quad (13)$$

in which $H(x)$ represents the Heaviside step function:

$$H(x) = \frac{d}{dx}max(x, 0). \quad (14)$$

As the series (13) converges toward 0, then:

$$\lim_{N\to+\infty}\sum_{i=1}^{N}[g^{-1}\left((g\circ\epsilon)(t_{i-1}) - 1\right) - \epsilon(t_{i-1})] = -\epsilon(t_0). \quad (15)$$

2.2. Uncertainty Analysis

Up to now, our formulation of the events generation assumes that there is no measurement uncertainties on both the initial value of the error ($\epsilon(t_0)$) and on the time interval estimation ($t_i - t_{i-1}$). In this paragraph, the effect of uncertainties on these values is analyzed.

2.2.1. Uncertainty on the Initial Value

First, we will analyze how an uncertainty on the initial estimate of the error, $\epsilon(t_0)$, influences the estimate of $\epsilon(t_N)$ and thus affects the convergence of the series. We will postulate that we have an infinitely accurate measurement of the time interval between two events.

Given an estimate of the initial error

$$\hat{\epsilon}(t_0) = \epsilon(t_0) + \delta_{\epsilon_0} \quad (16)$$

with $\epsilon(t_0)$ representing the true initial value, δ_{ϵ_0} representing an uncertainty on the true initial value, one can rewrite (13):

$$\begin{aligned}\hat{\epsilon}(t_N) &= \epsilon(t_0) + \delta_{\epsilon_0}\\ &+ \sum_{i=1}^{N}[g^{-1}\left(g(\epsilon(t_{i-1}) + \delta_{\epsilon_{i-1}}) - 1\right)\\ &- \epsilon(t_{i-1}) - \delta_{\epsilon_{i-1}}]H(t - t_i).\end{aligned} \quad (17)$$

The first terms of recursion (17) are:

$$\hat{\epsilon}(t_0) = \epsilon(t_0) + \delta_{\epsilon_0} \quad (18)$$

$$\begin{aligned}\hat{\epsilon}(t_1) &= \epsilon(t_1) + \delta_{\epsilon_1}\\ &= g^{-1}\left(g(\epsilon(t_0) + \delta_{\epsilon_0}) - 1\right)\end{aligned} \quad (19)$$

Because g is monotonically increasing, it follows that $\hat{\epsilon}(t_1) < \hat{\epsilon}(t_0)$. This leads to $g^{-1}\left(g(\epsilon(t_{i-1}) + \delta_{\epsilon_{i-1}}) - 1\right) < \epsilon(t_{i-1}) + \delta_{\epsilon_{i-1}}$ for arbitrary $i - 1$. Then for i, $g^{-1}\left(g(\epsilon(t_i) + \delta_{\epsilon_i}) - 1\right) < \epsilon(t_i) + \delta_{\epsilon_i}$ is also verified. Therefore, by induction:

$$\lim_{N\to\infty} g^{-1}\left(g(\epsilon(t_{i-1}) + \delta_{\epsilon_{i-1}}) - 1\right) = 0, \quad (20)$$

and

$$\lim_{N\to\infty}\hat{\epsilon}(t_N) = 0. \quad (21)$$

This result shows that, independently of the initial error on the measurement, the estimate of the error converges toward zero.

2.2.2. Uncertainties on Time Interval Measurement

In this section, we will analyze the effect of an uncertainty on the measurement of the time interval between two events. If one assumes that the uncertainty σ_i on the time value t_i is drawn from a uniform random distribution $\mathcal{U}_{-\varsigma}^{\varsigma}$ between $-\varsigma$ and ς, one can write Equation (13) as:

$$\begin{aligned}\epsilon(t) + \delta_\epsilon(t) &= \epsilon(t_0) + \sum_{i=1}^{N}[g^{-1}\left((g\circ\epsilon)(t_{i-1}) - 1\right)\\ &- \epsilon(t_{i-1})]H(t - t_i + \sigma_i).\end{aligned} \quad (22)$$

Using (13) and (22), one can extract $\delta_\epsilon(t)$, using the rectangular function Π:

$$\Pi(X) = H(X) - H(0) \quad (23)$$

$$\delta_\epsilon(t) = \sum_{i=1}^{N}[g^{-1}\left((g\circ\epsilon)(t_{i-1}) - 1\right) - \epsilon(t_{i-1})]\Pi(\sigma_i). \quad (24)$$

From (24), one can evaluate the bounds of the error if one supposes that all the uncertainties are equal to either ς or $-\varsigma$:

$$\delta_\epsilon^*(t) = \pm\varsigma\sum_{i=1}^{N}[g^{-1}\left((g\circ\epsilon)(t_{i-1}) - 1\right) - \epsilon(t_{i-1})]. \quad (25)$$

If $\epsilon(t_0) = \epsilon_0$, the limit of (25) representing the upper (lower) bound of $\delta_\epsilon(t)$ can be computed using (15):

$$\lim_{t\to+\infty}\delta_\epsilon^*(t) = \mp\varsigma\epsilon_0. \quad (26)$$

Similarly, the other bound is equal to $\varsigma\epsilon_0$. Equation (26) shows that the boundaries of the uncertainty on the error signal is only a function of the clock accuracy. Therefore, the proposed non-linear discretization can be used to reach an arbitrary precision of the controlled state, provided that the user has access to an infinitely accurate clock.

2.3. Logarithmic Event-Based Discretization

As the goal of the paper is to demonstrate the usefulness of the discretization method to control a system, we postulate in the following sections of the paper that the user has designed a control law such that the controlled system is globally asymptotically stable.

In the rest of the paper, we use a logarithmic function as event discretization function $g(\epsilon)$. We demonstrated in section 2.1 that a candidate discretization function g must fulfill three conditions:

1. $g : \mathbb{R}^{+,*} \to \mathbb{R}$

   $x \mapsto g(x)$
2. g must be continuous and strictly monotonic on $\mathbb{R}^{+,*}$ (and thus invertible)
3. $\lim_{x\to 0} g(x) = \begin{cases} -\infty & \text{if } dg/dx > 0 \\ +\infty & \text{if } dg/dx < 0 \end{cases}$

Selecting a logarithmic function with a base b of $\epsilon(t)$ for g such that:

$$\forall b \in \mathbb{R}^+, b \neq 1$$
$$g(\epsilon) = \frac{ln\,|\epsilon|}{ln\,b} = log_b|\epsilon|, \tag{27}$$

it is straightforward to demonstrate that the logarithm satisfies the three conditions on g on $\mathbb{R}^+$. The effect of the base of the logarithm on events generation will be presented in section 3 Using (27), one can compute a time series $\{t_i\}$:

$$\forall b \in \mathbb{R}^+, b \neq 1, i \in \mathbb{N}^+$$
$$\left\{ t_i \,\middle|\, \left| \frac{ln\,|\epsilon(t_i)|}{ln\,b} - \frac{ln\,|\epsilon(t_{i-1})|}{ln\,b} \right| = 1 \right\} \tag{28}$$

A polarity information, p_i, is added to the time series (28) to express if $\epsilon(t)$ has increased or decreased between two samples of the series:

$$b > 1$$
$$(t_i, p_i = +1) \mid \left| \frac{\epsilon(t_i)}{\epsilon(t_{i-1})} \right| = b \tag{29}$$
$$(t_i, p_i = -1) \mid \left| \frac{\epsilon(t_{i-1})}{\epsilon(t_i)} \right| = b \tag{30}$$

For the sake of simplicity, we will use $b > 1$ in the rest of the paper. But the same relationships used to generate an events series for $b > 1$ can be derived if $0 < b < 1$:

$$0 < b < 1$$
$$(t_i, p_i = +1) \mid \left| \frac{\epsilon(t_{i-1})}{\epsilon(t_i)} \right| = b, \tag{31}$$
$$(t_i, p_i = -1) \mid \left| \frac{\epsilon(t_i)}{\epsilon(t_{i-1})} \right| = b. \tag{32}$$

Finally, as $\epsilon(t)$ can change sign between two events, we added a third piece of information to each event representing a change of sign between the current event and the prior one: s_i.

Therefore, an event e_i is defined as a triplet (t_i, p_i, s_i) in which t_i represents the event time, p_i represents its polarity (either 1 or −1, could be represented by a single bit) and a Boolean s_i (0/1) representing the fact that the sign of the signal changed between the e_{i-1} and e_i.

Using these notations, one can construct the time course of the original signal $\epsilon(t)$ using[1]:

$$\epsilon(t) = \epsilon(t_0) + (-1)^{s_N} \sum_{i=1}^{N} (b^{-p_i} - 1)\,\epsilon(t_{i-1}) H(t - t_i), b > 1 \tag{33}$$

in which $H(x)$ represents the Heaviside step function:

$$H(x) = \frac{d}{dx} max(x, 0). \tag{34}$$

2.3.1. Arbitrary Accuracy

The goal of this section is to demonstrate that the proposed logarithmic discretization can be used to reach a stable point with an arbitrary accuracy. Using (29), the value of the error signal at time t_i can be written using a geometric recursion:

$$\epsilon(t_i) = \frac{\epsilon(t_0)}{b^i}. \tag{35}$$

From (35), it is clear that from any finite value $\epsilon(t_0)$:

$$\lim_{n\to+\infty} \epsilon(t_i) = 0. \tag{36}$$

Equation (36) shows that one can reach an arbitrary precision using the logarithmic discretization proposed in this section. The arbitrary precision at which the error is considered null (therefore at which the system has reached the setpoint) is a design parameter of the control law.

2.3.2. Refractory Period

Theoretically when ϵ reaches zero, the number of events goes to infinity and the duration between two events reaches zero. In addition, even if the accuracy of the clock is very high, practically the computations needed to evaluate the ratios (29) and (30) can take some time. Therefore, for practical implementations, we define the refractory period as the minimum time between two events that the system can generate. These limitations put some constraints on the overall system as the ratio between the past error and the current one could be crossed multiple times during the refractory period. To counter this issue, we added a last parameter to the event representing the number of times the threshold has been crossed between two events r_n:

$$r_i = \left\lfloor \frac{ln\left(\epsilon(t_{i-1})/\epsilon(t_i)\right)}{ln\,b} \right\rfloor \quad \text{if } p_i = -1 \tag{37}$$
$$= \left\lfloor \frac{ln\left(\epsilon(t_i)/\epsilon(t_{i-1})\right)}{ln\,b} \right\rfloor \quad \text{if } p_i = +1. \tag{38}$$

Equation (33) can be written to include r_n:

$$\epsilon(t) = \epsilon(t_0) + (-1)^{s_N} \sum_{i=1}^{N} (b^{-p_i r_i} - 1)\,\epsilon(t_{i-1}) H(t - t_i), b > 1. \tag{39}$$

[1]If $0 < b < 1, \epsilon(t) = \epsilon(t_0) + (-1)^{s_N} \sum_{i=1}^{N} (b^{p_i} - 1)\,\epsilon(t_{i-1}) H(t - t_i)$

The presented discretization method allows us to benefit from the event-based representation in the control context. In the experiments section, we present its application to the control of a second order dynamical system.

3. EXPERIMENTS

The proposed event-based non-linear discretization and the resulting control laws are tested on two systems different systems. The first one is a classical second order dynamical system while in the second example we stabilize an inverted pendulum on a cart.

3.1. Control of a Second Order Dynamical System

This section presents the control of a second order dynamical system with 0.1 and 0.01 s as time constants. The second-order system is controlled by a proportional-integral-derivative controller (see Appendix in **Supplementary Material**, section 3) with a proportional gain equal to 1, an integral gain equal to 5, and a derivative gain equal to zero. This example is used as an academic demonstrator of the new event-based discretization method. Therefore, this second-order model doesn't describe a particular system and the units of the controlled signals, the setpoint, the output or of any of the system internal signals are arbitrary.

Figure 3A presents the results of the simulation when a unitary step (the setpoint of the system goes from zero to one) is applied to the controlled system 100 ms after the beginning of the simulation. The base of the logarithmic discretization function was set to 1.05 and the refractory period was set to 1 millisecond. The upper row represents the time course of the setpoint and the output of the system. The second row represents the frequency of the events generated. Positive (negative) frequencies correspond to positive (negative) polarity events. The events frequency was computed using the convolution of a 2.5 ms normal distribution with the time of the events as it used in neuroscience to compute the discharge frequency of neuronal activities (MacPherson and Aldridge, 1979; Richmond et al., 1987). The last row represents the control sent to the system to reach the setpoint. Importantly, the control is updated only when an event is generated. **Figure 3A** shows that the output of the system reaches the setpoint. In addition, when the system output reaches the target (around 2 s, final error = 6.48e-4), no event is generated and the control remains constant. 142 events were generated during this simulation.

To test how a choice of base for the logarithmic discretization function affects the control quality, we ran a series of simulations for setpoint amplitude ranging from 1 to 1,000. Upper row in **Figure 3B** shows the evolution of the mean squared error during the last 250 ms as a function of the setpoint amplitude for three different bases (1.01: black lines, 1.05: red lines, 1.1: blue lines). The average mean squared error is statistically independent of the setpoint amplitude (mean ± standard error of the mean squared error. 1.01: 6.64e-5 ± 5.16e-6, 1.05: 6.56e-5 ± 8.92e-6, 1.1: 6.41e-5 ± 1.52e-5). Lower row in **Figure 3B** shows that the number of events generated during a simulation for the

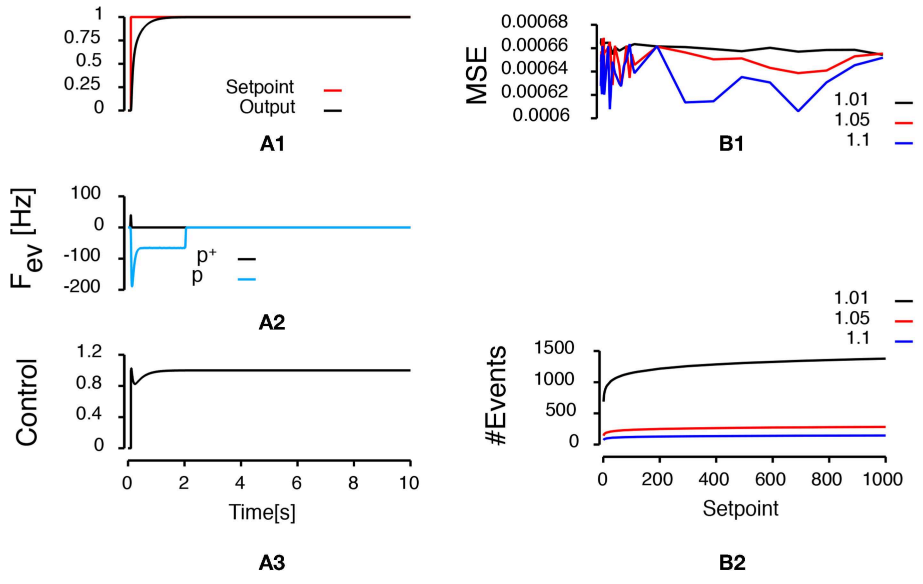


FIGURE 3 | Control of a second-order system. Left column presents the simulation of a second order transfer function controlled by an event-based proportional-integral controller when a step input is applied after 100 milliseconds. **(A1)** Represents the time course of the setpoint (red line) and the time course of the output of the system (black line). **(A2)** Represents the frequency of events as a function of time. Positive frequencies correspond to the frequency of positive polarity events representing an increase of the error signal (p^+: black line). Negative frequencies correspond to the frequency of negative polarity events representing a decrease of the error signal (p^-: light blue line). **(A3)** Represents the time course of the control applied to the system to reach the setpoint. The right column represents the sensitivity of the controlled system to a change of setpoint for different bases (represented by different colors). **(B1)** Represents the evolution of the mean squared error as a function of the setpoint. **(B2)** Represents the evolution of the number of events triggered during a simulation as a function of the setpoint.

same conditions decreases when the base of the logarithmic discretization function increases.

This first set of simulations demonstrated that the absolute error remained constant over a wide range of setpoint amplitudes (from 1 to 1,000) without changing either the discretization or the PID parameters. As the absolute error at the end of the simulation remained mostly constant with increasing amplitude, the relative error is decreasing.

3.2. Stabilization of an Inverted Pendulum

We have implemented a numerical simulation of an inverted pendulum put on a cart. When a perturbation (in the form of an external force T) is applied to the cart, the pendulum is moved out of its equilibrium position. An event-based proportional-integral-derivative (PID) controller stabilizes the pendulum in an inverted position through a control force F applied to the cart. The model of this dynamical system is presented in the Appendix in **Supplementary Material**.

In the first two simulations, we measure the angular orientation of the pendulum. In the last two simulations, we used two sensors to measure the cart position and the angular orientation of the pendulum. In all the simulations, we control the force applied to the cart to stabilize the pendulum in upright position. In each of the following simulations, the pendulum started in the upright inverted position and we injected a fifty milliseconds perturbation one hundred milliseconds after the beginning of the simulation.

3.2.1. Proportional-Integral-Derivative Control Law

In this section, the pendulum angle was controlled by a proportional-integral-derivative (PID) controller with a proportional gain of 100, an integration gain of 2 and a derivative gain of 10. We compared a traditional discrete PID controller with an event-based version of the PID controller. The base of the logarithmic discretization function was set to 1.05 for the event-based control law simulations and the refractory period of the sensor was set to 1 ms.

Figure 4 compares the two control laws when the control sampling rate of the discrete controller was set to 1 kHz. For the event-based control law, the control was updated each time an event arrives. **Figure 4A** shows the results of the simulation with the event-based control law. **Figure 4B** represents the results of the simulation with the discrete PID. The last row in **Figures 4A,B** represents the angular error of the pendulum. In

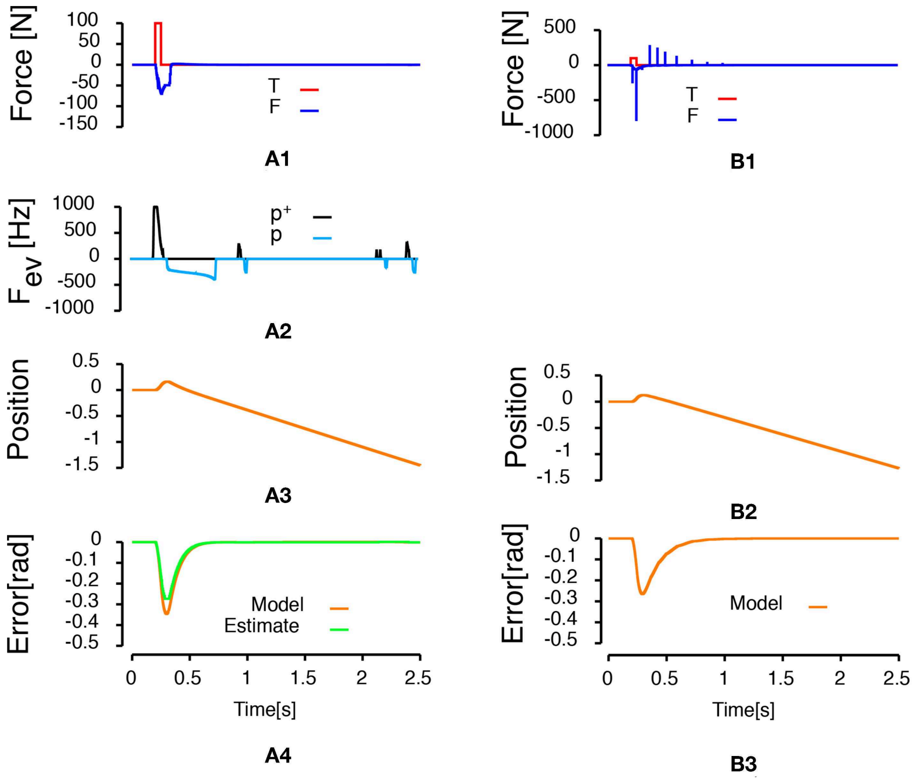


FIGURE 4 | Inverted pendulum on a cart controlled by either an event-based controller (Left column) or a discrete PID with a one millisecond sampling time (Right column). **(A1,B1)** Represent the time course of forces applied to the system. The red curve represents the 100 N perturbation applied to the cart during 50 ms. The blue curve represents the time course of the input force applied to the cart to stabilize the pendulum. **(A2)** Represents the frequency of events generated during the simulation. Positive frequencies correspond to positive polarity events representing an increase of the error signal (p^+: black line). Negative frequencies correspond to negative polarity event representing a decrease of the error signal (p^-: light blue line). **(A3,B2)** Represent the time course of the cart position. **(A4,B3)** Represent the angular error of system (orange line, angle with respect to the vertical) and the estimate of the error built from the received events (green line).

both control law conditions, the pendulum is stabilized by the control law. The maximum error is smaller in the case of the discrete PID controller. However, with the discrete PID, there are peaks of desired force above 500 N. In addition, the discrete version of the control law updated the force applied to the cart 2,500 times while the event-based version updated that force only 199 times, so the system is updated roughly 12.5 times less often using the event-based control.

This simulation shows that the frequency of events generation is sensitive to the amplitude of the error (the smaller the error, the higher the number of events) and to the amplitude of the time derivative of the error (the larger the amplitude of the time derivative of the error, the higher the events frequency). In **Figure 4**, the frequency of positive events increases during the first part of the error increase (when the time derivative of the error is important). Then, as the amplitude of the error time derivative decreases and the amplitude of the error increases, the frequency of events generation decreases as well. Afterwards, there is a small plateau during which no positive nor negative events are generated (corresponding to the peak of the error signal, when there is no modification of the error amplitude large enough to trigger an event). Finally, there is an increase of the frequency of negative events as the error decreases due to feedback control of the PID. It can be seen that, as the rate of change of the error during the decrease of the error is smaller, the overall frequency of the events is smaller. However, even if the time derivative of the error decreases, the events generation frequency increases toward the end of the stabilization period as the error reaches zero. These observations show the sensitivity of the events generation frequency to both the rate of change of the error and the amplitude of the error.

Figure 5 compares a discrete PID control law and an event-based control law when the discrete controller and the event-based controller are set to update the control output at 20 Hz. This experiment presents how the event-based controller can be used to update regularly a control output. Therefore, contrarily to all the other experiments, in this case, while the estimate of the error and the integral of the error are updated each time an event is generated, the control is updated every 50 ms. The striking point in **Figure 5** is that the event-based controller stabilizes the

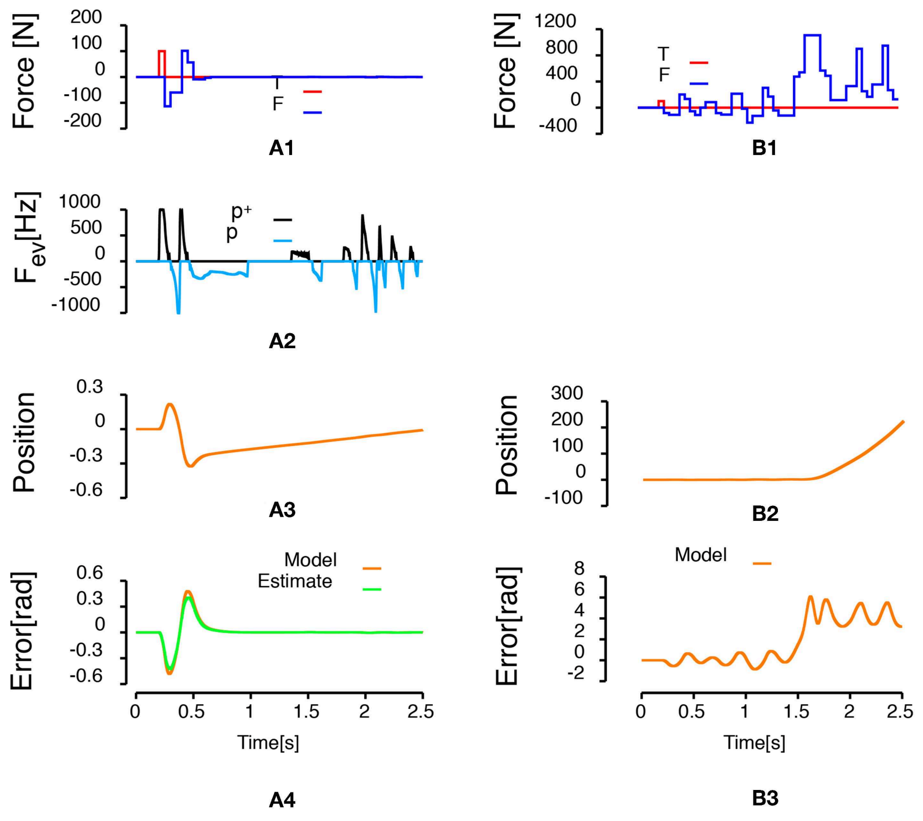


FIGURE 5 | Inverted pendulum controlled with the event-based controller and the discrete controller set to update the control output at 20 Hz. The discrete controller fails to stabilize the pendulum when the control frequency is reduced from 1KHz to 20 Hz. Each subfigures and the colors convention are defined as in **Figure 4**. **(A1,B1)** Represent the time course of forces applied to the system. The red curve represents the 100 N perturbation applied to the cart during 50 ms. The blue curve represents the time course of the input force applied to the cart to stabilize the pendulum. **(A2)** Represents the frequency of events generated during the simulation. Positive frequencies correspond to positive polarity events representing an increase of the error signal (p^+: black line). Negative frequencies correspond to negative polarity event representing a decrease of the error signal (p^-: light blue line). **(A3,B2)** Represent the time course of the cart position. **(A4,B3)** Represent the angular error of system (orange line, angle with respect to the vertical) and the estimate of the error built from the received events (green line).

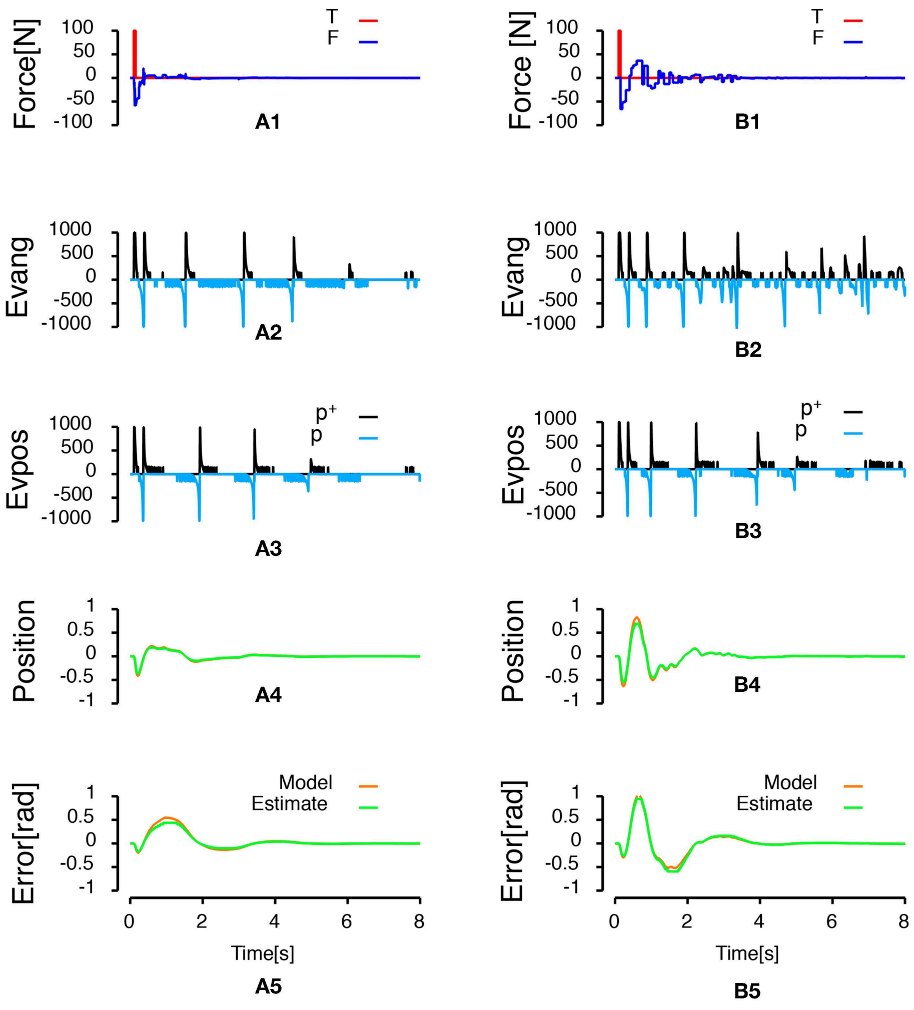


FIGURE 6 | Inverted pendulum controlled by state-space feedback law. Left column represents an event-based state-space feedback law with the force applied to the cart updated at each event. Right column represents the same control law but with the force applied to the cart updated every 50 ms. **(A1,B1)** Show the time course of forces applied to the system. The red curve represents the 100 N perturbation applied to the system during 50 ms. The blue curve represents the time course of the force applied to the cart to stabilize the pendulum. **(A2,A3,B2,B3)** Represent respectively the frequency of pendulum angle and cart position events as a function of time. Positive frequencies correspond to the frequency of positive polarity events representing an increase of the error signal (p^+: black lines). Negative frequencies correspond to the frequency of negative polarity events representing a decrease of the error signal (p^-: light blue lines). **(A4,B4)** Show the time course of the cart position (orange line, deviation from the central position) and the estimate of the position built from the received position events (green line). Finally, **(A5,B5)** Represent the angular error of system (orange line, angle with respect to the vertical) and the estimate of the error built from the received events (green line).

pendulum (the effect of the external force is negated and the pendulum remains in an upright position) while the system is unstable with the discrete controller. More events are generated in this condition compared to the condition in **Figure 4A** (432 events here vs. 199 events in **Figure 4A**).

In this set of simulations, we showed that the system could reject a perturbation applied to the cart and keep the pendulum stable while decreasing the number of times the controlled input is updated by a factor 12 compared to an equivalent time-discrete PID with a 1 ms sample time. We also showed that we can keep the pendulum stable with the same controller parameters while updating the force applied to the cart every 50 ms instead of each time an event is received, reducing drastically the burden put on the actuator.

3.2.2. State-Space Feedback Control Law

After the controllers reject the perturbation and stabilize the pendulum angle, the third row in **Figures 4**, **5** shows that the cart keeps moving. This displacement is generated because the stabilized angle is not the upright position. It is not possible to build an observer of the cart position based on pendulum angle measurements as the observability matrix in this condition is not full-rank. Therefore, it is not possible to stabilize both the pendulum angle and the cart position using a single PID with a single sensor on the pendulum angle. In this section, we present a state-space feedback control law that stabilizes both the pendulum angle and the cart position. To that goal, we used two sensors, one on the cart position and one on the pendulum angle. The optimal gain K of the state-space feedback law was computed using the linear-quadratic regulator[2] updated each time an event is emitted by one of the sensors. We used the estimate of the cart position, the derivative of the estimate of the cart position, the estimate of the angular error and the derivative of the estimate of the angular error as states for the feedback. The estimate of the derivatives was built like the derivative component in the PID controller of the previous section. **Figure 6A** shows the results of the simulation when the force applied to the cart is updated every time one of the sensors emits an event. **Figure 6B** shows the results of the simulation when the force applied to the cart is updated only every 50 ms. The simulations show that the designed state-space feedback control law rejects the perturbation and that both the cart position and the pendulum angle are stable by the end of the simulations. A total of 1049 events (451 position events, 598 angular events) were generated during the simulation presented in **Figure 6A**, while a total of 1,679 events (1,113 position events, 566 angular events) were generated during the simulation in **Figure 6B**.

In this final set of simulations, we stabilized both the inverted pendulum angle and the cart position. The designed system rejected the perturbation and stabilized both the pendulum angle and the cart position. As for the PID simulations, the force applied to cart was updated every 50 ms and we showed that we could keep the system stable.

4. DISCUSSION

This work proposes a new non-linear event-based discretization method and the associated proportional-integral-derivative and state-space feedback control laws. The key novelty of the new event generator function lies in the principle that the time between two events must decrease when the discretized signal (e.g., an error signal) tends to zero. Contrary to current event-based discretization/control schemes (e.g., Arzén, 1999; Bernhardsson and Aström, 1999; Miskowicz, 2005, 2007; Tabuada, 2007; Lunze and Lehmann, 2010; Donkers and Heemels, 2012; Heemels et al., 2012), the generated events do not contain the value of the signal. Instead, a transmitted event contains four parts: the time of the change, a polarity (did the signal increased or decreased), a sign changed bit (did the signal's sign changed) and a refractory gain (the number of times the threshold set in the level-sampling mechanism has been crossed during a refractory period) to account for multiple level crossings during the shortest measurable duration between two events.

[2]A detailed description of the linear-quadratic regulator is outside the scope of this paper. The interested reader can learn about it in various textbooks (e.g., in Corriou, 2004).

Using the new event-triggering scheme, we showed that all the uncertainties on the signal approximation emerge from uncertainties on the measurement of the duration between two events. Also, because the inter-event duration is the key information to the signal reconstruction, the components of the system do not require synchronized clocks but all of them must measure a duration with the same accuracy.

In addition, the precision of the representation of the error signal stored in any active part of the control mechanism (e.g., 128 bits to represent the state of the system) can be much higher than the number of bits used to transmit an event (e.g., an 32 bits unsigned integer to represent milliseconds). As a result, the design of a control scheme based on the new method of this paper can increase the controlled signal dynamic range without increasing the network data bandwidth needed to transmit the information between the different components of the controlled system.

Our results demonstrate that the new method combines the advantages of analog continuous time systems

- it can reach an arbitrary precision with a very high dynamic range
- with the advantages of event-based control
- no events are generated when the error is null.

In addition, the amount of data transmitted between active parts of the control system can be smaller than the memory needed to store the signals value. This makes the overall system more efficient energetically.

DATA AVAILABILITY

All datasets generated for this study are included in the manuscript and the **Supplementary Files**.

AUTHOR CONTRIBUTIONS

PD, S-HI, and RB conceived of the presented idea. PD developed the theory and performed the computations. All authors discussed the results and contributed to the final manuscript.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2019.00827/full#supplementary-material

REFERENCES

Arzén, K.-E. (1999). "A simple event-based pid controller," in *Proceedings of 14th IFAC World Congress*, Vol. 18 (Toulouse), 423–428.

Astrom, K. J., and Bernhardsson, B. M. (2002). "Comparison of riemann and lebesgue sampling for first order stochastic systems," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, Vol. 2 (Las Vegas, NV: IEEE), 2011–2016.

Bernhardsson, B., and Aström, K. (1999). "Comparison of periodic and event based sampling for first-order stochastic systems," in *Preprints of the 14th IFAC World Congress* (Toulouse).

Cervin, A., and Astrom, K. J. (2007). "On limit cycles in event-based control systems," in *Decision and Control, 2007 46th IEEE Conference on* (New Orleans, LA: IEEE), 3190–3195.

Corriou, J.-P. (2004). *Process Control: Theory and Applications*. London: Springer-Verlag.

Donkers, M., and Heemels, W. (2012). Output-based event-triggered control with guaranteed $\mathcal{L}_\infty$-gain and improved and decentralized event-triggering. *IEEE Trans. Autom. Control* 57, 1362–1376. doi: 10.1109/TAC.2011.2174696

Dorf, R., Farren, M., and Phillips, C. (1962). Adaptive sampling frequency for sampled-data control systems. *IRE Trans. Autom. Control* 7, 38–47. doi: 10.1109/TAC.1962.1105415

Heemels, W., Johansson, K. H., and Tabuada, P. (2012). "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on* (Maui, HI: IEEE), 3270–3285.

Hetel, L., Fiter, C., Omran, H., Seuret, A., Fridman, E., Richard, J.-P., et al. (2017). Recent developments on the stability of systems with aperiodic sampling: an overview. *Automatica* 76, 309–335. doi: 10.1016/j.automatica.2016.10.023

Heymann, M., Lin, F., Meyer, S., and Resmerita, S. (2005). Analysis of zeno behaviors in hybrid systems. *IEEE Trans. Autom. Control* 50, 376–383. doi: 10.1109/TAC.2005.843874

Lampersky, A. G., and Ames, A. D. (2013). Lyapunov theory for zeno stability. *IEEE Trans. Autom. Control* 58, 110–112. doi: 10.1109/TAC.2012.2208292

Lunze, J., and Lehmann, D. (2010). A state-feedback approach to event-based control. *Automatica* 46, 211–215. doi: 10.1016/j.automatica.2009.10.035

MacPherson, J. M., and Aldridge, J. W. (1979). A quantitative method of computer analysis of spike train data collected from behaving animals. *Brain Res.* 175, 183–187. doi: 10.1016/0006-8993(79)90530-4

Maxwell, J. C. (1867). On governors. *Proc. R. Soc. Lond.* 16, 270–283. doi: 10.1098/rspl.1867.0055

Miskowicz, M. (2005). "Sampling of signals in energy domain," in *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, Vol. 1 (Catania: IEEE), 4.

Miskowicz, M. (2007). Asymptotic effectiveness of the event-based sampling according to the integral criterion. *Sensors* 7, 16–37. doi: 10.3390/s7010016

Pontryagin, L. S., Mishchenko, E. F., Boltyanskii, V. G., and Gamkrelidze, R. V. (1962). *The Mathematical Theory of Optimal Processes*. New York, NY; London: Interscience Publishers John Wiley & Sons, Inc.

Posch, C., Matolin, D., and Wohlgenannt, R. (2008). "An asynchronous time-based image sensor," in *2008 IEEE International Symposium on Circuits and Systems* (Seattle, WA), 2130–2133.

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid State Circuits* 46, 259–275. doi: 10.1109/JSSC.2010.2085952

Richmond, B. J., Optican, L. M., Podell, M., and Spitzer, H. (1987). Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. i. response characteristics. *J. Neurophysiol.* 57, 132–146. doi: 10.1152/jn.1987.57.1.132

Tabuada, P. (2007). Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. Autom. Control* 52, 1680–1685. doi: 10.1109/TAC.2007.904277

Tomovic, R., and Bekey, G. (1966). Adaptive sampling based on amplitude sensitivity. *IEEE Trans. Autom. Control* 11, 282–284. doi: 10.1109/TAC.1966.1098308

Conflict of Interest Statement: 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.

ORIGINAL RESEARCH
published: 14 November 2019
doi: 10.3389/fnins.2019.01201

Accelerated Physical Emulation of Bayesian Inference in Spiking Neural Networks

Akos F. Kungl[1]*, *Sebastian Schmitt*[1], *Johann Klähn*[1], *Paul Müller*[1], *Andreas Baumbach*[1], *Dominik Dold*[1], *Alexander Kugele*[1], *Eric Müller*[1], *Christoph Koke*[1], *Mitja Kleider*[1], *Christian Mauch*[1], *Oliver Breitwieser*[1], *Luziwei Leng*[1], *Nico Gürtler*[1], *Maurice Güttler*[1], *Dan Husmann*[1], *Kai Husmann*[1], *Andreas Hartel*[1], *Vitali Karasenko*[1], *Andreas Grübl*[1], *Johannes Schemmel*[1], *Karlheinz Meier*[1] and *Mihai A. Petrovici*[1,2]

[1] *Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany,* [2] *Department of Physiology, University of Bern, Bern, Switzerland*

OPEN ACCESS

Edited by:
*Themis Prodromakis,
University of Southampton,
United Kingdom*

Reviewed by:
*Alexantrou Serb,
University of Southampton,
United Kingdom
Adnan Mehonic,
University College London,
United Kingdom*

***Correspondence:**
*Akos F. Kungl
fkungl@kip.uni-heidelberg.de*

Specialty section:
*This article was submitted to
Neuromorphic Engineering,
a section of the journal
Frontiers in Neuroscience*

Received: *10 July 2019*
Accepted: *23 October 2019*
Published: *14 November 2019*

Citation:
Kungl AF, Schmitt S, Klähn J, Müller P, Baumbach A, Dold D, Kugele A, Müller E, Koke C, Kleider M, Mauch C, Breitwieser O, Leng L, Gürtler N, Güttler M, Husmann D, Husmann K, Hartel A, Karasenko V, Grübl A, Schemmel J, Meier K and Petrovici MA (2019) Accelerated Physical Emulation of Bayesian Inference in Spiking Neural Networks. Front. Neurosci. 13:1201. doi: 10.3389/fnins.2019.01201

The massively parallel nature of biological information processing plays an important role due to its superiority in comparison to human-engineered computing devices. In particular, it may hold the key to overcoming the von Neumann bottleneck that limits contemporary computer architectures. Physical-model neuromorphic devices seek to replicate not only this inherent parallelism, but also aspects of its microscopic dynamics in analog circuits emulating neurons and synapses. However, these machines require network models that are not only adept at solving particular tasks, but that can also cope with the inherent imperfections of analog substrates. We present a spiking network model that performs Bayesian inference through sampling on the BrainScaleS neuromorphic platform, where we use it for generative and discriminative computations on visual data. By illustrating its functionality on this platform, we implicitly demonstrate its robustness to various substrate-specific distortive effects, as well as its accelerated capability for computation. These results showcase the advantages of brain-inspired physical computation and provide important building blocks for large-scale neuromorphic applications.

Keywords: physical models, neuromorphic engineering, massively parallel computing, spiking neurons, recurrent neural networks, neural sampling, probabilistic inference

1. INTRODUCTION

The aggressive pursuit of Moore's law in conventional computing architectures is slowly but surely nearing its end (Waldrop, 2016), with difficult-to-overcome physical effects, such as heat production and quantum uncertainty, representing the main limiting factors. The so-called von Neumann bottleneck between processing and memory units represents the main cause, as it effectively limits the speed of these largely serial computation devices. The most promising solutions come in the form of massively parallel devices, many of which are based on brain-inspired computing paradigms (Indiveri et al., 2011; Furber, 2016), each with its own advantages and drawbacks.

Among the various approaches to such neuromorphic computing, one class of devices is dedicated to the physical emulation of cortical circuits; not only do they instantiate neurons and synapses that operate in parallel and independently of each other, but these units are actually represented by distinct circuits that emulate the dynamics of their biological archetypes (Mead, 1990; Indiveri et al., 2006; Jo et al., 2010; Schemmel et al., 2010; Pfeil et al., 2013; Qiao et al., 2015; Chang et al., 2016; Wunderlich et al., 2019). Some important advantages of this approach lie in their reduced power consumption and enhanced speed compared to conventional simulations of biological neuronal networks, which represent direct payoffs of replacing the resource-intensive numerical calculation of neuro-synaptic dynamics with the physics of the devices themselves.

However, such computation with analog dynamics, without the convenience of binarization, as used in digital devices, has a downside of its own: variability in the manufacturing process (fixed pattern noise) and temporal noise both lead to reduced controllability of the circuit dynamics. Additionally, one relinquishes much of the freedom permitted by conventional algorithms and simulations, as one is confined by the dynamics and parameter ranges cast into the silicon substrate. The main challenge of exploiting these systems, therefore, lies in designing performance network models using the available components while maintaining a degree of robustness toward the substrate-induced distortions. Just like for the devices themselves, inspiration for such models often comes from neuroscience, as the brain needs to meet similar demands.

With accumulating experimental evidence (Berkes et al., 2011; Pouget et al., 2013; Haefner et al., 2016; Orbán et al., 2016), the view of the brain itself as an analytical computation device has shifted. The stochastic nature of neural activity *in vivo* is being increasingly regarded as an explicit computational resource rather than a nuisance that needs to be dealt with by sophisticated error-correcting mechanisms or by averaging over populations. Under the assumption that stochastic brain dynamics reflect an ongoing process of Bayesian inference in continuous time, the output variability of single neurons can be interpreted as a representation of uncertainty. Theories of neural sampling (Buesing et al., 2011; Hennequin et al., 2014; Aitchison and Lengyel, 2016; Petrovici et al., 2016; Kutschireiter et al., 2017) provide an analytical framework for embedding this type of computation in spiking neural networks.

In this paper we describe the realization of neural sampling with networks of leaky integrate-and-fire neurons (Petrovici et al., 2016) on the BrainScaleS accelerated neuromorphic platform (Schemmel et al., 2010). With appropriate training, the variability of the analog components can be naturally compensated and incorporated into a functional network structure, while the network's ongoing dynamics make explicit use of the analog substrate's intrinsic acceleration for Bayesian inference (section 2.3). We demonstrate sampling from low-dimensional target probability distributions with randomly chosen parameters (section 3.1) as well as inference in high-dimensional spaces constrained by real-world data, by solving associated classification and constraint satisfaction problems (pattern completion, section 3.2). All network components are fully contained on the neuromorphic substrate, with external inputs only used for sensory evidence (visual data). Our work thereby contributes to the search for novel paradigms of information processing that can directly benefit from the features of neuro-inspired physical model systems.

2. METHODS

2.1. The BrainScaleS System

BrainScaleS (Schemmel et al., 2010) is a mixed-signal neuromorphic system, realized in 180 nm CMOS technology, that emulates networks of spiking neurons. Each BrainScaleS wafer module consists of a 20 cm silicon wafer with 384 HICANN (High Input Count Analog Neural Network) chips, see **Figure 1A**. On each chip, 512 analog circuits emulate the adaptive exponential integrate-and-fire (AdEx) model (Brette and Gerstner, 2005; Millner et al., 2010) of spiking neurons with conductance-based synapses. The dynamics evolve with an acceleration factor of 10^4 with respect to biological time, i.e., all specific time constants (synaptic, membrane, adaptation) are $\sim 10^4$ times smaller than typical corresponding values found in biology (Schemmel et al., 2010; Petrovici et al., 2014). To preserve compatibility with related literature (Petrovici et al., 2016; Schmitt et al., 2017; Leng et al., 2018; Dold et al., 2019), we refer to system parameters in the biological domain unless otherwise specified, e.g., a membrane time constant given as 10 ms is actually accelerated to 1 μs on the chip.

The parameters of the neuron circuits are stored in analog memory cells (floating gates) with 10 bit resolution, and the synaptic weights are stored in 4 bit SRAM (Schemmel et al., 2010). The analog memory cells are similar to the ones in Lande et al. (1996), and they are described in Loock (2006) and Millner (2012).

Spike events are transported digitally and can reach all other neurons on the wafer with the help of an additional redistribution layer that instantiates an on-wafer circuit-switched network (Zoschke et al., 2017) (**Figures 1B,C**).

Because of mismatch effects (fixed-pattern noise) inherent to the substrate, the response to incoming stimuli varies from neuron to neuron (**Figure 1D**). In order to bring all neurons into the desired regime and to reduce the neuron-to-neuron response variability, we employ a standard calibration procedure that is performed only once, during the commissioning of the system (Petrovici et al., 2017b; Schmitt et al., 2017). Nevertheless, even after calibration, a significant degree of diversity persists (**Figure 1E**). The emulation of functional networks that do not rely on population averaging therefore requires appropriate training algorithms (section 3.2).

2.2. Sampling With Leaky Integrate-and-Fire Neurons

The theory of sampling with leaky integrate-and-fire neurons (Petrovici et al., 2016) describes a mapping between the dynamics of a population of neurons with conductance-based synapses (equations given in **Table 1**) and a Markov-chain Monte Carlo sampling process from an underlying probability distribution over binary random variables (RVs). Each neuron in such a sampling network corresponds to one of these RVs: if the k-th neuron has spiked in the recent past and is currently refractory, then it is considered to be in the *on-state* $z_k = 1$, otherwise it is in

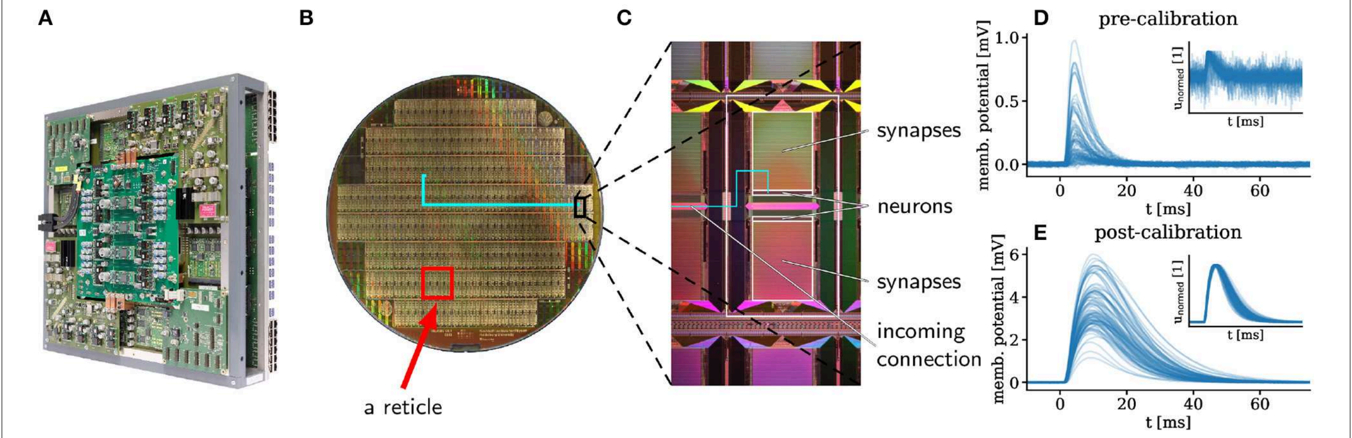


FIGURE 1 | (A) Photograph of a fully assembled wafer module of the BrainScaleS system (dimensions: 50 × 50 × 15 cm). One module hosts 384 HICANN chips on 48 reticles, with 512 physical neurons per chip and 220 synapse circuits per neuron. The wafer itself lies at the center of the module and is itself not visible. 48 FPGAs are responsible for I/O and experiment control. Support PCBs provide power supply for the on-wafer circuits as well as access to neuron membrane voltages. The connectors for inter-wafer (sockets resembling USB-A) and off-wafer/host connectivity (Gigabit-Ethernet sockets) are distributed over all four edges of the main PCB. Mechanical stability is provided by an aluminum frame. **(B)** The wafer itself is composed of 48 reticles (e.g., red rectangle), each containing 8 HICANN chips (e.g., black rectangle, enlarged in C). Inter-reticle connectivity is added in a post-processing step. **(C)** On a single HICANN chip, the largest area is occupied by the two synapse matrices which instantiate connections to the neurons positioned in the neuron array. **(D,E)** Postsynaptic potentials (PSPs) measured on 100 different neuron membranes using the same parameter settings before **(D)** and after **(E)** calibration. The insets show the height-normalized PSPs. The calibration serves two purposes. First, it provides a translation rule between the desired neuron parameters and the technical parameters set on the hardware. In this case, it brings the time constants τ_{mem} and τ_{syn} close to the target of 8 ms, as evidenced by the small spread of the normalized PSPs. Second, in the absence of such a translation rule, it sets the circuits to their correct working points. Here, this happens for the synaptic weights: after calibration, PSP heights are, on average closer to the target working point of 3 mV, but they remain highly diverse due to the variability of the substrate. For more details see Schmitt et al. (2017). The PSPs are averaged over 375 presynaptic spikes and smoothed with a Savitzky-Golay filter (Savitzky and Golay, 1964) to eliminate readout noise. The time-constants are given in the biological domain, but they are 10^4 faster on the system.

the *off-state* $z_k = 0$ (**Figures 2A,B**). With appropriate synaptic parameters, such a network can approximately sample from a Boltzmann distribution defined by

$$p^*(\boldsymbol{z}) = \frac{1}{Z}\exp\left(\frac{1}{2}\boldsymbol{z}^T\boldsymbol{W}\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{b}\right), \quad (1)$$

where Z is the partition sum, $\boldsymbol{W}$ a symmetric, zero-diagonal effective weight matrix and b_i the effective bias of the i-th neuron (**Figure 2D**).

In the original model, each neuron receives excitatory and inhibitory Poisson input. This plays two important roles: it transforms a deterministic LIF neuron into a stochastic firing unit and induces a high-conductance state, with an effective membrane time constant that is much smaller than other time constants in the system: $\tau_{\mathrm{eff}} \ll \tau_{\mathrm{syn}}, \tau_{\mathrm{ref}}$ (see e.g., Destexhe et al., 2003; Petrovici, 2016), which symmetrizes the neural activation function, as explained in the following. The activation function of an LIF neuron without noise features a sharp onset, but only a slow converge to its maximum value, hence being highly asymmetric around the point of 50 % activity. Background Poisson noise smears out the onset of the activation function, while the reduced membrane time constant accelerates the convergence to the maximum, making the activation function more symmetric and thus more similar to a logistic function, which is a pre-requisite for this form of sampling. For the explicit derivation see Petrovici (2016) and Petrovici et al. (2016). A mapping of this activation function to the abovementioned logistic function $1/[1 + \exp(-x)]$ provides the translation from the dimensionless weights and biases of the target distribution to the corresponding biological parameters of the spiking network (Petrovici, 2016).

TABLE 1 | Description of the neuron and synapse model.

Type	**Leaky integrate-and-fire (LIF), conductance-based synapse, exponential kernel**
Subthreshold dynamics	Subthreshold dynamics $[t \notin [t_{\mathrm{sp}}, t_{\mathrm{sp}} + \tau_{\mathrm{ref}})]$: $C_{\mathrm{m}}(\mathrm{d}/\mathrm{d}t)u(t) = -g_{\mathrm{l}}[u(t) - E_{\mathrm{leak}}] - g_{\mathrm{syn}}^{\mathrm{inh}}(t)[u(t) - E_{\mathrm{inh}}] - g_{\mathrm{syn}}^{\mathrm{exc}}(t)[u(t) - E_{\mathrm{exc}}]$
Reset and refractoriness	$[t \in [t_{\mathrm{sp}}, t_{\mathrm{sp}} + \tau_{\mathrm{ref}})]$: $u(t) = V_{\mathrm{reset}}$ This model was emulated on the BrainScaleS system (Schemmel et al., 2010)
Spiking	If $u(t)$ crosses V_{thresh} from below at $t = t_{\mathrm{sp}}$, neuron emits a spike with timestamp t_{sp}
Synapse dynamics	For each presynaptic spike at t_{sp}: $g_{\mathrm{syn}}(t) = J\exp[-(t - t_{\mathrm{sp}} - d)/(\tau_{\mathrm{syn}})]\theta(t - t_{\mathrm{sp}} - d)$ where J is the synaptic weight, d the synaptic delay and θ the Heaviside function This model was emulated on the BrainScaleS system (Schemmel et al., 2010)

The variables are described including their numerical values in the experiment in ***Table 2****.*

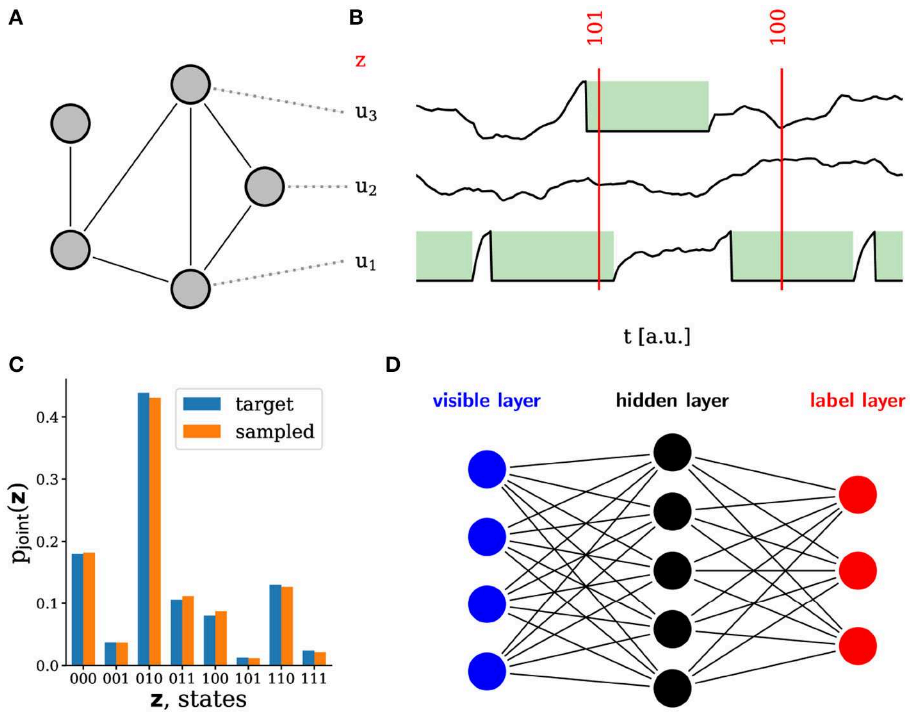


FIGURE 2 | Sampling with leaky integrate-and-fire (LIF) neurons. **(A)** Schematic of a spiking sampling network (SSN) with 5 neurons. Each line represents two reciprocal synaptic connections with equal weights. **(B)** Example membrane potentials of three neurons in the network. Following a spike, the refractory mechanism effectively clamps the membrane potential to the reset value for a duration τ_{ref}. During this time, the RV corresponding to that neuron is in the state $z = 1$ (marked in green). At any point in time, the state sampled by the network can therefore be constructed directly from its output spikes and the refractory time τ_{ref} of the neurons. **(C)** Probability distribution sampled by an SSN with three neurons as compared to the target distribution. **(D)** Based on this framework (Petrovici et al., 2016), hierarchical sampling networks can be built, which can be trained on real-world data. Each line represents a reciprocal connection (two synapses) between the connected neurons.

Although different in their dynamics, such sampling spiking networks (SSNs, **Figure 2D**) function similar to (deep) Boltzmann machines (Hinton et al., 1984), which makes them applicable to the same class of machine learning problems (Leng et al., 2018). Training can be done using an approximation of wake-sleep algorithm (Hinton et al., 1995; Hinton, 2012), which implements maximum-likelihood learning on the training set:

$$\Delta b_i = \eta(\langle z_i \rangle^* - \langle z_i \rangle) \,, \tag{2}$$

$$\Delta W_{ij} = \eta(\langle z_i z_j \rangle^* - \langle z_i z_j \rangle) \,, \tag{3}$$

where $\langle \cdot \rangle$ and $\langle \cdot \rangle^*$ represent averages over the sampled (model or sleep phase) and target (data or wake phase) distribution, respectively, and η is the learning rate.

In order to enable a fully-contained neuromorphic emulation on the BrainScaleS system, the original model had to be modified. The changes in the network structure, noise generation mechanism, and learning algorithm are described in section 2.3.

For low-dimensional, fully specified target distributions, we used the Kullback-Leibler divergence (DKL, Kullback and Leibler, 1951) as a measure of discrepancy between the sampled (p) and the target (p^*) distributions:

$$D_{\text{KL}}(p \parallel p^*) = -\sum_{z_i \in \Omega} p(z_i) \ln \left(\frac{p(z_i)}{p^*(z_i)} \right) \tag{4}$$

This was done in part to preserve comparability with previous studies (Buesing et al., 2011; Petrovici et al., 2015, 2016), but also because the DKL is the natural loss function for maximum likelihood learning. For visual datasets, we used the error rate (ratio of misclassified images in the test set) for discriminative tasks and the mean squared error (MSE) between reconstruction and original image for pattern completion tasks. The MSE is defined as

$$\text{MSE} = \frac{1}{N_{\text{pixels}}} \sum_{k=1}^{N_{\text{pixels}}} \left(z_k^{\text{data}} - z_k^{\text{recon}} \right)^2 \,, \tag{5}$$

where z_k^{data} is the reference data value, z_k^{recon} is the model reconstruction and the sum goes over the N_{pixels} pixels to be reconstructed by the SSN.

2.3. Experimental Setup

The physical emulation of a network model on an analog neuromorphic substrate is not as straightforward as a

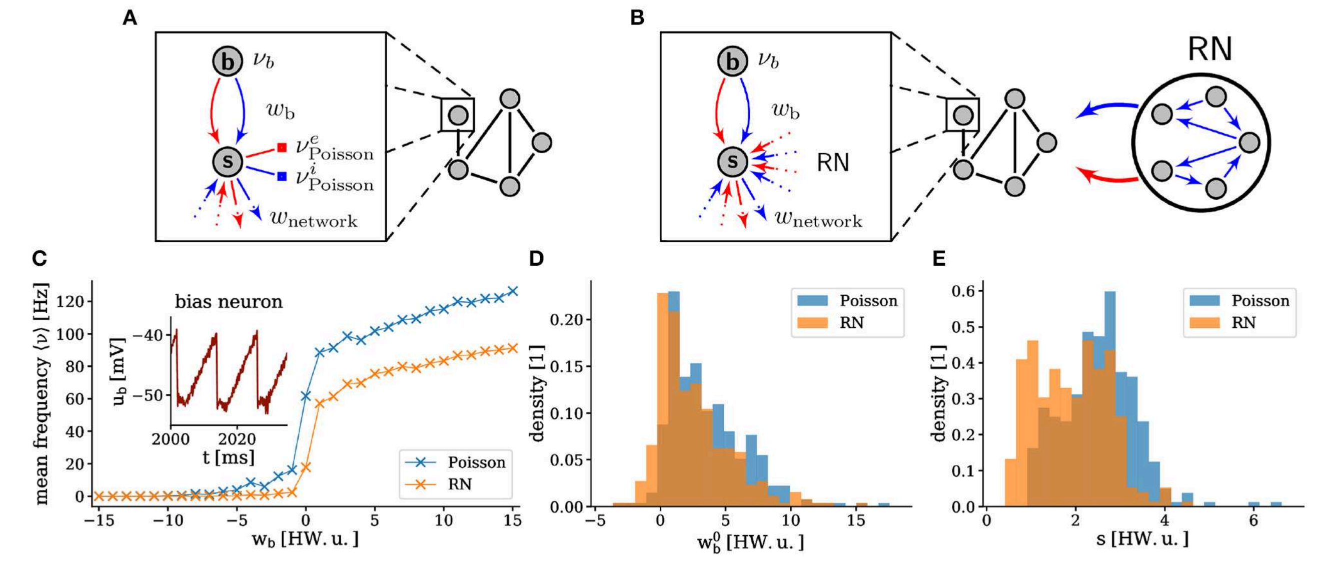


FIGURE 3 | Experimental setup. Each sampling unit is instantiated by a pair of neurons on the hardware. The bias neuron ⓑ is configured with a suprathreshold leak potential and generates a regular spike train that impinges on the sampling neuron Ⓢ, thereby serving as a bias, controlled by w_b. **(A)** As a benchmark, we provided each sampling neuron with private, off-substrate Poisson spike sources. **(B)** Alternatively, in order to reduce the I/O load, the noise was generated by a random network (RN). The RN consisted of randomly connected inhibitory neurons with $E_\mathrm{leak} > V_\mathrm{thresh}$. Connections were randomly assigned, such that each sampling neuron received a fixed number of excitatory and inhibitory pre-synaptic partners (**Table 1**). **(C)** Exemplary activation function (mean firing frequency) of a single sampling neuron with Poisson noise and with an RN as a function of the bias weight. The standard deviation of the trial-to-trial variability is on the order of 0.1 Hz for both activation functions, hence the error bars are too small to be shown. The inset shows the membrane trace of the corresponding bias neuron. **(D,E)** The figures show histograms over all neurons in a sampling network on a calibrated BrainScaleS system. The width s and the midpoint w_b^0 of the activation functions with Poisson noise and with an RN are calculated by fitting the logistic function $\langle\nu\rangle = \nu_0/\{1 + \exp[-(w_\mathrm{b} - w_\mathrm{b}^0)/s]\}$ to the data.

software simulation, as it needs to comply with the constraints imposed by the emulating device. Often, it may be tempting to fine-tune the hardware to a specific configuration that fits one particular network, e.g., by selecting specific neuron and synapse circuits that operate optimally given a particular set of network parameters, or by manually tweaking individual hardware parameters after the network has been mapped and trained on the substrate. Here, we explicitly refrained from any such interventions in order to guarantee the robustness and scalability of our results.

All experiments were carried out on a single module of the BrainScaleS system using a subset of the available HICANN chips. The network setup was specified in the BrainScaleS-specific implementation of PyNN (Davison et al., 2009) and the standard calibration (Schmitt et al., 2017) was used to set the analog parameters. The full setup consisted of two main parts: the SSN and the source of stochasticity.

In the original sampling model (Petrovici et al., 2016), in order to affect biases, the wake-sleep algorithm (Equation 1) requires access to at least one reversal potential (E_l, E_exc, or E_inh), which are all controlled by analog memory cells. Given that rewriting analog memory cells is both less precise and slower than rewriting the SRAM cells controlling the synaptic weights, we modified our SSNs to implement biases by means of synaptic weights. To this end, we replaced individual sampling neurons by sampling units, each realized using two hardware neurons (**Figures 3A,B**). Like in the original model, a sampling neuron was set up to encode the corresponding binary RV. Each sampling neuron was accompanied by a bias neuron set up with a suprathreshold leak potential that ensured regular firing (**Figure 3C**, inset). Each bias neuron projected to its target sampling neuron with both an excitatory and an inhibitory synapse (with independent weights), thus inducing a controllable offset of the sampling neuron's average membrane potential. Because excitatory and inhibitory inputs are routed through different circuits for each neuron, two types of synapses were required to allow the sign of the effective bias to change during training. For larger networks, in order to optimize the allocation of hardware resources, we shared the use of bias neurons among multiple sampling neurons (connected via distinct synapses). Similarly, in order to allow sign switches during training, connections between sampling neurons were implemented by pairs of synapses (one excitatory and one inhibitory) as well.

The dynamics of the sampling neurons were rendered stochastic in two different ways. The first setup served as a benchmark and represented a straightforward implementation of the theoretical model from (Petrovici et al., 2016), with Poisson noise generated on the host computer and fed in during the experiment (**Figure 3A**). In the second setup, we used the spiking activity of a sparse recurrent random network (RN) of inhibitory neurons, instantiated on the same wafer, as a source of noise (**Figure 3B**). For a more detailed study of sampling-based Bayesian inference with noise generated by deterministic networks, we refer to (Jordan et al., 2017). The mutual inhibition ensured a relatively constant (sub)population firing rate with suitable random statistics that can replace the ideal Poisson noise in our application. Projections from the RN to the SSN were chosen as random and sparse; this resulted in weak, but non-zero shared-input correlations. The remaining correlations are compensated by appropriate training; the Hebbian learning rule (Equation 1) changes the weights and biases in the network such that they cancel the input correlations induced by the RN activity (Bytschok et al., 2017; Dold et al., 2019). Hence, the same plasticity rule simultaneously addresses three issues: the learning procedure itself, the compensation of analog variability in neuronal excitability, and the compensation of cross-correlations in the input coming from the background network. This allowed the hardware-emulated RN to replace the Poisson noise required by the theoretical model.

With these noise-generating mechanisms, the activation function of the neurons, defined by the firing rate as a function of the bias weight w_b, took on an approximately logistic shape, as required by the sampling model (**Figure 3C**). Due mainly to the variability of the hardware circuits and the precision of the analog parameters, the exact shape of this activation function varied significantly between neurons (**Figures 3D,E**). Effectively, this means that initial weights and biases were set randomly, but also that the effective learning rates were different for each neuron. However, as we show below, this did not prevent the training procedure from converging to a good solution. This robustness, with respect to substrate variability, represents an important result of this work. The used neuron parameters are shown in **Table 2** and a summary of the used networks is given in **Table 3**. Our largest experiment, a network of 609 neurons with 208 sampling neurons, one bias neuron and 400 neurons in the RN (**Table 3C**) used hardware resources on 28 HICANN chips distributed over seven reticles. Each of these functional neurons was realized by combining four of the 512 neuronal compartments ("denmems") available on each HICANN, in order to reduce variability in their leak potentials and membrane time constants; for details see (Schemmel et al., 2010).

To train the networks on a neuromorphic substrate without embedded plasticity, we used a training concept often referred to as in-the-loop training (Schmuker et al., 2014; Esser et al., 2016; Schmitt et al., 2017). With the setup discussed above, the only parameters changed during training were digital, namely the synaptic weights between sampling neurons and the weights between bias and sampling neurons. This allowed us to work with a fixed set of analog parameters, which significantly amplified the precision and speed of reconfiguration during learning, as compared to having used the analog storage instead. The updates of the digital parameters (synaptic weights) were calculated on the host computer based on the wake-sleep algorithm (Equation 1) but using the spiking activity measured on the hardware. During the iterative procedure, the values of the weights were saved and updated as a double precision floating point variable, followed by (deterministic) discretization in order to comply with the single-synapse weight resolution of 4 bits. The learning parameters are given in **Table 4**. Clamping (i.e., forcing neurons into state 1 or 0 with strong excitatory or inhibitory input) was done by injecting regular spike trains with a 100 Hz frequency from the host through five synapses simultaneously, excitatory for $z_k = 1$ and inhibitory for $z_k = 0$. These multapses (multiple synapses connecting two neurons) were needed to

TABLE 2 | Neuron parameters.

(A) Sampling neuron

Name	Value	Description
V_{reset}	−35 mV	Reset potential
E_{leak}	−20 mV	Resting potential
V_{thresh}	−20 mV	Threshold potential
E_{inh}	−100 mV	Inhibitory reversal potential
E_{exc}	60 mV	Excitatory reversal potential
τ_{ref}	4 ms	Refractory time
τ_{mem}	ca. 7 ms	Membrane time constant*
C_{mem}	0.2 nF	Membrane capacity
$\tau_{\text{syn}}^{\text{exc}}$	8 ms	Excitatory synaptic time constant
$\tau_{\text{syn}}^{\text{inh}}$	8 ms	Inhibitory synaptic time constant

(B) Bias neuron

Name	Value	Description
V_{reset}	−30 mV	Reset potential
E_{leak}	60 mV	Resting potential
V_{thresh}	−20 mV	Threshold potential
E_{inh}	−100 mV	Inhibitory reversal potential
E_{exc}	60 mV	Excitatory reversal potential
τ_{ref}	1.5 ms	Refractory time
τ_{mem}	ca. 7 ms	Membrane time constant*
C_{mem}	0.2 nF	Membrane capacity
$\tau_{\text{syn}}^{\text{exc}}$	5 ms	Excitatory synaptic time constant
$\tau_{\text{syn}}^{\text{inh}}$	5 ms	Inhibitory synaptic time constant

(C) Neurons of the random network

Name	Value	Description (all analog)
V_{reset}	−60 mV	Reset potential
E_{leak}	−10 mV	Resting potential
V_{thresh}	−20 mV	Threshold potential
E_{inh}	−100 mV	Inhibitory reversal potential
E_{exc}	60 mV	Excitatory reversal potential
τ_{ref}	4 ms	Refractory time
τ_{mem}	ca. 7 ms	Membrane time constant*
C_{mem}	0.2 nF	Membrane capacity
$\tau_{\text{syn}}^{\text{exc}}$	8 ms	Excitatory synaptic time constant
$\tau_{\text{syn}}^{\text{inh}}$	8 ms	Inhibitory synaptic time constant

(D) Synapse

Name	Value	Description
w_{bias}	[0,15]	Synaptic bias weight in hardware values (digital)
w_{network}	[0,15]	Synaptic network weight in hardware values (digital)
d	On the order of 1 ms (uncalibrated)	Synaptic delay, estimated in Schemmel et al. (2010)

*Parameters of the network setup specified in **Table 1**. The analog parameters are shown as specified in the software setup and not as realized on the hardware. For details on the calibration procedure see e.g., Schmitt et al. (2017). *The calibration of the membrane time constant was not available at the time of this work, and the corresponding technical parameter was set to the smallest available value instead (fastest possible membrane dynamics for each neuron).*

TABLE 3 | Network parameters.

(A) Probability distribution with poisson noise

Name	Value	Description
N_s	5	Number of sampling neurons
N_b	1	Number of bias neurons
N_r	0	Number of random neurons
K_RN	–	Within-population in-degree of neurons in the random network
K_noise	–	In-degree of sampling neurons from the random network
w_RN	–	Synaptic weights in the random network in hardware units
$\nu^{\mathrm{e/i}}_{\mathrm{Poisson}}$	300Hz	Poisson frequency to sampling neurons per synapse type

(B) Probability distribution with random network

Name	Value	Description
N_s	5	Number of sampling neurons
N_b	1	Number of bias neurons
N_r	200	Number of random neurons
K_RN	20	Within-population in-degree of neurons in the random network
K_noise	15	In-degree of sampling neurons from the random network
w_RN	10	Synaptic weights in the random network in hardware units
$\nu^{\mathrm{e/i}}_{\mathrm{Poisson}}$	–	Poisson frequency to sampling neurons per synapse type

(C) High-dimensional dataset

Name	Value	Description
N_s	$\{207, 208\}$	Number of sampling neurons, { rFMNIST, rMNIST }
N_b	1	Number of bias neurons
N_r	400	Number of random neurons
K_RN	20	Within-population in-degree of neurons in the random network
K_noise	15	In-degree of sampling neurons from the random network
w_RN	10	Synaptic weights in the random network in hardware units
$\nu^{\mathrm{e/i}}_{\mathrm{Poisson}}$	–	Poisson frequency to sampling neurons per synapse type

Parameters are shown for the three different cases described in the manuscript: ***(A)*** *Target Boltzmann distribution, Poisson noise.* ***(B)*** *Target Boltzmann distribution, random network for stochasticity.* ***(C)*** *Learning from data, random network for stochasticity. Note that the in-degree, sometimes also referred to as a fan-in factor, represents a neuron's number of pre-synaptic partners coming from some specific population.*

exceed the upper limit of single synaptic weights and thus ensure proper clamping.

3. RESULTS

3.1. Learning to Approximate a Target Distribution

The experiments described in this section serve as a general benchmark for the ability of our hardware-emulated SSNs and the associated training algorithm to approximate fully specified target Boltzmann distributions. The viability of our proposal to simultaneously embed deterministic RNs as sources of pseudo-stochasticity is tested by comparing the sampling accuracy of RN-driven SSNs to the case where noise is injected from the host as perfectly uncorrelated Poisson spike trains.

TABLE 4 | Parameters for learning.

Experiment	Learning rate	Momentum factor	Minibatch-size	Initial (W, b)
Target distribution, Poisson	1.0	0.6	–	$\mathcal{U}(-15, 15)$
Target distribution, random network	0.5	0.6	–	$\mathcal{U}(-15, 15)$
rMNIST	0.4	0.6	7/class	Pre-trained
rFMNIST	0.4	0.6	7/class	Pre-trained

We did not carry any systematic hyper-parameter optimization. Note that the used learning parameters in the experiments in section 3.1 are not directly comparable because the different statistics of the background noise (Poisson or random network) correspond to different effective learning rates.

Target distributions p^* over 5 RVs were chosen by sampling weights and biases from a Beta distribution centered around zero: $b_i, w_{ji} \sim 2[\mathrm{Beta}(0.5, 0.5) - 0.5]$. Similar to previous studies (Petrovici et al., 2016; Jordan et al., 2017), by giving preference to larger absolute values of the target distribution's parameters, we thereby increased the probability of instantiating rougher, more interesting energy landscapes. The initial weights and biases of the network were sampled from a uniform distribution over the possible hardware weights. Due to the small size of the state space, the "wake" component of the wake-sleep updates could be calculated analytically as $\langle z_i z_j \rangle = p^*(z_i = 1, z_j = 1)$ and $\langle z_i \rangle = p^*(z_i = 1)$ by explicit marginalization of the target distribution over non-relevant RVs.

For training, we used 500 iterations with 1×10^5 ms sampling time per iteration. Afterwards, the parameter configuration that produced the lowest $D_{\mathrm{KL}}(p \parallel p^*)$ was tested in a longer (5×10^5 ms) experiment. To study the ability of the trained networks to perform Bayesian inference, we clamped two of the five neurons to fixed values $(z_1, z_2) = (0, 1)$ and compared the sampled conditional distribution to the target conditional distribution. Results for one of these target distributions are shown in **Figure 4**.

On average, with Poisson noise, the training showed fast convergence during the first 20 iterations, followed by fine-tuning and full convergence within 200 iterations. As expected, the convergence of the setups using RNs was significantly slower due to the need to overcome the additional background correlations, but they were still able to achieve similar performance (**Figure 4A**).

In both setups, during the test run, the trained SSNs converged to the target distribution following an almost identical power law, which indicates similar mixing properties (**Figure 4B**). For longer sampling durations ($\gg 10 \times 10^3$ ms), the systematic deviations from the target distributions become visible and the $D_{\mathrm{KL}}(p \parallel p^*)$ reaches the same plateau at approximately $D_{\mathrm{KL}}(p \parallel p^*) \approx 2 \times 10^{-2}$ as observed during training. **Figures 4C,D**, respectively show the sampled joint and marginal distributions after convergence (**Supplementary Video 1**). These observations remained consistent across a set of 20 different target distributions (see **Figure 4E** for a representative selection).

Similar observations were made for the inference experiments. Due to the smaller state space, convergence happened faster

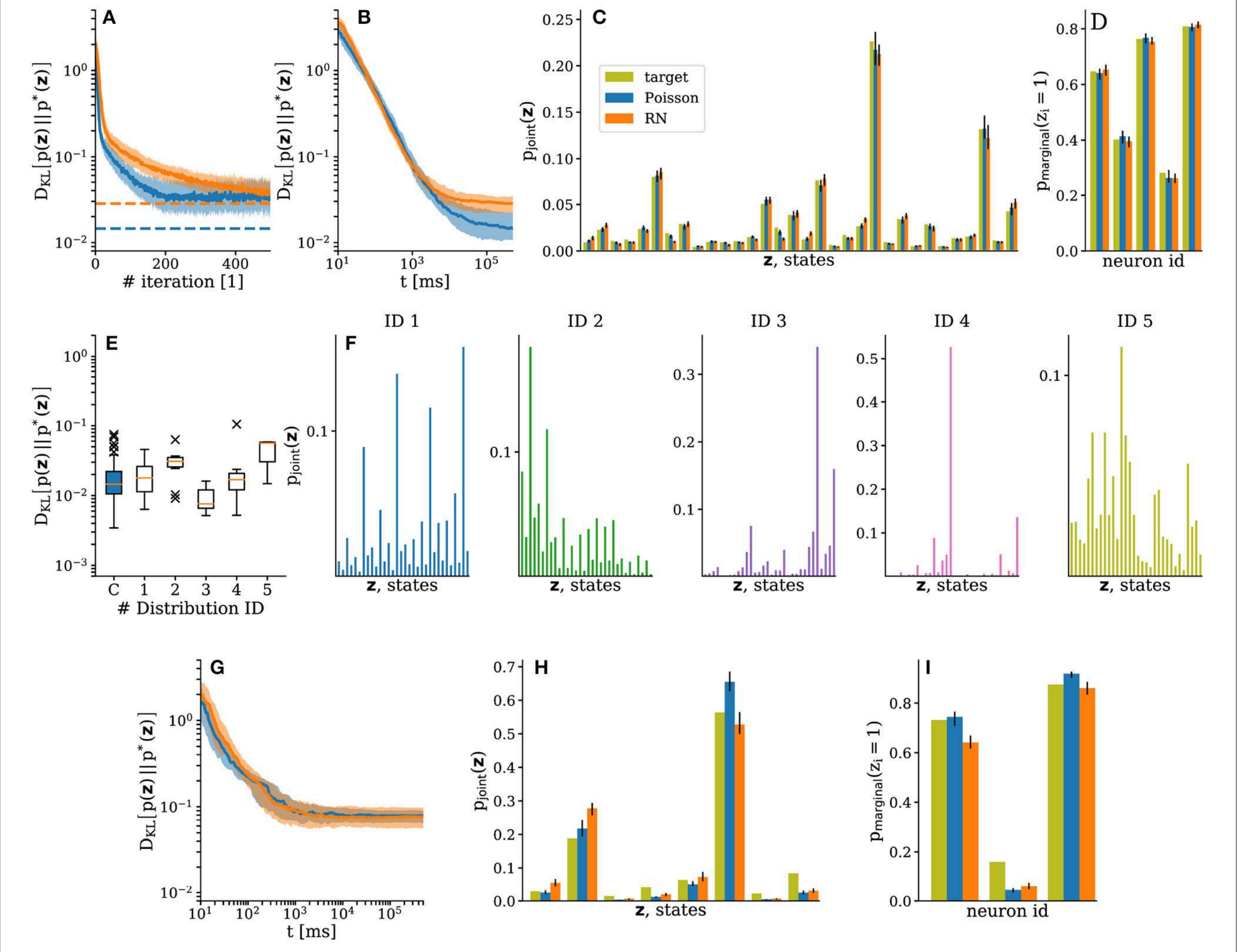


FIGURE 4 | Emulated SSNs sampling from target Boltzmann distributions. Sampled distributions are depicted in blue for setups with Poisson noise and in orange for setups using RNs. Target distributions shown in dark yellow. Data was gathered from 150 runs with random initializations. Median values are shown as dark colors and interquartile ranges as either light colors or error bars. **(A)** Improvement of sampled distributions during training. The observed variability after convergence (during the plateau) is not due to noise in the system, but rather a consequence of the weight discretization: when the ideal (target) weights lie approximately mid-way between two consecutive integer values on the hardware, training leads to oscillations between these values. The parameter configuration showing the best performance during a training run—which, due to the abovementioned oscillations, was not necessarily the one in the final iteration—was chosen as the end result of the training phase. Averages of these results are shown as dashed lines. **(B)** Convergence of sampled distributions for the trained SSNs. **(C,D)** Sampled joint and marginal distributions of the trained SSNs after 5×10^5 ms, respectively. **(E)** Consistency of training results for different target distributions using Poisson noise. Here, we show a representative selection of 6 distributions with 10 independent runs per distribution. The box highlighted in blue corresponds to the target distribution used in the other panels of **Figure 4**. The data is plotted following the traditional box-and-whiskers scheme: the orange line represents the median, the box represents the interquartile range, the whiskers represent the full data range and the × represent the far outliers. **(F)** Target distributions corresponding to the last five box-and-whiskers plots in **(E)**. **(G)** Convergence of conditional distributions for the trained SSNs. **(H)** and **(I)** Sampled conditional joint and marginal distributions of the trained SSNs after 5×10^5 ms, respectively.

(**Figure 4G**). The corresponding joint and marginal distributions are shown in **Figures 4H,I**, respectively. The lower accuracy of these distributions is mainly due to the asymmetry of the effective synaptic weights caused by the variability of the substrate, toward which the learning algorithm is agnostic. The training took 5×10^2 s wall-clock time, including the pure experiment runtime, the initialization of the hardware and the calculation of the updates on the host computer (total turn-over time of the training). This corresponds to a speed-up factor of 100 compared to the equivalent 5×10^4 s of biological real time. While the nominal 10^4 speed-up remained intact for the emulation of network dynamics, the total speed-up factor was reduced due to the overhead imposed by network (re)configuration and I/O between the host and the neuromorphic substrate.

We carried out the same experiments as described previously with 20 different samples for the weights and the biases of the

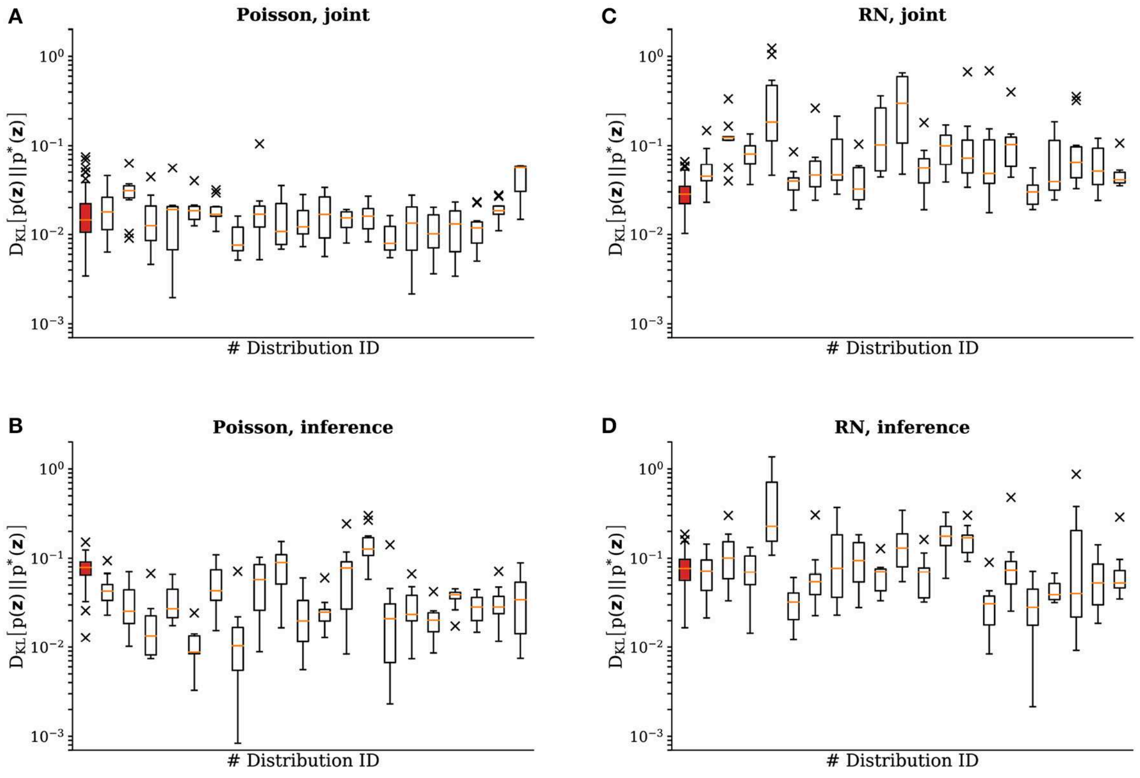


FIGURE 5 | Emulated SSNs sampling from different target Boltzmann distributions. This figure shows the results of experiments identical to the ones in section 3.1 for 20 different target distributions with 10 repetitions for each sample. We show the $D_{KL}(p \parallel p^*)$ of the test-run after training for **(A)** the joint distributions with Poisson noise, **(B)** the inference experiment with Poisson noise, **(C)** the joint distributions with a random background network and **(D)** the inference experiment with a random background network. The data is plotted following the traditional box-and-whiskers scheme: the orange line represents the median, the box represents the interquartile range, the whiskers represent the full data range and the × represent the far outliers. In each subplot the leftmost data (highlighted in red) corresponds to the distribution shown in **Figure 4**.

target distribution. In **Figure 5** we show the final DKLs after training to represent a target distribution both with Poisson noise and with the activity of a random network. The experiments were repeated 10 times for each sample. Median learning results remained consistent across target distributions, with the variability reflecting the difficulty of the problem (discrepancies between LIF and Glauber dynamics become more pronounced for larger weights and biases). Variability across trials for the same target distribution is due to the trial-to-trial variability of the analog parameter storage (floating gates), due to the inherent stochasticity in the learning procedure (sampling accuracy in an update step), as well as due to systematic discrepancies between the effective pre-post and post-pre interaction strengths between sampling units, which are themselves a consequence of the aforementioned floating gate variability.

3.2. Learning From Data

In order to obtain models of labeled data, we trained hierarchical SSNs analogously to restricted Boltzmann machines (RBMs). Here, we used two different datasets: a reduced version of the MNIST (LeCun et al., 1998) and the fashion MNIST (Xiao et al., 2017) datasets, which we abbreviate as rMNIST and rFMNIST in the following. The images were first reduced with nearest-neighbor resampling [`misc.imresize` function in the SciPy library (Jones et al., 2001)] and then binarized around the median gray value over each image. We used all images from the original datasets (~6,000 per class) from four classes (0, 1, 4, 7) for rMNIST and three classes (**T**-shirts, **Tr**ousers, **S**neakers) for rFMNIST (**Figures 6A,B**). The emulated SSNs consisted of three layers, with 144 visible, 60 hidden, and either four label units for rMNIST or three for rFMNIST.

Pre-training was done on simulated classical RBMs using the CAST algorithm (Salakhutdinov, 2010). The pre-training provided a starting point for training on the hardware in order to accelerate the convergence of the in-the-loop training procedure. We use the performance of these RBMs in software simulations using Gibbs sampling as a reference for the results obtained with the hardware-emulated SSNs. After pre-training, we mapped these RBMs to approximately equivalent SSNs on the hardware, using an empirical translation factor based on an average activation function (**Figure 3C**) to calculate the initial hardware synaptic weights from weights and biases of the RBMs. Especially for rMNIST, this resulted in a significant deterioration of the classification performance (**Figure 6C**). After mapping, we continued training using the wake-sleep algorithm, with the

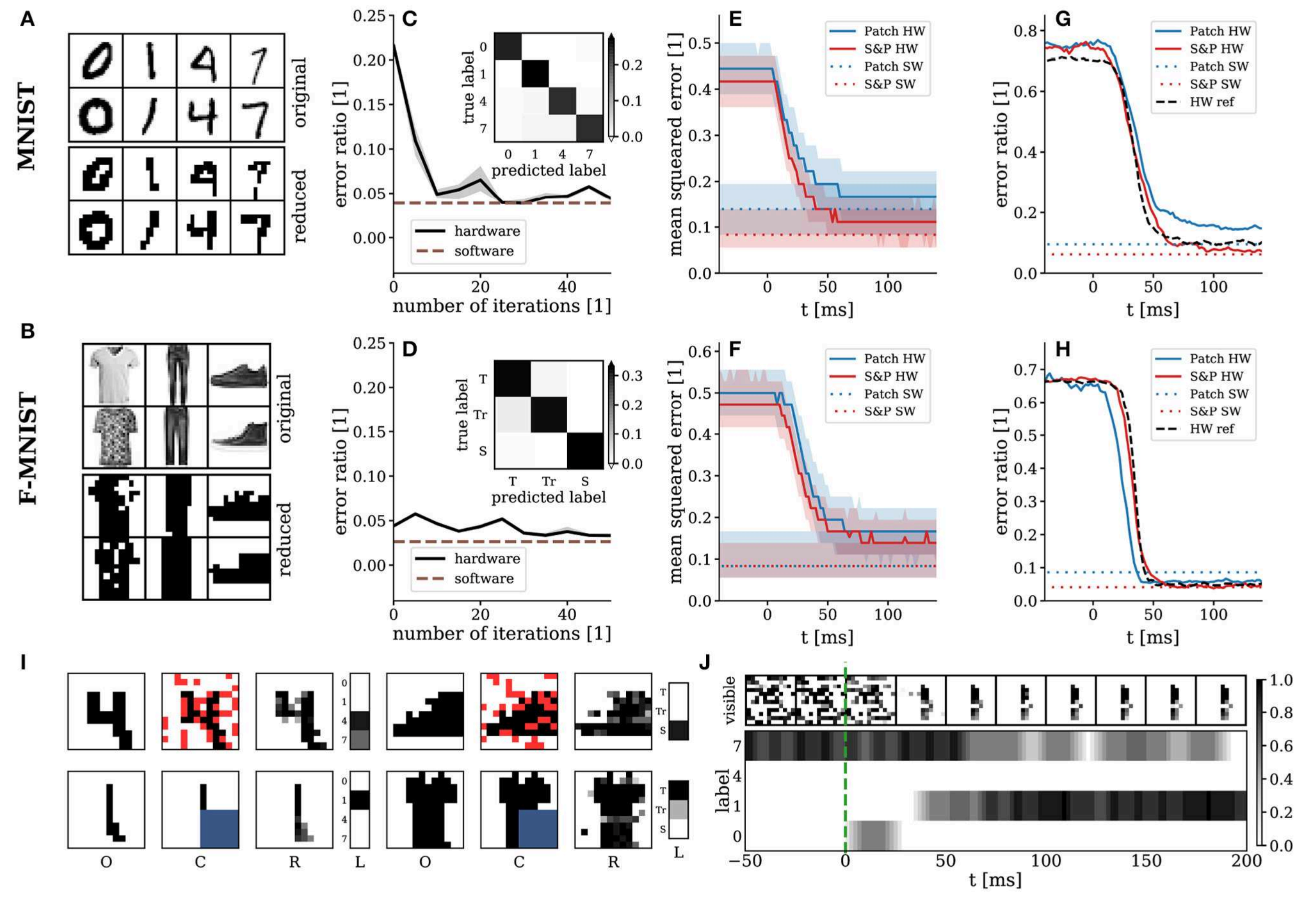


FIGURE 6 | Behavior of hierarchical SSNs trained on data. Top row: rMNIST; middle row: rFMNIST; bottom row: exemplary setups for the partial occlusion scenarios. **(A,B)** Exemplary images from the rMNIST **(A)** and rFMNIST **(B)** datasets used for training and comparison to their MNIST and FMNIST originals. **(C,D)** Training with the hardware in the loop after translation of pre-trained parameters. Confusion matrices after training shown as insets. Performance of the reference RBMs shown as dashed brown lines. Results are given as median and interquartile values over 10 test runs. **(E,F)** Pattern completion and **(G,H)** error ratio of the inferred label for partially occluded images (blue: patch; red: salt&pepper). Solid lines represent median values and shaded areas show interquartile ranges over 250 test images per class. Performance of the reference RBMs shown as dashed lines. As a reference, we also show the error ratio of the SNNs on unconcluded images in **(G)** and **(H)**. **(I)** Snapshots of the pattern completion experiments: O—original image, C—clamped image (red and blue pixels are occluded), R—response of the visible layer, L—response of the label layer. **(J)** Exemplary temporal evolution of a pattern completion experiment with patch occlusion. For better visualization of the activity in the visible layer in **(I,J)**, we smoothed out its discretized response to obtain grayscale pixel values, by convolving its state vector with a box filter of 10 ms width.

hardware in the loop. While in the previous task it was possible to calculate the data term explicitly, it now had to be sampled as well. In order to ensure proper clamping, the synapses from the hidden to the label layer and from the hidden layer to the visible layer were turned off during the wake phase.

The SSNs were tested for both their discriminative and their generative properties. For classification, the visible layer was clamped to images from the test set (black pixels correspond to $z_k = 1$ and white pixels to $z_k = 0$). Each image was presented for 500 biological milliseconds, which corresponds to 50 μs wall-clock time. The neuron in the label layer with the highest firing rate was interpreted as the label predicted by the model. The spiking activity of the neurons was read out directly from the hardware, without additional off-chip post-processing. For both datasets, training was able to restore the performance lost in the translation of the abstract RBM to the hardware-emulated SSN. The emulated SSNs achieved error rates of $4.45^{+0.12}_{-0.36}$% on rMNIST and $3.32^{+0.27}_{-0.04}$% on rFMNIST. These values are close to the ones obtained by the reference RBMs: $3.89^{+0.10}_{-0.02}$% on rMNIST and $2.645^{+0.002}_{-0.010}$% on rFMNIST (**Figures 6C,D**, confusion matrices shown as insets).

The gross wall-clock time needed to classify the 4125 images in the rMNIST test set was 10 s (2.4 ms per image, 210× speed-up). For the 3,000 images in the rFMNIST test set, the emulation ran for 9.4 s (3.1 ms per image; 160× speed-up). This subsumes the runtime of the BrainScaleS software stack, hardware configuration and the network emulation. The runtime of the software-stack includes the translation from a PyNN-based network description to a corresponding hardware configuration. As before, the difference between the nominal acceleration factor and the effective speed-up stems from the I/O and initialization overhead of the hardware system.

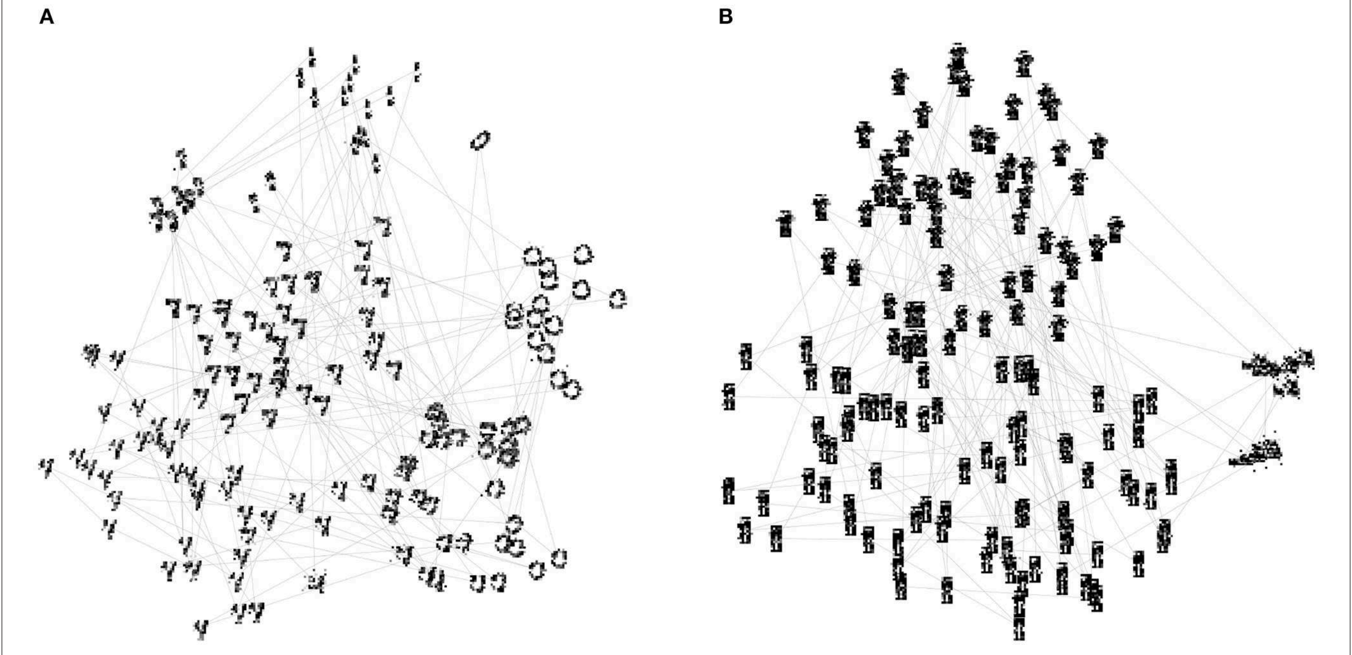


FIGURE 7 | Generated images during guided dreaming. The visible state space, along with the position of the generated images within it, was projected to two dimensions using t-SNE (Maaten and Hinton, 2008). The thin lines connect consecutive samples. **(A)** rMNIST; **(B)** rFMNIST.

To test the generative properties of our emulated SSNs, we set up two scenarios requiring them to perform pattern completion (**Supplementary Video 2**). For each class, 250 incomplete images were presented as inputs to the visible layer. For each image, 25 % of visible neurons received no input, with the occlusion following two different schemes: salt&pepper (upper row in **Figure 6I**) and patch (lower row in **Figure 6I**). Each image was presented for 500 ms. In order to remove any initialization bias resulting from preceding images, random input was applied to the visible layer between consecutive images.

Reconstruction accuracy was measured using the mean squared error (MSE) between the reconstructed and original occluded pixels. For binary images, as in our case, the MSE reflects the average ratio of mis-reconstructed to total reconstructed pixels. Simultaneously, we also recorded the classification accuracy on the partially occluded images. After stimulus onset, the MSE converged from chance level ($\approx$ 50 %) to its minimum ($\approx$ 10 %) within 50 ms (**Figures 6E,F**). Given an average refractory period of $\approx$ 10 ms (**Figure 3C**), this suggests that the network was able to react to the input with no more than 5 spikes per neuron. For all studied scenarios, the reconstruction performance of the emulated SSNs closely matched the one achieved by the reference RBMs. Examples of image reconstruction are shown in **Figures 6I,J** for both datasets and occlusion scenarios. The classification performance deteriorated only slightly compared to non-occluded images and also remained close to the performance of the reference RBMs (**Figures 6G,H**). The temporal evolution of the classification error closely followed that of the MSE.

As a further test of the generative abilities of our hardware-emulated SSNs, we recorded the images produced by the visible layer during guided dreaming. In this task, the visible and hidden layers of the SSN evolved freely without external input, while the label layer was periodically clamped with external input such that exactly one of the label neurons was active at any time (enforced one-hot coding). In a perfect model, this would cause the visible layer to sample only from configurations compatible with the hidden layer, i.e., from images corresponding to that particular class. Between the clamping of consecutive labels, we injected 100 ms random input to visible layer to facilitate the changing of the image. The SSNs were able to generate varied and recognizable pictures, within the limits imposed by the low resolution of the visible layer (**Figure 7**). For rMNIST, all used classes appeared in correct correspondence to the clamped label. For rFMNIST, images from the class "Sneakers" were not always triggered by the corresponding guidance from the label layer, suggesting that the learned modes in the energy landscape are too deep, and sneakers too dissimilar to T-shirts and Trousers, to allow good mixing during guided dreaming.

4. DISCUSSION

This article presents the first scalable demonstration of sampling-based probabilistic inference with spiking networks on a highly accelerated analog neuromorphic substrate. We trained fully connected spiking networks to sample from target distributions and hierarchical spiking networks as discriminative and generative models of higher-dimensional input data. Despite the inherent variability of the analog substrate, we were able to achieve performance levels comparable to those of software simulations in several benchmark tasks, while maintaining a significant overall acceleration factor compared to systems that operate in biological real time. Importantly, by co-embedding the generation of stochasticity within the same substrate, we demonstrated the viability of a fully embedded neural sampling model with significantly reduced demands on off-substrate I/O bandwidth. Having a fully embedded implementation allows the runtime of the experiments to scale as $\mathcal{O}(1)$ with the size of the emulated network; this is inherent to the nature of physical emulation, for which wall-clock runtime only depends on the emulated time in the biological reference frame. In the following sections, we address the limitations of our study, point out links to related work and discuss its implications within the greater context of computational neuroscience and bio-inspired AI.

4.1. Limitations and Constraints

The most notable limitation imposed by the current commissioning state of the BrainScaleS system was on the size of the emulated SSNs. At the time of writing, due to limited software flexibility, system assembly and substrate yield, the usable hardware real-estate was reduced to a patchy and non-contiguous area, thereby strongly limiting the maximum connectivity between different locations within this area. In order to limit synapse loss to small values (below 2 %), we restricted ourselves to using a small but contiguous functioning area of the wafer, which in turn limited the maximum size of our SSNs and noise-generating RNs. Ongoing improvements in post-production and assembly, as well as in the mapping and routing software, are expected to enhance on-wafer connectivity and thereby automatically increase the size of emulable networks, as the architecture of our SSNs scales naturally to such an increase in hardware resources.

To a lesser extent, the sampling accuracy was also affected by the limited precision of hardware parameter control. The writing of analog parameters exhibits significant trial-to-trial variability; in any given trial, this leads to a heterogeneous substrate, which is known to reduce the sampling accuracy (Probst et al., 2015). Most of this variability is compensated during learning, but the 4 bit resolution of the synaptic weights and the imperfect symmetry in the effective weight matrix due to analog variability of the synaptic circuits ultimately limit the ability of the SSN to approximate target distributions. This leads to the "jumping" behavior of the $D_{\mathrm{KL}}(p \parallel p^*)$ in the final stages of learning (**Figure 4A**). In smaller networks, synaptic weight resolution is a critical performance modifier (Petrovici et al., 2017b). However, the penalty imposed by a limited synaptic weight resolution is known to decrease for larger deep networks with more and larger hidden layers, both spiking and non-spiking (Courbariaux et al., 2015; Petrovici et al., 2017a). Furthermore, the successor system (BrainScaleS-2, Aamir et al., 2016) is designed with a 6-bit weight resolution.

In the setup we used shared bias neurons for several neurons in the sampling network. This helped us save hardware resources, thus allowing the emulation of larger functional networks. Such bias neuron sharing is expected to introduce some small amount of temporal correlations between the sampling neurons. However, this effect was too small to observe in our experiments for several reasons. First, the high firing rate of the bias neurons helped smooth out the bias voltage induced into the sampling neurons. Second, the different delays and spike timing jitter on the hardware reduces such cross-correlations. Third, other dominant limitations overshadow the effect of shared bias neurons. In any case, the used training procedure inherently compensates for excess cross-correlations, thus effectively removing any distortions to the target distribution that this effect might introduce (Bytschok et al., 2017; Dold et al., 2019).

In the current setup, our SSNs displayed limited mixing abilities. During guided dreaming, images from one of the learned classes were more difficult to generate (**Figure 7**). Restricted mixing due to deep modes in the energy landscape carved out by contrastive learning is a well-known problem for classical Boltzmann machines, which is usually alleviated by computationally costly annealing techniques (Desjardins et al., 2010; Salakhutdinov, 2010; Bengio et al., 2013). However, the fully-commissioned BrainScaleS system will feature embedded short-term synaptic plasticity (Schemmel et al., 2010), which has been shown to promote mixing in spiking networks (Leng et al., 2018) while operating purely locally, at the level of individual synapses.

Currently, the execution speed of emulation runs is dominated by the I/O overhead, which in turn is mostly spent on setting up the experiment. This leads to the classification (section 3.2) of one image taking 2.4 to 3.9 ms, whereas the pure network runtime is merely 50 μs. A streamlining of the software layer that performs this setup is expected to significantly reduce this discrepancy.

The synaptic learning rule was local and Hebbian, but updates were calculated on a host computer using an iterative in-the-loop training procedure, which required repeated stopping, evaluation and restart of the emulation, thereby reducing the nominal acceleration factor of 10^4 by two orders of magnitude. By utilizing on-chip plasticity, as available, for example, on the BrainScaleS-2 successor system (Friedmann et al., 2017; Wunderlich et al., 2019), this laborious procedure becomes obsolete and the accelerated nature of the substrate can be exploited to its fullest extent.

4.2. Relation to Other Work

This study builds upon a series of theoretical and experimental studies of sampling-based probabilistic inference using the dynamics of biological neurons. The inclusion of refractory times was first considered in Buesing et al. (2011). An extension to networks of leaky integrate-and-fire neurons and a theoretical framework for their dynamics and statistics followed in Petrovici et al. (2013) and Petrovici et al. (2016). The compensation of shared-input correlations through inhibitory feedback and learning was discussed in Bytschok et al. (2017), Jordan et al. (2017), and Dold et al. (2019), inspired by the early study of asynchronous irregular firing in Brunel (2000) and by preceding correlation studies in theoretical (Tetzlaff et al., 2012) and experimental (Pfeil et al., 2016) work.

Previous small-scale studies of sampling on accelerated mixed-signal neuromorphic hardware include (Petrovici et al., 2015, 2017a,b). An implementation of sampling with spiking neurons and its application to the MNIST dataset was shown in

Pedroni et al. (2016) using the fully digital, real-time TrueNorth neuromorphic chip (Merolla et al., 2014).

We stress two important differences between (Pedroni et al., 2016) and this work. First, the nature of the neuromorphic substrate: the TrueNorth system is fully digital and calculates neuronal state updates numerically, in contrast to the physical-model paradigm instantiated by BrainScaleS. In this sense, TrueNorth emulations are significantly closer to classical computer simulations on parallel machines: updates of dynamical variables are precise and robustness to variability is not an issue; however TrueNorth typically runs in biological real time (Merolla et al., 2014; Akopyan et al., 2015), which is 10,000 times slower than BrainScaleS. Second, the nature of neuron dynamics: the neuron model used in (Pedroni et al., 2016) is an intrinsically stochastic unit that sums its weighted inputs, thus remaining very close to classical Gibbs sampling and Boltzmann machines, while our approach considers multiple additional aspects of its biological archetype (exponential synaptic kernels, leaky membranes, deterministic firing, stochasticity through synaptic background, shared-input correlations etc.). Moreover, our approach uses fewer hardware neuron units to represent a sampling unit, enabling a more parsimonious utilization of the neuromorphic substrate.

4.3. Conclusion

In this work we showed how sampling-based Bayesian inference using hierarchical spiking networks can be robustly implemented on a physical model system despite inherent variability and imperfections. Underlying neuron and synapse dynamics are deterministic and close to their biological archetypes, but with much shorter time constants, hence the intrinsic acceleration factor of 10^4 with respect to biology. The entire architecture—sampling network plus background random network—was fully deterministic and entirely contained on the neuromorphic substrate, with external communication used only to represent input patterns and labels. Considering the deterministic nature of neurons *in vitro* (Mainen and Sejnowski, 1995; Reinagel and Reid, 2002; Toups et al., 2012), such an architecture also represents a plausible model for neural sampling in cortex (Jordan et al., 2017; Dold et al., 2019).

We demonstrated sampling from arbitrary Boltzmann distributions over binary random variables, as well as generative and discriminative properties of networks trained with visual data. The framework can be extended to sampling from arbitrary probability distributions over binary random variables, as it was shown in software simulations (Probst et al., 2015). For such networks, the two abovementioned computational tasks (pattern completion and classification) happen simultaneously, as they both require the calculation of conditional distributions, which is carried out implicitly by the network dynamics. Both during learning and for the subsequent inference tasks, the setup benefitted significantly from the fast-intrinsic dynamics of the substrate, achieving a net speedup of 100–210 compared to biology.

We view these results as a contribution to the nascent but expanding field of applications for biologically inspired physical-model systems. They demonstrate the feasibility of such devices to solve problems in machine learning, as well as studying biological phenomena. Importantly, they explicitly address the search for robust computational models that are able to harness the strengths of these systems, most importantly their speed and energy efficiency. The proposed architecture scales naturally to substrates with more neuronal real-estate and can be used for a wide array of tasks that can be mapped to a Bayesian formulation, such as constraint satisfaction problems (Jonke et al., 2016; Fonseca Guerra and Furber, 2017), prediction of temporal sequences (Sutskever and Hinton, 2007), movement planning (Taylor and Hinton, 2009; Alemi et al., 2015), simulation of solid-state systems (Edwards and Anderson, 1975), and quantum many-body problems (Carleo and Troyer, 2017; Czischek et al., 2018).

DATA AVAILABILITY STATEMENT

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

AUTHOR CONTRIBUTIONS

AFK, AB, DD, LL, SS, PM, and MP designed the study. AFK conducted the experiments and the evaluations. NG contributed to the evaluations and provided software support for the evaluation. AFK wrote the initial manuscript. EM, CM, JK, SS, KH, and OB supported the experiment realization. EM coordinated the software development for the neuromorphic systems. AK, CK, and MK contributed with the characterization, calibration testing, and debugging of the system. AG, DH, and MG were responsible for system assembly. AG did the digital front- and back-end implementation. VK provided the FPGA firmware and supported system commissioning. JS is the architect and lead designer of the neuromorphic platform. MP, KM, JS, SS, and EM provided the conceptual and scientific advice. All authors contributed to the final manuscript.

FUNDING

The work leading to these results received funding from the European Union Seventh Framework Programme (FP7) under grant agreement No. #604102, the EU's Horizon 2020 research and innovation programme under grant agreements No. #720270 and #785907 (Human Brain Project, HBP), the EU's research project BrainScaleS #269921 and the Heidelberg Graduate School of Fundamental Physics.

ACKNOWLEDGMENTS

We thank Johannes Bill for many fruitful discussions. We acknowledge financial support by Deutsche Forschungsgemeinschaft within the funding programme

Open Access Publishing, by the Baden-Württemberg Ministry of Science, Research and the Arts and by Ruprecht-Karls-Universität Heidelberg. We owe particular gratitude to the sustained support of our research by the Manfred Stärk Foundation.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2019.01201/full#supplementary-material

REFERENCES

Aamir, S. A., Müller, P., Hartel, A., Schemmel, J., and Meier, K. (2016). "A highly tunable 65-nm cmos lif neuron for a large scale neuromorphic system," in *42nd European Solid-State Circuits Conference, ESSCIRC Conference 2016* (Lausanne: IEEE), 71–74.

Aitchison, L., and Lengyel, M. (2016). The hamiltonian brain: efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics. *PLoS Comput. Biol.* 12:e1005186. doi: 10.1371/journal.pcbi.1005186

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., et al. (2015). Truenorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 34, 1537–1557. doi: 10.1109/TCAD.2015.2474396

Alemi, O., Li, W., and Pasquier, P. (2015). "Affect-expressive movement generation with factored conditional restricted Boltzmann machines," in *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)* (Xian: IEEE), 442–448.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2013). "Better mixing via deep representations," in *International Conference on Machine Learning* (Atlanta, GA), 552–560.

Berkes, P., Orbán, G., Lengyel, M., and Fiser, J. (2011). Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science* 331, 83–87. doi: 10.1126/science.1195870

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci.* 8, 183–208. doi: 10.1023/A:1008925309027

Buesing, L., Bill, J., Nessler, B., and Maass, W. (2011). Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput. Biol.* 7:e1002211. doi: 10.1371/journal.pcbi.1002211

Bytschok, I., Dold, D., Schemmel, J., Meier, K., and Petrovici, M. A. (2017). Spike-based probabilistic inference with correlated noise. *BMC Neurosci.* 18:200. doi: 10.1186/s12868-017-0372-1

Carleo, G., and Troyer, M. (2017). Solving the quantum many-body problem with artificial neural networks. *Science* 355, 602–606. doi: 10.1126/science.aag2302

Chang, Y. F., Fowler, B., Chen, Y. C., Zhou, F., Pan, C. H., Chang, T. C., et al. (2016). Demonstration of synaptic behaviors and resistive switching characterizations by proton exchange reactions in silicon oxide. *Sci. Rep.* 6:21268. doi: 10.1038/srep21268

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). "Binaryconnect: training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, eds C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Montreal, QC: Neural Information Processing Systems Foundation, Inc.), 3123–3131.

Czischek, S., Gärttner, M., and Gasenzer, T. (2018). Quenches near ising quantum criticality as a challenge for artificial neural networks. *Phys. Rev. B* 98:024311. doi: 10.1103/PhysRevB.98.024311

Davison, A. P., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., et al. (2009). Pynn: a common interface for neuronal network simulators. *Front. Neuroinformatics* 2:11. doi: 10.3389/neuro.11.011.2008

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. (2010). "Parallel tempering for training of restricted boltzmann machines," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Cambridge, MA: MIT Press), 145–152.

Destexhe, A., Rudolph, M., and Paré, D. (2003). The high-conductance state of neocortical neurons *in vivo*. *Nat. Rev. Neurosci.* 4:739. doi: 10.1038/nrn1198

Dold, D., Bytschok, I., Kungl, A. F., Baumbach, A., Breitwieser, O., Senn, W., et al. (2019). Stochasticity from function—why the bayesian brain may need no noise. *Neural Netw.* 119, 200–213. doi: 10.1016/j.neunet.2019.08.002

Edwards, S. F., and Anderson, P. W. (1975). Theory of spin glasses. *J. Phys. F Metal Phys.* 5:965.

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., et al. (2016). From the cover: convolutional networks for fast, energy-efficient neuromorphic computing. *Proc. Natl. Acad. Sci. U.S.A.* 113:11441. doi: 10.1073/pnas.1604850113

Fonseca Guerra, G. A., and Furber, S. B. (2017). Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems. *Front. Neurosci.* 11:714. doi: 10.3389/fnins.2017.00714

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., and Meier, K. (2017). Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Trans. Biomed. Circuits Syst.* 11, 128–142. doi: 10.1109/TBCAS.2016.2579164

Furber, S. (2016). Large-scale neuromorphic computing systems. *J. Neural Eng.* 13:051001. doi: 10.1088/1741-2560/13/5/051001

Haefner, R. M., Berkes, P., and Fiser, J. (2016). Perceptual decision-making as probabilistic inference by neural sampling. *Neuron* 90, 649–660. doi: 10.1016/j.neuron.2016.03.020

Hennequin, G., Aitchison, L., and Lengyel, M. (2014). Fast sampling for bayesian inference in neural circuits. *arXiv 1404.3521*.

Hinton, G. E. (2012). "A practical guide to training restricted boltzmann machines," in *Neural Networks: Tricks of the Trade*, eds G. Montavon, G. B. Orr, and K.-R. Müller (Berlin; Heidelberg: Springer), 599–619. doi: 10.1007/978-3-642-35289-8

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science* 268, 1158–1161.

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). *Boltzmann Machines: Constraint Satisfaction Networks That Learn*. Pittsburgh, PA: Carnegie-Mellon University, Department of Computer Science.

Indiveri, G., Chicca, E., and Douglas, R. J. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 1297–1301. doi: 10.1021/nl904092h

Jones, E., Oliphant, T., and Peterson, P. (2001). *SciPy: Open Source Scientific Tools for Python.*

Jonke, Z., Habenschuss, S., and Maass, W. (2016). Solving constraint satisfaction problems with networks of spiking neurons. *Front. Neurosci.* 10:118. doi: 10.3389/fnins.2016.00118

Jordan, J., Petrovici, M. A., Breitwieser, O., Schemmel, J., Meier, K., Diesmann, M., et al. (2017). Stochastic neural computation without noise. *arXiv 1710.04931*.

Kullback, S., and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Stat.* 22, 79–86.

Kutschireiter, A., Surace, S. C., Sprekeler, H., and Pfister, J.-P. (2017). Nonlinear bayesian filtering and learning: a neuronal dynamics for perception. *Sci. Rep.* 7:8722. doi: 10.1038/s41598-017-17246-9

Lande, T. S., Ranjbar, H., Ismail, M., and Berg, Y. (1996). "An analog floating-gate memory in a standard digital technology," in *Proceedings of Fifth International Conference on Microelectronics for Neural Networks* (Lausanne: IEEE), 271–276.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324.

Leng, L., Martel, R., Breitwieser, O., Bytschok, I., Senn, W., Schemmel, J., et al. (2018). Spiking neurons with short-term synaptic plasticity form superior generative networks. *Sci. Rep.* 8:10651. doi: 10.1038/s41598-018-28999-2

Loock, J.-P. (2006). *Evaluierung eines floating gate analogspeichers für neuronale netze in single-poly umc 180nm cmos-prozess* (Diploma thesis), University of Heidelberg, Heidelberg, Germany.

Maaten, L. V. D., and Hinton, G. (2008). Visualizing data using t-sne. *J. Mach. Learn. Res.* 9, 2579–2605. Available online at: http://www.jmlr.org/papers/v9/vandermaaten08a.html

Mainen, Z. F., and Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science* 268, 1503–1506.

Mead, C. (1990). Neuromorphic electronic systems. *Proc. IEEE* 78, 1629–1636.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Millner, S. (2012). *Development of a multi-compartment neuron model emulation* (PhD thesis) (Heidelberg: Heidelberg University, Faculty of Physics and Astronomy). doi: 10.11588/heidok.00013979

Millner, S., Grübl, A., Meier, K., Schemmel, J., and Schwartz, M.-O. (2010). A VLSI implementation of the adaptive exponential integrate-and-fire neuron model. *Adv. Neural Inform. Process. Syst.* (Vancouver, QC) 23, 1642–1650. Available online at: https://papers.nips.cc/paper/3995-a-vlsi-implementation-of-the-adaptive-exponential-integrate-and-fire-neuron-model

Orbán, G., Berkes, P., Fiser, J., and Lengyel, M. (2016). Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron* 92, 530–543. doi: 10.1016/j.neuron.2016.09.038

Pedroni, B. U., Das, S., Arthur, J. V., Merolla, P. A., Jackson, B. L., Modha, D. S., et al. (2016). Mapping generative models onto a network of digital spiking neurons. *IEEE Trans. Biomed. Circuits Syst.* 10, 837–854. doi: 10.1109/TBCAS.2016.2539352

Petrovici, M. A. (2016). *Form Versus Function: Theory and Models for Neuronal Substrates*. Springer International Publishing Switzerland. Available online at: https://www.springer.com/gp/book/9783319395517

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2013). Stochastic inference with deterministic spiking neurons. *arXiv 1311.3211*.

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2016). Stochastic inference with spiking neurons in the high-conductance state. *Phys. Rev. E* 94:042312. doi: 10.1103/PhysRevE.94.042312

Petrovici, M. A., Schmitt, S., Klähn, J., Stöckel, D., Schroeder, A., Bellec, G., et al. (2017a). "Pattern representation and recognition with accelerated analog neuromorphic systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (Baltimore, MD: IEEE), 1–4.

Petrovici, M. A., Schroeder, A., Breitwieser, O., Grübl, A., Schemmel, J., and Meier, K. (2017b). "Robustness from structure: inference with hierarchical spiking networks on analog neuromorphic hardware," in *2017 International Joint Conference on Neural Networks (IJCNN)* (Anchorage, AL: IEEE), 2209–2216.

Petrovici, M. A., Stöckel, D., Bytschok, I., Bill, J., Pfeil, T., Schemmel, J., et al. (2015). "Fast sampling with neuromorphic hardware," in *Advances in Neural Information Processing Systems (NIPS)*, eds C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, (Montreal, QC: Neural Information Processing Systems Foundation, Inc.) Vol. 28.

Petrovici, M. A., Vogginger, B., Müller, P., Breitwieser, O., Lundqvist, M., Muller, L., et al. (2014). Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLoS ONE* 9:e108590. doi: 10.1371/journal.pone.0108590

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., et al. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci.* 7:11. doi: 10.3389/fnins.2013.00011

Pfeil, T., Jordan, J., Tetzlaff, T., Grübl, A., Schemmel, J., Diesmann, M., et al. (2016). Effect of heterogeneity on decorrelation mechanisms in spiking neural networks: a neuromorphic-hardware study. *Phys. Rev. X* 6:021023. doi: 10.1103/PhysRevX.6.021023

Pouget, A., Beck, J. M., Ma, W. J., and Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. *Nat. Neurosci.* 16:1170. doi: 10.1038/nn.3495

Probst, D., Petrovici, M. A., Bytschok, I., Bill, J., Pecevski, D., Schemmel, J., et al. (2015). Probabilistic inference in discrete spaces can be implemented into networks of LIF neurons. *Front. Comput. Neurosci.* 9:13. doi: 10.3389/fncom.2015.00013

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., et al. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Front. Neurosci.* 9:141. doi: 10.3389/fnins.2015.00141

Reinagel, P., and Reid, R. C. (2002). Precise firing events are conserved across neurons. *J. Neurosci.* 22, 6837–6841. doi: 10.1523/JNEUROSCI.22-16-06837.2002

Salakhutdinov, R. (2010). "Learning deep boltzmann machines using adaptive MCMC," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, (Haifa), 943–950.

Savitzky, A., and Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.* 36, 1627–1639.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., and Millner, S. (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE), 1947–1950.

Schmitt, S., Klähn, J., Bellec, G., Grübl, A., Guettler, M., Hartel, A., et al. (2017). "Neuromorphic hardware in the loop: training a deep spiking network on the brainscales wafer-scale system," in *2017 International Joint Conference on Neural Networks (IJCNN)* (Anchorage, AL: IEEE), 2227–2234.

Schmuker, M., Pfeil, T., and Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proc. Natl. Acad. Sci. U.S.A.* 111, 2081–2086. doi: 10.1073/pnas.1303053111

Sutskever, I., and Hinton, G. (2007). "Learning multilevel distributed representations for high-dimensional sequences," in *Artificial Intelligence and Statistics*, eds M. Meila and X. Shen (San Juan), 548–555.

Taylor, G. W., and Hinton, G. E. (2009). "Factored conditional restricted boltzmann machines for modeling motion style," in *Proceedings of the 26th Annual International Conference on Machine Learning* (ACM), 1025–1032.

Tetzlaff, T., Helias, M., Einevoll, G. T., and Diesmann, M. (2012). Decorrelation of neural-network activity by inhibitory feedback. *PLoS Comput. Biol.* 8:e1002596. doi: 10.1371/journal.pcbi.1002596

Toups, J. V., Fellous, J. M., Thomas, P. J., Sejnowski, T. J., and Tiesinga, P. H. (2012). Multiple spike time patterns occur at bifurcation points of membrane potential dynamics. *PLoS Comput. Biol.* 8:e1002615. doi: 10.1371/journal.pcbi.1002615

Waldrop, M. M. (2016). The chips are down for Moore's law. *Nat. News* 530:144. doi: 10.1038/530144a

Wunderlich, T., Kungl, A. F., Müller, E., Hartel, A., Stradmann, Y., Aamir, S. A., et al. (2019). Demonstrating advantages of neuromorphic computation: a pilot study. *Front. Neurosci.* 13:260. doi: 10.3389/fnins.2019.00260

Xiao, H., Rasul, K., and Vollgraf, R. (2017). *Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms* (Zalando SE).

Zoschke, K., Güttler, M., Böttcher, L., Grübl, A., Husmann, D., Schemmel, J., et al. (2017). "Full wafer redistribution and wafer embedding as key technologies for a multi-scale neuromorphic hardware cluster," in *2017 IEEE 19th Electronics Packaging Technology Conference* (Singapore).

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.

ORIGINAL RESEARCH
published: 14 January 2020
doi: 10.3389/fnins.2019.01330

A Digital Hardware System for Spiking Network of Tactile Afferents

Nima Salimi-Nezhad[1], Erfan Ilbeigi[1], Mahmood Amiri[2], Egidio Falotico[3] and Cecilia Laschi[3]*

[1] Medical Biology Research Center, Kermanshah University of Medical Sciences, Kermanshah, Iran, [2] Medical Technology Research Center, Kermanshah University of Medical Sciences, Kermanshah, Iran, [3] The BioRobotics Institute, Scuola Superiore Sant'Anna, Pontedera, Italy

In the present research, we explore the possibility of utilizing a hardware-based neuromorphic approach to develop a tactile sensory system at the level of first-order afferents, which are slowly adapting type 1 (SA-I) and fast adapting type 1 (FA-I) afferents. Four spiking models are used to mimic neural signals of both SA-I and FA-I primary afferents. Next, a digital circuit is designed for each spiking model for both afferents to be implemented on the field-programmable gate array (FPGA). The four different digital circuits are then compared from source utilization point of view to find the minimum cost circuit for creating a population of digital afferents. In this way, the firing responses of both SA-I and FA-I afferents are physically measured in hardware. Finally, a population of 243 afferents consisting of 90 SA-I and 153 FA-I digital neuromorphic circuits are implemented on the FPGA. The FPGA also receives nine inputs from the force sensors through an interfacing board. Therefore, the data of multiple inputs are processed by the spiking network of tactile afferents, simultaneously. Benefiting from parallel processing capabilities of FPGA, the proposed architecture offers a low-cost neuromorphic structure for tactile information processing. Applying machine learning algorithms on the artificial spiking patterns collected from FPGA, we successfully classified three different objects based on the firing rate paradigm. Consequently, the proposed neuromorphic system provides the opportunity for development of new tactile processing component for robotic and prosthetic applications.

Keywords: tactile sensing, spiking network, digital circuit, mechanoreceptor, primary afferents

OPEN ACCESS

Edited by:
Michael Pfeiffer,
Bosch Center for Artificial Intelligence, Germany

Reviewed by:
Benjamin Ward-Cherrier,
University of Bristol, United Kingdom
Jim Harkin,
Ulster University, United Kingdom

****Correspondence:***
Mahmood Amiri
ma_amiri_bme@yahoo.com

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *31 July 2019*
Accepted: *26 November 2019*
Published: *14 January 2020*

Citation:
Salimi-Nezhad N, Ilbeigi E, Amiri M, Falotico E and Laschi C (2020) A Digital Hardware System for Spiking Network of Tactile Afferents. Front. Neurosci. 13:1330. doi: 10.3389/fnins.2019.01330

INTRODUCTION

The sense of touch covers the whole body using a variety of receptors in different depth of skin. Information coming from muscles and tendons (kinesthetic sensing) and rich signals from touch receptors embedded in the skin (cutaneous sensing) play a crucial role in our sensory experience, and thus, we are able to actively communicate with our surrounding world. Specifically, when we interact with an object, information about that object characteristics such as its shape and texture is carried in the spatiotemporal pattern of action potentials evoked in a variety of tactile afferents. These action potentials or *spikes* are transmitted by the primary afferents to the spinal cord, cuneate nucleus, thalamus, and finally somatosensory cortex for decoding and decision making. Consequently, we are able to recognize objects based on tactile exploration (Dahiya et al., 2010, 2013). The specialized mechanoreceptors in the human glabrous skin are composed of two main types, based on their functionality and their receptive field, (1) slowly adapting (SA) afferent and

(2) the fast adapting (FA) afferent (Dahiya and Valle, 2012; Tiwana et al., 2012). The SA type 1 (SA-I) and type II (SA-II) afferents innervate Merkel and Ruffini cylinder, respectively, and are mostly sensitive to static stimuli. The FA type 1 (FA-I) and type II (FA-II) afferents, which are sensitive to transient events such as vibration, innervate the Meissner corpuscle and Pacinian corpuscle, respectively (Lucarotti et al., 2013). In this study, we focus on the SA-I and FA-I tactile afferents, which are necessary elements for object manipulation (Johansson and Flanagan, 2009).

Recent approaches aim to mimic the behavior of the biological tactile receptors using advanced skin dynamics (Saal et al., 2017) and neuromorphic models (Oddo et al., 2016) to progress the efficiency and performance over traditional techniques. Application of spiking neural networks and neuromorphic approaches in tactile systems are increasing in the past few years (Kim et al., 2009; Friedl et al., 2016; Oddo et al., 2016; Yi and Zhang, 2016). Pearson et al. (2006, 2007, 2011) developed a biomimetic vibrotactile sensory system using leaky integrate-and-fire neuron models, which replicates rat whiskers for enabling a robot to navigate its environment. To discriminate local curvature of an object, Lee et al. (2013) used a fabric-based binary tactile sensor array. The tactile signals were converted into spikes using the Izhikevich model (Lee et al., 2014). For decoding Braille letters, a closed perception-action loop was made by converting force sensor data to spike trains using the leaky integrate-and-fire model (Bologna et al., 2011, 2013). An Izhikevich neuron model was used by Spigler et al. (2012) for characterizing surface properties. Zhengkun and Yilei (2017) transformed the outputs of polyvinylidene difluoride tactile sensors to spike trains using the Izhikevich model and then applied machine learning algorithm for classification of surface roughness. Rongala et al. (2017) classified 10 naturalistic textures by converting the outputs of an array of four piezoresistive sensors into spike trains. They used the Izhikevich model and analyzed the obtained spiking patterns (Rongala et al., 2017). Using the same sensor, Oddo et al. transduced haptic stimulus into a spatiotemporal pattern of spikes and then applied them to the rat skin afferents using stimulation electrodes. In this way, they showed a potential for neuro-prosthetic approach to communicate with the rat brain (Oddo et al., 2017). Moreover, neuromorphic techniques have been used to induce tactile sensation for differentiating textures using SA-like dynamics through nerve stimulation of an amputee (Oddo et al., 2016) and to enhance grip functionality of the prosthesis (Osborn et al., 2017). In Osborn et al. (2018), it was focused on pain detection through a neuromorphic interface and initiated an automated pain reflex in the prosthesis.

One of the most common methods to realize the neural computational models is developing digital circuit due to its high efficiency for practical applications (Cassidy et al., 2011). Digital execution with field-programmable gate array (FPGA) affords flexibility necessary for algorithm exploration while filling time and performance constraints. Therefore, FPGAs have increasing applications in the neural computing area (Nanami and Kohno, 2016). Furthermore, with the advancement in HDL (high-level hardware description language) synthesis tools, FPGA can also be operated as the effective hardware accelerators (Misra and Saha, 2010; Arthur et al., 2012). Some researchers have worked on efficient hardware implementations (Wang et al., 2018; Zjajo et al., 2018). Grassia et al. (2016) simulated a stochastic neuron in FPGA. An approximate circuit technique was used for FPGA implementation of real-time processing of tactile data to be utilized in the e-skin applications (Franceschi et al., 2017). Ambroise et al. (2017) proposed a biomimetic neural network implemented on FPGA for bi-directional communication with living neurons cultured in microelectrode array. A digital hardware realization was proposed for spiking model of cutaneous mechanoreceptor in order to identify the applied pressure stimulus (Salimi-Nezhad et al., 2018). They used the Izhikevich neuron model for simulation and then digital execution of the SA-I and FA-I afferents on the FPGA. Indeed, their approach is the proof of concept that digital circuit implementation of tactile afferents has great potential. However, it is necessary to extend previous work that considers one SA-I or FA-I digital circuit with single input. Actually, tactile information is conveyed not only using multiple sub-modalities but also through ensembles of different afferent types. Consequently, developing a hardware-based neuromorphic system to run a population of various afferents and receive multiple inputs is necessary for modeling study and fabrication of novel tactile sensory system for robotic and prosthetic applications. Accordingly, in this paper, we report that designing of a neuromorphic tactile system using a population of 243 digital afferents includes SA-I and FA-I. To this purpose, first, four spiking models including Izhikevich model (Izh), linearized Izhikevich model (L-Izh), Quadratic Integrated and Fire model (QIF), and linearized QIF model (L-QIF) are considered for simulating the neural afferents. Next, for all of these spiking models, an appropriate digital circuit is presented and simulated in VIVADO. The performance comparison is done to find which of the designed circuit is efficient from area and power consumption viewpoint while maintaining the characteristics of their original mathematical model. Then, the superior circuit is further improved by replacing multipliers with logical shifter. Consequently, the *improved* L-QIF was hired for each afferent to create a neuromorphic network of artificial SA-I/FA-I afferents. Employing an experimental setup, the performance of the digital spiking network, which is executed on the FPGA, is explored. In this case, the indentation data of a 3×3 pressure sensor grid are sent to the FPGA through an interface board. FPGA runs the digital circuits of the 243 spiking model of afferents and processes the incoming data of nine pressure sensors in parallel to deliver tactile spike patterns for the next level of processing. To the best our knowledge, the proposed neuromorphic system is the first digital system implementing a population of tactile afferents (both SA-I and FA-I) while receiving multiple inputs. Finally, by applying machine learning algorithm, the artificial spike responses are analyzed based on the firing rate paradigm, and thus, we classify three objects to show a real application of the proposed neuromorphic tactile system in a haptic experiment.

The rest of the paper are prepared in this way: the spiking models and their digital circuits are described in sections

"Materials and Methods" and "Digital Circuits," respectively. The hardware implementation results are discussed in section "Hardware Implementation." Finally, the section "Conclusion" concludes the paper.

MATERIALS AND METHODS

The mathematical description of four spiking models used in this research has been explained in **Appendix**. Based on these spiking models, we present an appropriate digital circuit for each model. The designed digital circuits are compared to obtain the circuit with minimum area and power consumption characteristics to be used for developing a neuromorphic tactile system.

Spiking Model of Tactile Afferent

The primary afferents in the glabrous skin that convey tactile information are SA-I, II and FA-I, II. In human hand, there are approximately 43% FA-I afferents end with Meissner corpuscles, 13% FA-II units with Pacinian endings, 25% SA-I units innervate Merkel cells, and 19% SA-II units with Ruffini endings (McGlone and Reilly, 2010). Merkel receptors, located superficially in the skin (Roudaut et al., 2012), are triggered by lower-frequency skin deformations and are essential for texture discrimination and fine tactile perception. The SA-I afferents, which branch and innervate the Merkel discs, are active throughout the physical stimulation. Meissner receptors have particularly high density on the fingertips and respond whenever a change in the stimuli is detected (i.e., when the stimuli is applied or when it is removed) (Roudaut et al., 2012). The FA-I afferents, which branch and innervate the Meissner corpuscles, have small receptive fields and detect dynamic skin deformations (Johansson and Vallbo, 1979). They are responsible for detection of low-frequency vibration, slip, and motion.

Figure 1 shows the afferent model used in this study. It was shown that this model reproduces the spike trains generated in the FA-I and SA-I biological counterpart for various stimuli (Saal and Bensmaia, 2015; Friedl et al., 2016; Rongala et al., 2017, 2018; Salimi-Nezhad et al., 2018). In this model, the amount of force value is measured by the sensor, $f(t)$ and its variations $\dot{f}(t)$ (in mN), are weighted separately (C_{x1}, C_{x2}) to make the current $I(t)$ (in mA) for spike generation. Four neural models including Izh, L-Izh, QIF, and L-QIF are used for spiking part of the afferent model, independently. The mathematical descriptions of these four spiking models are explained in **Appendix**.

DIGITAL CIRCUITS

For designing neuromorphic systems, FPGAs are frequently used in recent years, and several successful cases were reported in the literature. Indeed, its parallel and high-speed computation ability afford real-time implementation of spiking neural networks. In this section, spiking models are first discretized using Euler method, and then the digital circuits to be executed on FPGA are presented. For the designed digital circuits, the resource utilization is compared to find the circuit, which has fewer logic blocks. In this way, we can implement a large population of afferents. The discretizing step for all equations is $h = 0.0078125$ ms. In the following equations, we consider that C_m and τ are equal to 1 F and 1 s, respectively.

The Izh Digital Circuit

Equations 21–23 describing the spiking behavior of the SA-I model can be discretized as:

$$v[n+1] = v[n] + h \times (0.04 \times v[n] \times v[n] + 5\times$$

$$v[n] + 140 - u[n] + C_{11} \times I[n]) \tag{1}$$

$$u[n+1] = u[n] + h \times a \times \left(b \times v[n] - u[n]\right) \tag{2}$$

$$\text{if } v[n+1] \geq 30 \text{ mV} \rightarrow \quad \text{then} \quad \begin{cases} v[n+1] & \leftarrow c \\ u[n+1] & \leftarrow u[n] + d \end{cases} \tag{3}$$

The scheduling diagram for this model is illustrated in **Figure 2A**. Similarly, for the FA-I model, the discretized equations are as follows:

$$v[n+1] = v[n] + h \times (0.04 \times v[n] \times v[n] + 5\times$$

$$v[n] + 140 - u[n]) + C_{12} \times (I[n+1] - I[n]) \tag{4}$$

$$u[n+1] = u[n] + h \times a \times \left(b \times v[n] - u[n]\right) \tag{5}$$

$$\text{if } v[n+1] \geq 30 \text{ mV} \rightarrow \quad \text{then} \quad \begin{cases} v[n+1] & \leftarrow c \\ u[n+1] & \leftarrow u[n] + d \end{cases} \tag{6}$$

The scheduling diagram for the FA-I model is shown in **Figure 2B**. It illustrates how membrane potential (v) and recovery variable (u) of the afferent model in each iteration are generated. There are also memory registers to store the outputs for use in the subsequent steps. The register length, N, to solve individual state variables is $N = 32$ (1 bit for sign, 13 bits for integer part, and 18 bits for fractional part) to obtain a low-error and high-speed circuit (Salimi-Nezhad et al., 2018). It should be pointed out that "N" directly affects the computational time and the required precision for implementation.

The L-Izh Digital Circuit

To design the digital circuit for the L-Izh model of the SA-I afferent, Eqs 27–29 are discretized as follows:

$$v[n+1] = v[n] + h \times (k_1 \times |v[n] +$$

$$62.5| - k_2 - u[n] + C_{21} \times I[n]) \tag{7}$$

$$u[n+1] = u[n] + h \times a \times \left(b \times v[n] - u[n]\right) \tag{8}$$

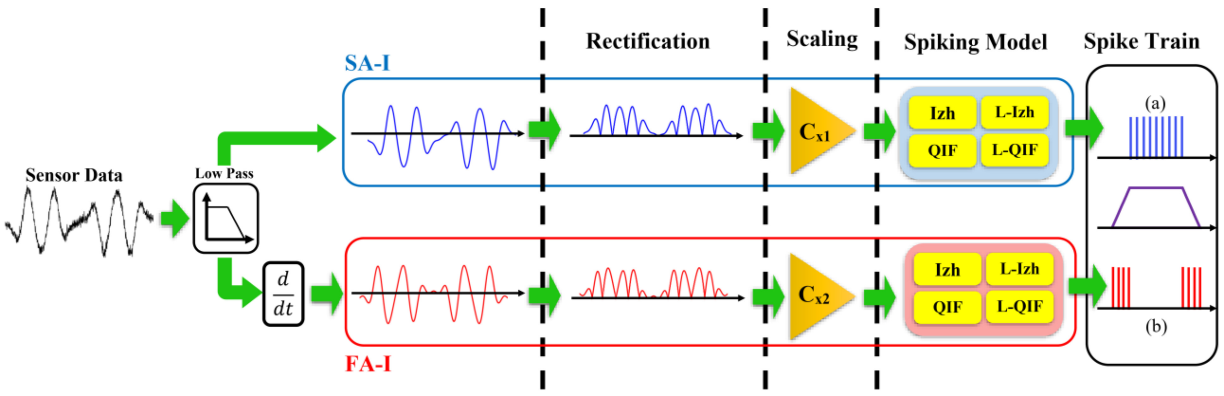


FIGURE 1 | The slowly adapting type 1 (SA-I) and fast adapting type 1 (FA-I) afferents model. The SA-I responds to the absolute value of the stimulus and is active all over the interval of the stimulus contact. The FA-I delivers spikes when stimulus has dynamic, i.e., during onset and offset phases of indentation profile. Four neural models including the Izhikevich model (Izh), linearized Izhikevich model (L-Izh), Quadratic Integrated and Fire model (QIF), and linearized QIF model (L-QIF) are used for spike generation of the afferent model, independently.

$$\text{if } v[n+1] \geq 30\text{ mV} \rightarrow \text{ then } \begin{cases} v[n+1] \leftarrow c \\ u[n+1] \leftarrow u[n]+d \end{cases} \quad (9)$$

Accordingly, the scheduling diagram is depicted in **Figure 3A**. For FA-I afferent, the discrete equations of L-Izh model are as follows:

$$v[n+1] = v[n] + h \times (k_1 \times |v[n] + 62.5|$$

$$-k_2 - u[n]) + C_{22} \times (I[n+1] - I[n]) \quad (10)$$

$$u[n+1] = u[n] + h \times a \times \big(b \times v[n] - u[n]\big) \quad (11)$$

$$\text{if } v[n+1] \geq 30\text{ mV} \rightarrow \text{ then } \begin{cases} v[n+1] \leftarrow c \\ u[n+1] \leftarrow u[n]+d \end{cases} \quad (12)$$

and the scheduling diagram is presented in **Figure 3B**. It shows how membrane potential (v) and recovery variable (u) of the afferent model in each iteration are produced.

The QIF Digital Circuit

Equations (33)–(34), which are responsible for producing spiking patterns in the SA-I model, are discretized as follows:

$$v[n+1] = v[n] + h \times (M_1 \times v[n] \times v[n] + C_{31} \times I[n]) \quad (13)$$

$$\text{if } v[n+1] \geq v_{\text{peak}} \rightarrow \text{ then } v[n+1] = v_{\text{reset}} \quad (14)$$

and the scheduling diagram for this model is presented in **Figure 4A**. Also, the discretized equations for FA-I model are:

$$v[n+1] = v[n] + h \times (M_1 \times v[n] \times v[n]) +$$

$$C_{32} \times (I[n+1] - I[n]) \quad (15)$$

$$\text{if } v[n+1] \geq v_{\text{peak}} \rightarrow \text{ then } v[n+1] = v_{\text{reset}} \quad (16)$$

The scheduling diagram for this model is shown in **Figure 4B**.

The L-QIF Digital Circuit

Parallel to the method used in the previous subsections, Eqs 37 and 38 for SA-I model are discretized as follows:

$$v[n+1] = v[n] + h \times (M_2 \times |v[n]| + C_{41} \times I[n]) \quad (17)$$

$$\text{if } v[n+1] \geq v_{\text{peak}} \rightarrow \text{ then } v[n+1] = v_{\text{reset}} \quad (18)$$

and the scheduling diagram for this model is demonstrated in **Figure 5A**. Finally, the discretized equations for FA-I model are:

$$v[n+1] = v[n] + h \times (M_2 \times |v[n]|) +$$

$$C_{42} \times (I[n+1] - I[n]) \quad (19)$$

$$\text{if } v[n+1] \geq v_{\text{peak}} \rightarrow \textit{ then } v[n+1] = v_{\text{reset}} \quad (20)$$

and the scheduling diagram is illustrated in **Figure 5B**.

The digital circuits, **Figures 2–5**, based on spiking model of afferents, are the neuromorphic conversion of sensor output to spiking patterns conveying tactile information. **Table 1** compares the resources utilized by the different digital circuits for both SA-I and FA-I models. As it is observed, the digital circuits for the linearized models (L-Izh and L-QIF) are more area efficient compared with their original counterparts (the Izh and QIF models). Also, it is apparent that the L-QIF digital circuit uses the minimum resources. Considering **Table 1**, the hardware resource utilization even for the Izh digital circuits compared to the circuits reported in Salimi-Nezhad et al. (2018) is decreased. Specifically, in the present research, for the

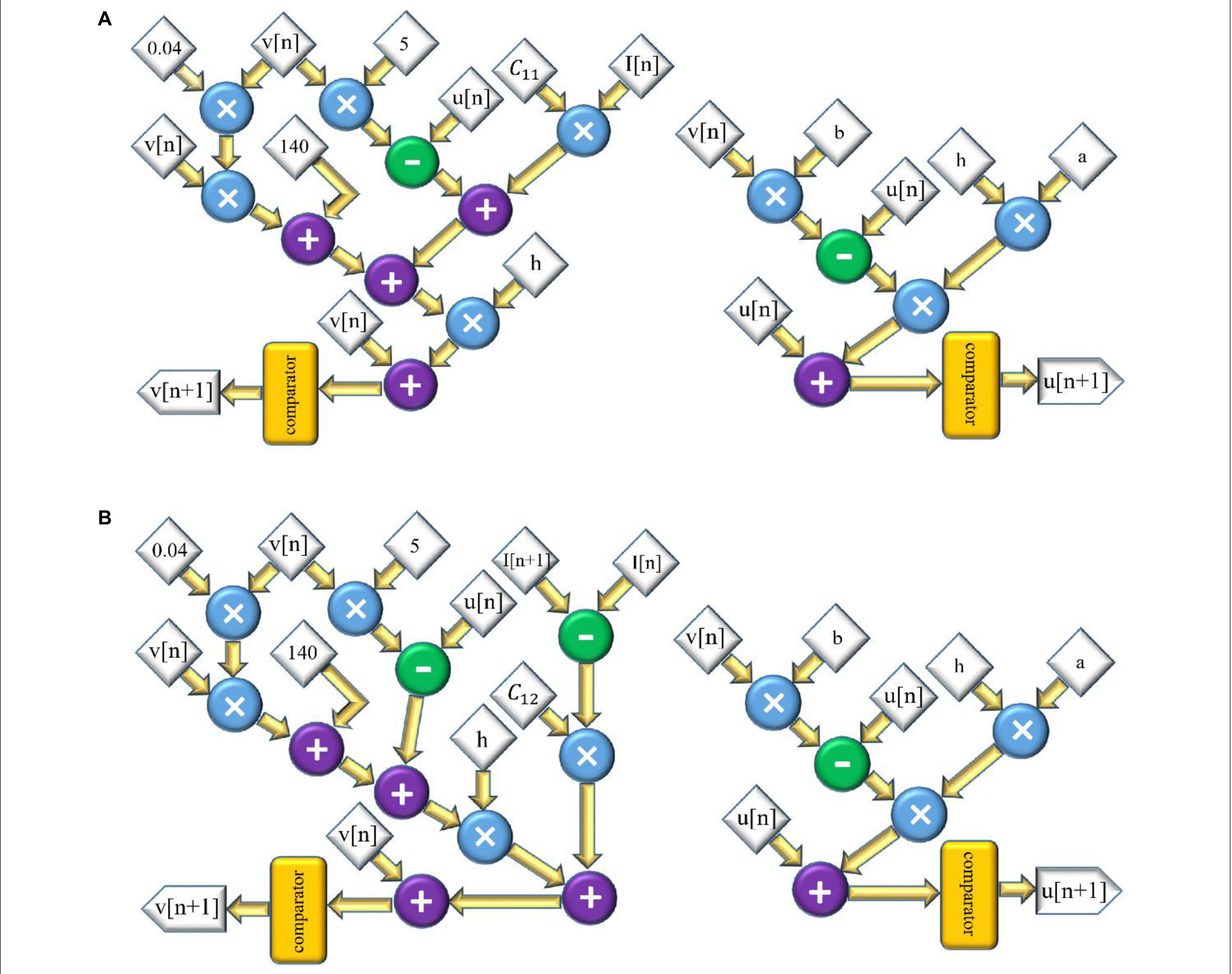


FIGURE 2 | The scheduling diagram for spike generation of SA-I afferent **(A)** and FA-I afferent **(B)** using the Izh spiking model. In these diagrams, there are two state variables, *v* and *u*, so two digital circuits are designed for each variable, distinctly.

Izh digital circuits, we have used less number of DSP for SA-I and FA-I afferents compared to the circuits reported in Salimi-Nezhad et al. (2018).

Simulation Results

In this section, we present the results of MATLAB simulation of four types of spiking models for both afferents (SA-I, FA-I) and VIVADO simulations of their digital circuits. **Figure 6** shows the time responses of the SA-I spiking model with trapezoidal input. Increasing input current causes decreasing inter-spike interval. **Figure 7** demonstrates the time responses of the FA-I spiking model with trapezoidal input. Higher value of slope motivates the model to produce spike patterns with higher rate. In **Figures 6**, **7**, the first panels display the trapezoidal pulse as the input signal, the second panels present the MATLAB simulations of the afferent model, and the third panels demonstrate the VIVADO simulation of the digital circuit.

Considering **Figures 6**, **7**, the SA-I afferent fires throughout a sustained phase of stimulus and the FA-I afferent responds at the onset and offset phases of that stimulus. This result is functionally in agreement with the response measured by the observations reported in Jörntell et al. (2014). In other words, the spiking model and their digital circuit have similar responses and functionally are compatible with spiking activity of biological afferent.

Population of Digital Afferents

Although in previous sections, we found that the L-QIF model has the least area consumption in comparison with the other three models, we can also use other techniques for further reduction in the hardware utilization. Indeed, multipliers are costly blocks, which consume more power and use more area compared to the simple blocks such as adders or shifters. For this reason, by replacing multipliers with logical shifter, the *improved*

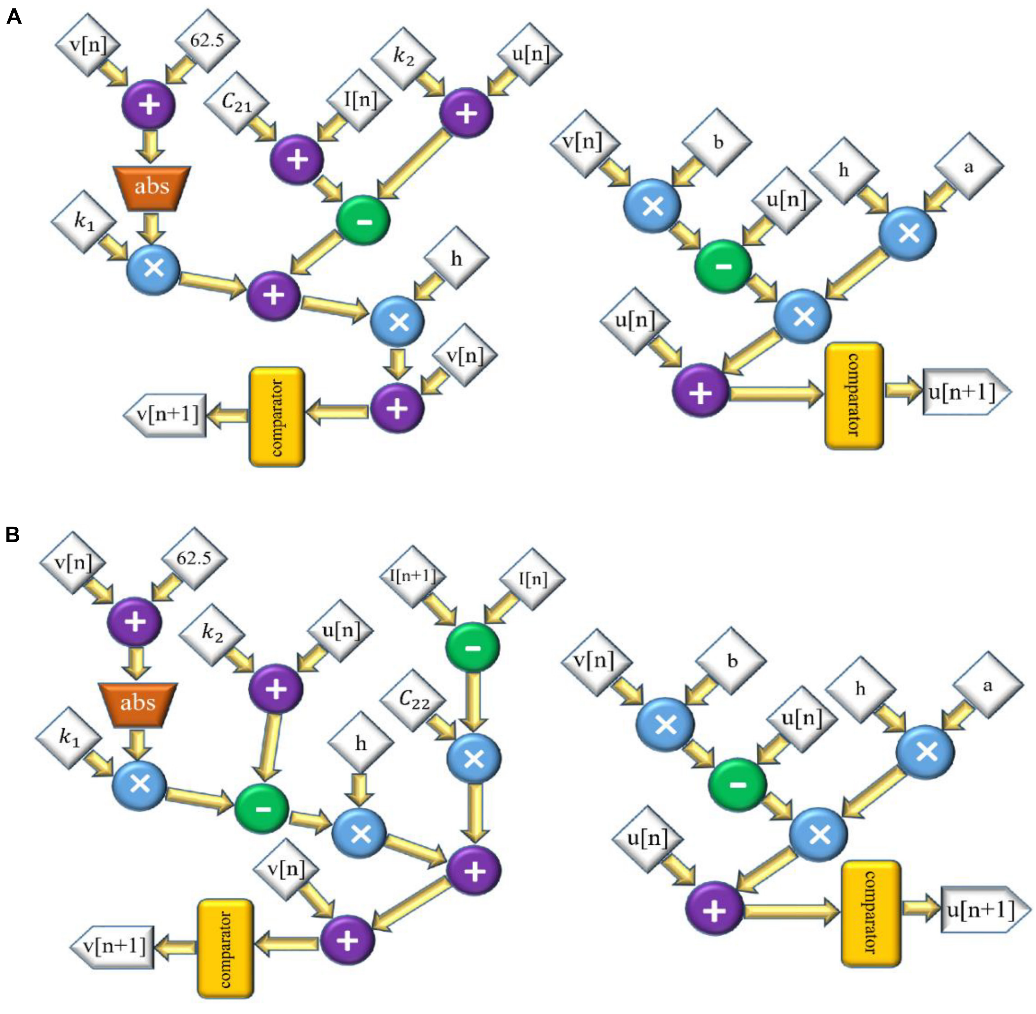


FIGURE 3 | The scheduling diagram for spike generation of SA-I afferent **(A)** and FA-I afferent **(B)** using the L-Izh model. In these diagrams, there are two state variables, *v* and *u*, so two digital circuits are designed for each variable, separately. Compared to **Figure 2**, due to linearization, this model consumes less hardware area and has less power consumption.

L-QIF is obtained with the coefficients described in **Table 2**. Consequently, we expect an increase in operational frequency, due to the lack of high-cost operation (multipliers) to slow down the important paths. Furthermore, this approach reimburses the limited number of available multipliers on the chip and supports the implementation of larger spiking networks on the FPGA. Parameter values in **Table 2** are chosen to show a better and a clear view of the spiking responses in the raster plot of population of afferents. In this way, we modified and tuned the experimental parameters from the simulation parameters.

Table 3 compares the *improved* L-QIF digital circuit with the L-QIF circuit. It is apparent that replacing multipliers with shift registers leads to the decrease in DSP and Slice LUTs count, while the number of LUT Flip Flop increases. In this way, more resources can be saved if one uses the improved L-QIF model for spiking model of afferents. This can be quite important when a population of afferents is implemented on the FPGA. It should be pointed out that, although modern FPGAs have a significant number of DSP slices, equipping prosthesis and robotic hands with human-like skin needs implementation of thousands of mechanoreceptors and afferents to enable simultaneous transmission of tactile information. Therefore, saving energy and area utilization is quite important for practical applications. Here, we demonstrate a prototype for 243 artificial afferents that transmit spikes asynchronously conveying spatiotemporal features necessary for tactile perception.

HARDWARE IMPLEMENTATION

The neuromorphic implementation of tactile afferents can speed up the development of novel artificial tactile sensory systems in the field of telerobotics and teleoperation. Consequently, in the current research, the hardware-based neuromorphic implementation is performed. To show the performance of the designed circuit and to illustrate the spiking patterns of

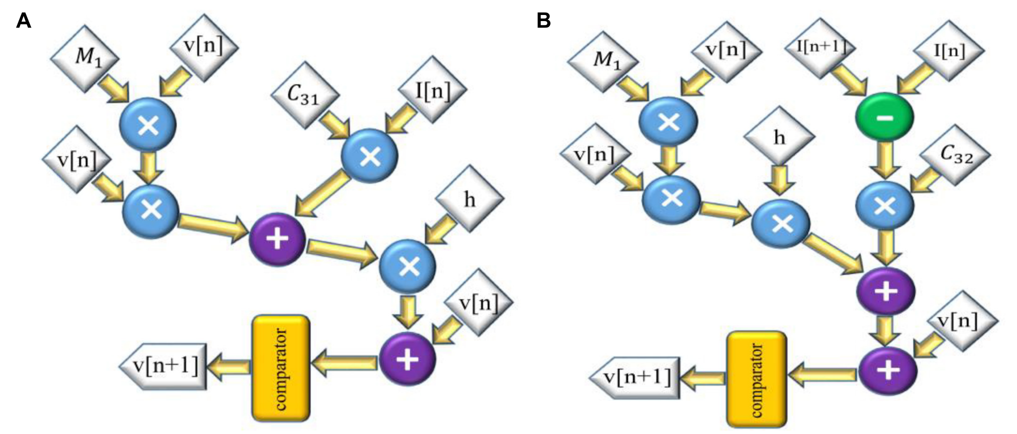


FIGURE 4 | The scheduling diagram for spike generation of SA-I afferent **(A)** and FA-I afferent **(B)** using the QIF model. In these diagrams, there is only one state variable, *v*, the membrane potential. Compared to **Figures 2**, **3**, the digital circuit of the QIF model is simpler, consumes less hardware area, and has less power consumption.

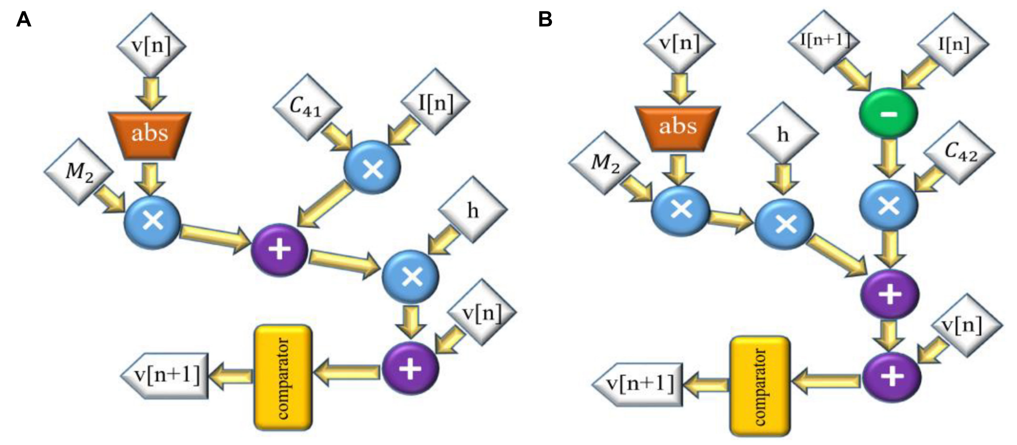


FIGURE 5 | The scheduling diagram for spike generation of SA-I afferent **(A)** and FA-I afferent **(B)** using the L-QIF model. In these diagrams, there is only one state variable, *v*, the membrane potential. Compared to **Figures 2–4**, the digital circuit for the linearized version of the QIF model is much simpler, consumes less hardware area, and has less power consumption.

TABLE 1 | Device utilization summary for the four designed digital circuits for both afferents.

	Izh		L-Izh		QIF		L-QIF		
	SA-I	FA-I	SA-I	FA-I	SA-I	FA-I	SA-I	FA-I	Available
Slice LUTs	1341	1436	1098	1192	839	882	250	308	53,200
Slice registers	97	129	97	129	65	97	65	97	106,400
Slice	726	750	558	582	380	396	223	246	13,300
LUT flip flop pairs	34	34	34	34	18	18	18	18	53,200
DSP48E1	8	8	6	6	4	4	3	3	220

Izh, Izhikevich model; L-Izh, linearized Izhikevich model; QIF, Quadratic Integrated and Fire model; L-QIF, linearized QIF model; SA-I, slowly adapting type 1; FA-I, fast adapting type 1.

a population of digital afferents, an experimental setup was developed as demonstrated in **Figure 8**. It consists of nine sensing units (a matrix of 3 × 3) connected to a ZedBoard through a custom interfacing board. The applied force to the individual Force-Sensitive Resistors (FSRs) provides an analog signal for the 10-bit ADC (analog to digital convertor), which is fed to the ZYNQ (in this case, ZedBoard). The ZedBoard (a particular ZYNQ evaluation board) is one of the low-cost and high-speed

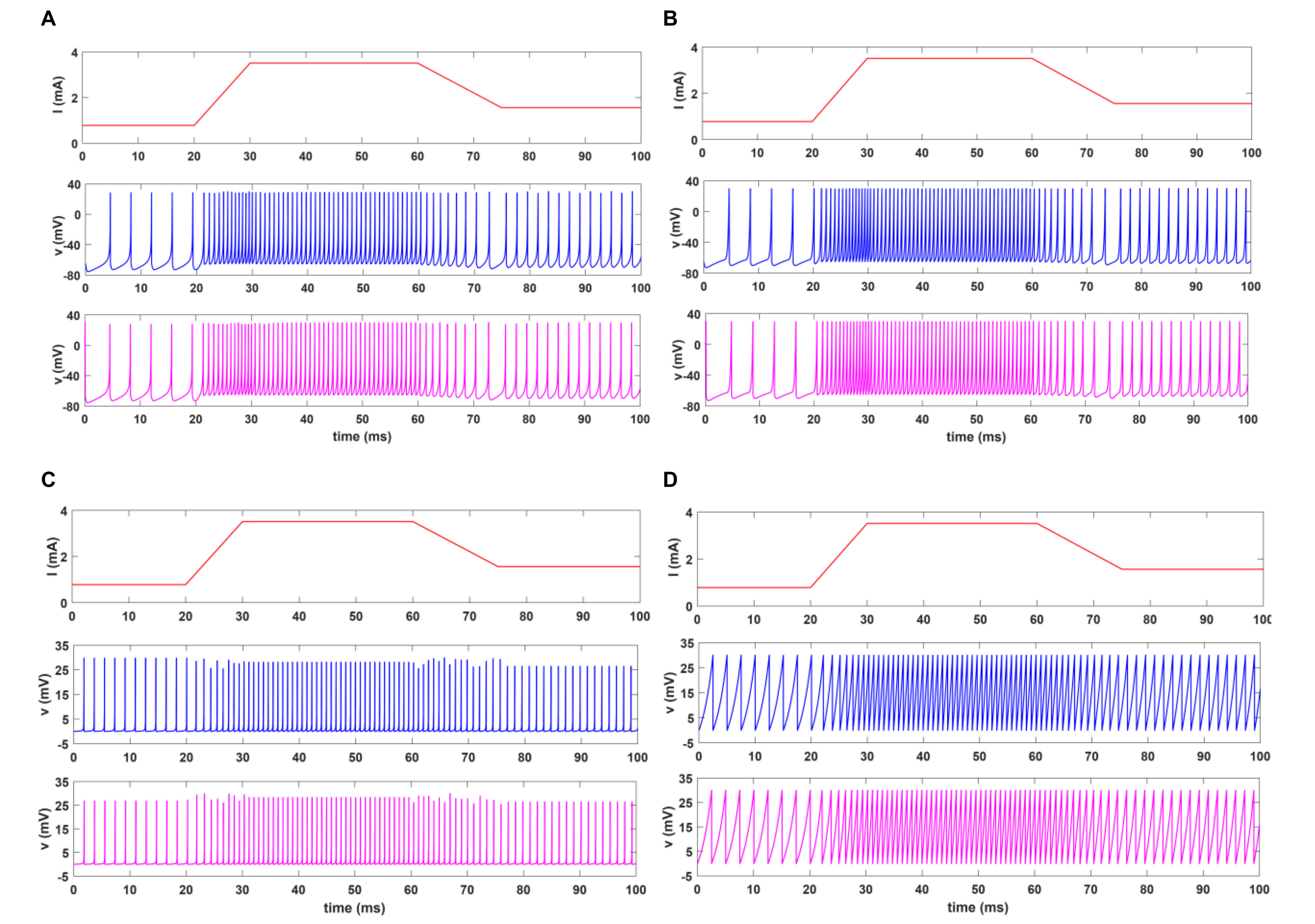


FIGURE 6 | The time response of the spiking model of the SA-I afferent in 100 ms for Izh **(A)**, L-Izh **(B)**, QIF **(C)**, and L-QIF **(D)**. In these simulations, the first panels show the input signal, the second panels display the MATLAB simulation of the mathematical model, and the third panels illustrate the VIVADO simulation of the digital circuit. Average frequencies in 100-ms simulation for Izh, L-Izh, QIF, and L-QIF are 760, 840, 880, and 780 Hz, respectively.

devices for digital realizations of spiking neurons. It is composed of two major sections: Programmable Logic (PL) and Processing System (PS). The PL section is a platform that can be configured using VHDL language and the PS section is a dual-core ARM cortex-A9 processor that can be programmed by C language. The output of ZedBoard is illustrated in two ways. One way is to show on oscilloscope, and the other is to display on the screen. Oscilloscope is used to show the spiking responses of the individual SA-I or FA-I digital circuit, and screen is employed to illustrate the activities of the whole population or subpopulation of digital afferents, simultaneously.

Due to resources available in the ZedBoard evaluation kit, we have implemented 243 digital circuits of the improved L-QIF models in the PL section including 90 SA-I and 153 FA-I. This ratio is chosen to consider that the number of SA-I and FA-I afferents exists in the fingertip (McGlone and Reilly, 2010; Pasluosta et al., 2017). In our design, for each FSR sensor, 27 digital afferents are run on the ZedBoard (10 SA-I and 17 FA-I). The hardware utilization for realization of 243 digital afferents is presented in **Table 4**. It should be mentioned that the operating frequency of ZYNQ is 100 MHz. Accordingly, in this experimental setup, the delay from the onset of applying force to the FSRs to the appearance of spiking responses on the ZYNQ output pins is in the range of nanoseconds.

Considering final applications, simplicity of hardware implementation is an important factor. This feature is essential for development of sensory modules, which tries to integrate sensory and processing circuits. Indeed, spike-based representation of information has a significant potential to improve performance and efficiency of artificial tactile sensing systems. In this way, the proposed digital circuit enabled us to design a hardware architecture for executing a population of afferents on the PL. This new approach for fabricating sensory systems artificially replicates the firing patterns of the SA-I and FA-I afferents. The compartmentalized structure of the proposed approach and the ability to control parameters facilitate for easy scalability without extensive circuit redesign.

Next, using the prepared experimental setup, we touch one, two, or three randomly selected FSR sensors simultaneously from

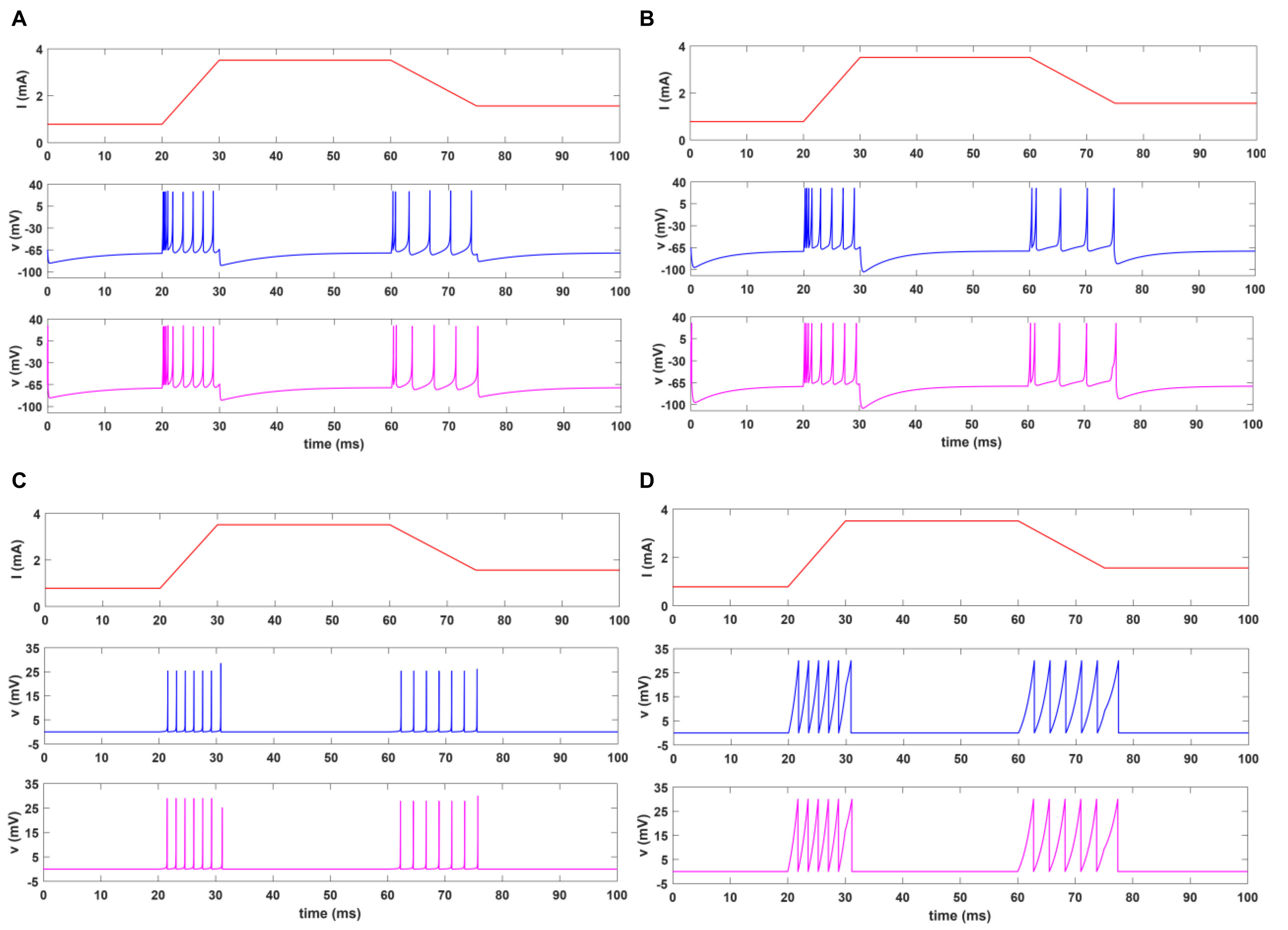


FIGURE 7 | The time response of the spiking model of the FA-I afferent in 100 ms for Izh **(A)**, L-Izh **(B)**, QIF **(C)**, and L-QIF **(D)**. In these simulations, the first panels show the input signal, the second panels display the MATLAB simulation of the mathematical model, and the third panels illustrate the VIVADO simulation of the digital circuit.

TABLE 2 | Parameter values of the improved L-QIF digital circuit.

M_2	0.25
C_{41}	0.5
C_{42}	16

TABLE 3 | Comparison of the hardware utilization for the L-QIF and the improved L-QIF digital circuits for both afferents.

	L-QIF		Improved L-QIF		
	SA-I	FA-I	SA-I	FA-I	Available
Slice LUTs	250	308	108	167	53,200
Slice registers	65	97	32	64	106,400
Slice	223	246	30	56	13,300
LUT flip flop pairs	18	18	29	32	53,200
DSP48E1	3	3	0	0	220

the 3 × 3 pressure sensor matrix as illustrated in **Figure 9**. In this figure, the activated sensors are shown by the red boxes. For instance, in **Figure 9D**, three sensors are simultaneously touched, while in **Figures 9B,F,H**, two randomly selected sensors are touched at the same time.

The spiking responses of the touched sensors in **Figure 9** are shown in **Figure 10**. **Figure 10A** shows the spiking activity of the all 90 SA-I digital circuits, and **Figure 10B** demonstrates the spiking patterns of the all 153 FA-I digital circuits. Indeed, we used the FPGA parallel processing capability for realizing population of digital afferents. In **Figure 10**, for the first 4 s, no sensor has been touched and only the background activity of the population of artificial afferents is observed. Next, regarding **Figure 9A**, the first sensor, S1, is touched. In this case, from $t = 4$ to $t = 6$ s, the applied force to S1 sensor increases from zero to a desired level. From $t = 6$ to $t = 9.5$ s, the force value is maintained in this level. From $t = 9.5$ to 10.5 s, the applied force is reduced to its initial value, which is zero. **Figures 10A,B** show the firing activities of the population of artificial afferents, which are running on the ZedBoard. The digital SA-I afferents remain active during the period of stimulus contact, while the digital FA-I afferents react whenever a change in the stimuli is

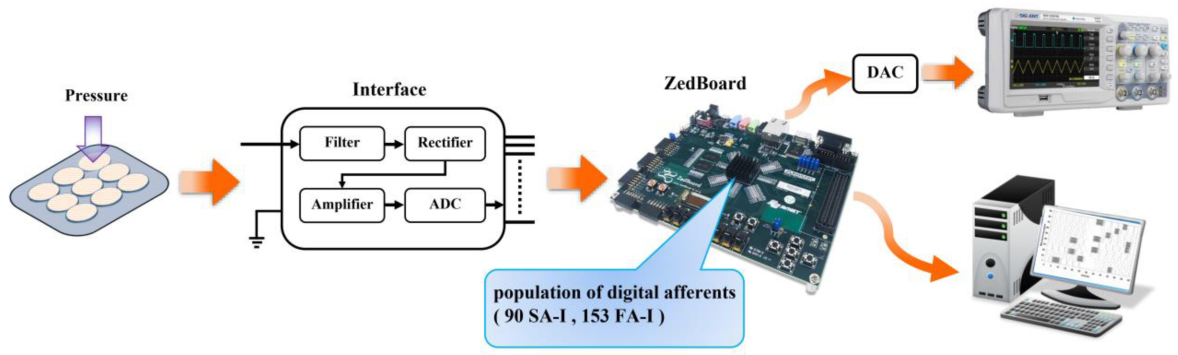


FIGURE 8 | The experimental setup for evaluating the neuromorphic tactile system. A population of 90 digital SA-I afferents and 153 digital FA-I afferents are implemented on the field-programmable gate array (FPGA). In addition to the ZYNQ evaluation board, the system is composed of two other components: a matrix of 3 × 3 sensing units and an interface circuit (equipped with a 10-bit ADC unit) between the sensing unit and the ZedBoard. The sensing unit is composed of nine Force-Sensitive Resistors (FSRs) to deliver the detected pressure as an analog voltage signal to the interface unit. This unit filters, rectifies, and scales its input signal and then converts it to the digital signal to be sent to the ZedBoard. Resistance of the FSRs changes by applying an external force. Depending on the amount of pressure applied to the individual FSR sensor, digital afferents send spike trains to the screen or to the output pin of the ZedBoard to be displayed on the oscilloscope (after analog conversion).

TABLE 4 | Hardware operation for realization of 243 afferents (SA-I/FA-I) in the ZedBoard using the improved L-QIF digital circuit.

	Used	Available
Slice LUTs	33, 249 (62%)	53,200
Slice registers	15, 111 (14%)	106,400
Slice	10, 202 (77%)	13,300
LUT as logic	33, 187 (62%)	53,200
LUT as memory	62 (1%)	17,400
LUT flip flop pairs	8723 (16%)	53,200
DSP48E1	180 (82%)	220
Bonded IO	21 (10%)	200

detected. Similarly, considering **Figure 9B**, both sensors S3 and S5 are touched concurrently. In this way, from t = 14 to 17 s, the applied forces to S3 and S5 increase from zero to another chosen level. From t = 17 to 19 s, the force value is maintained in this selected level. From t = 19 to 20 s, the applied force is again reduced to its initial value, which is zero. It should be pointed out that the amount of applied force to S3 is higher than S5, and thus, firing rate is increased, accordingly. Regarding **Figure 10**, the firing rate of the artificial SA-I is proportional to the intensity of stimulus, while firing patterns of the artificial FA-I appear when there is a changes in the stimulus intensity. Indeed, the different spiking sequences are evoked by applying different force profiles to the FSR sensors.

To obtain more insights, we select four cases from **Figure 10**, the colored regions, and then explore the behavior of the SA-I and FA-I digital circuits in more detail as shown in **Figures 11**, **12**, respectively. In other words, not only the firing patterns of the whole population are illustrated in **Figure 10**, but also we show the spiking responses of the selected afferent on the oscilloscope screen (**Figures 11**, **12**). Yellow, magenta, cyan, and green illustrate the spiking patterns arising from touching S1, S2, S4, and S8, respectively (see **Figure 9**). In **Figures 11**, **12**, from each subpopulation, one (the first) implemented artificial afferent is chosen (the red rectangle in the colored regions) to be displayed on the oscilloscope. In this case, the output of the selected digital afferent after converting to analog signal is demonstrated on the oscilloscope. In these figures, the output of the ZedBoard was shown in yellow color (membrane voltage). As it is observed, as the force magnitude increases, the firing frequency of the spiking patterns for digital SA-I is also increased. This approach makes possible to decode stimuli, while the tactile data are collected. Moreover, it is observed from **Figure 12** that the rate of spiking responses in the offset phase is less than the onset phase for digital FA-I, due to the smaller slope for the offset phase. Indeed, the SA-I afferents provide encoding of pressure and FA-I afferents encode transient performance of the signal.

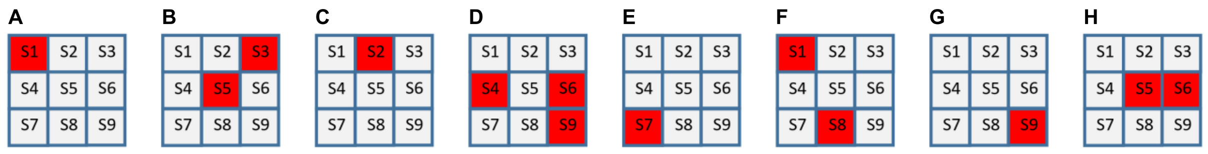


FIGURE 9 | Randomly touching one **(A,C,E,G)**, two **(B,F,H)**, or three FSR sensors **(D)** from the 3 × 3 grid in the experimental setup shown in **Figure 8**. The activated sensors are shown by the red boxes. Different amounts of forces with dissimilar time profiles are applied to the FSR sensors. **(A)** to **(H)** show the sequence of touched sensors in eight stages, respectively.

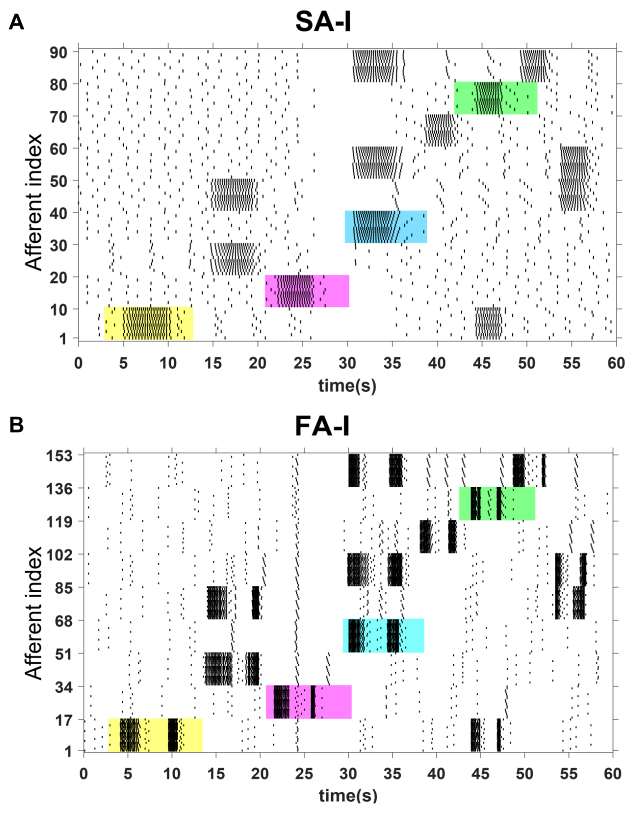


FIGURE 10 | The raster plot for firing patterns of population of 243 digital afferents executed on the ZedBoard. The spiking patterns for 90 SA-I afferents **(A)** and 153 FA-I afferents **(B)** are shown for 60 s. Not only the touched sensors are selected randomly, but also the time duration and the onset and offset velocities are different. The spiking responses of four cases were highlighted by the colored regions to be further investigated in **Figures 11**, **12**.

In addition, in order to show a practical application of the proposed neuromorphic setup, we attached five FSR sensors on a glove (each FSR sensor was allocated to a finger) and performed some haptic experiments while sending FSR outputs to the spiking network of tactile afferents implemented on the ZedBoard. The subject wears the glove to pick up, hold, and put in the place three different objects (a glass, a tape dispenser, and a book) while the firing activity of population of afferents is being measured. As shown in **Figure 13**, these objects have various size and weights. Object A, the glass, has the lowest weight, and object C, the book, is the heaviest one. Each experiment takes 4 s, and the hold phase is 3 s fixed. The subject accomplished the experiment for three cases: first by three fingers (thumb, index, and middle), then four fingers (thumb, index, middle, and ring), and finally with all five fingers. Each three-, four-, and five-finger experiment was done 20 times for individual objects. Consequently, 60 trials were collected for each object, and for every trial, firing responses of 50 digital SA-I and 85 FA-I afferents were recorded for 4 s from the ZedBoard. Indeed, the spiking patterns of the 135 artificial tactile afferents were recorded for 180 trials (3 objects, 3 cases, 20 repetitions) to be analyzed by the machine learning algorithms.

Next, the machine learning approaches to interpret the recorded firing patterns are employed. In this way, first, feature extraction from spiking responses is accomplished using one of the fundamental coding paradigm for neural information processing, *rate coding*. The firing rate (FR) is defined by the number of spikes occurring at the time interval Δt, $FR = (spikes) / \Delta t$. Change of firing rate as the stimulus changes called *rate coding*. It is typically pointed out that sensory neurons transmit information by their firing rate. In this study, the decoding algorithm is based on the spike count; that is, different stimuli elicit a different number of spikes (Vreeken, 2003). Principal components analysis is exploited for dimension reduction. The first, three principal components are considered.

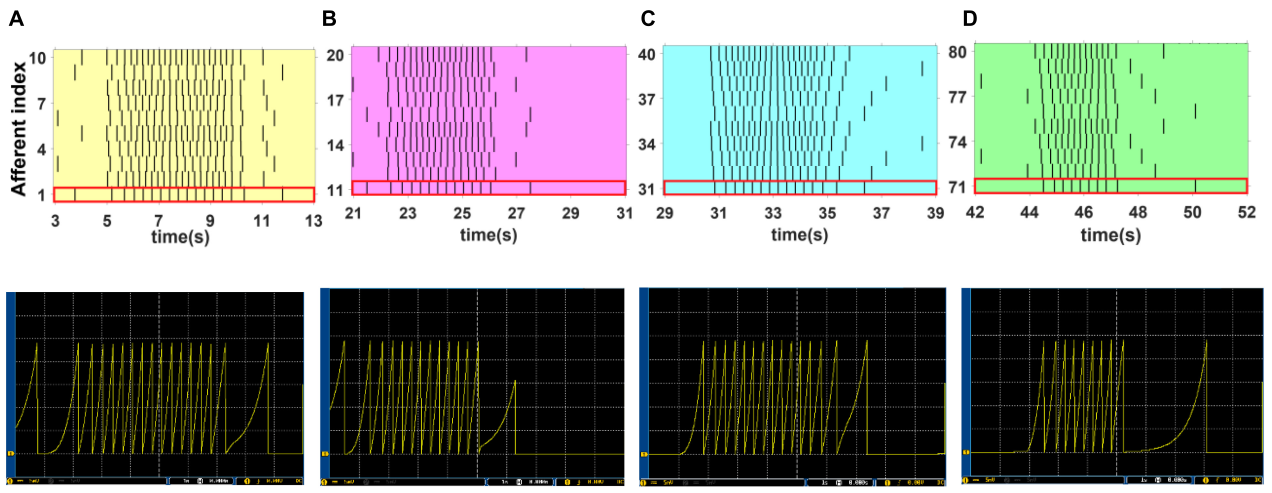


FIGURE 11 | The firing activity of 10 digital SA-I afferents implemented on the ZedBoard. The sustained firing to the input is clear. Considering **Figures 9**, **10**, yellow, magenta, cyan, and green illustrate the spiking activity arising from touching S1 **(A)**, S2 **(B)**, S4 **(C)** and S8 **(D)** respectively **(upper panels)**. We use a 16-bit DAC to convert the digital outputs of the ZedBoard to analog signals to be displayed on the oscilloscope screen **(lower panels)**. From each subpopulation, the first artificial afferent is chosen (the red rectangle in the upper panels) to be shown on the oscilloscope screen **(lower panels)**. The volt division was set on 5 mV.

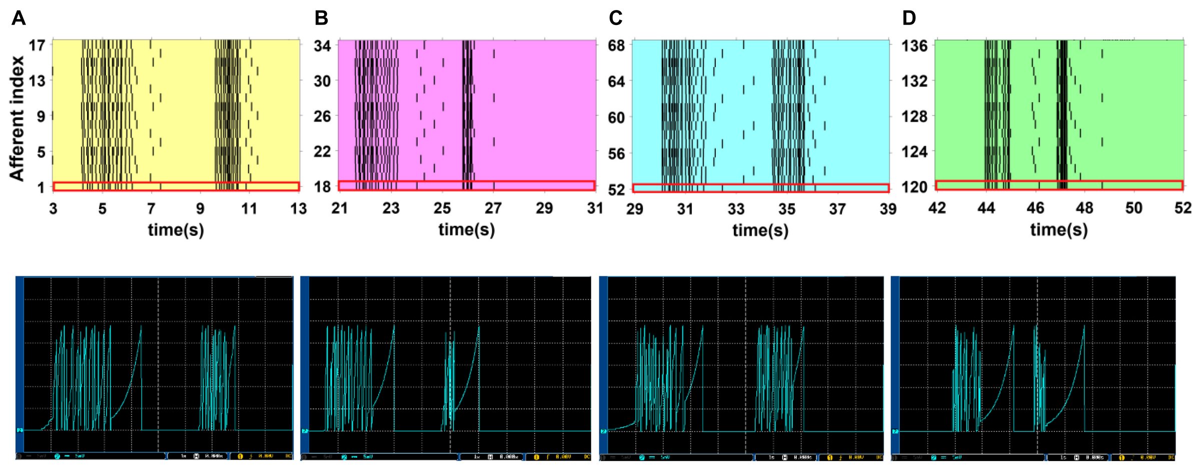


FIGURE 12 | The firing activity of 17 digital FA-I afferents implemented on the ZedBoard. Considering **Figures 9**, **10**, yellow, magenta, cyan, and green illustrate the spiking activity by touching S1 **(A)**, S2 **(B)**, S4 **(C)** and S8 **(D)**, respectively **(upper panels)**. Individual digital FA-I afferent fires during stimulus onset and offset and changes in input. We use a 16-bit DAC to convert the digital outputs of the ZedBoard to analog signals to be shown on the oscilloscope screen **(lower panels)**. From each subpopulation, the first artificial afferent is chosen (the red rectangle in the upper panels) to be displayed on the oscilloscope **(lower panels)**. The volt division was set on 5 mV.

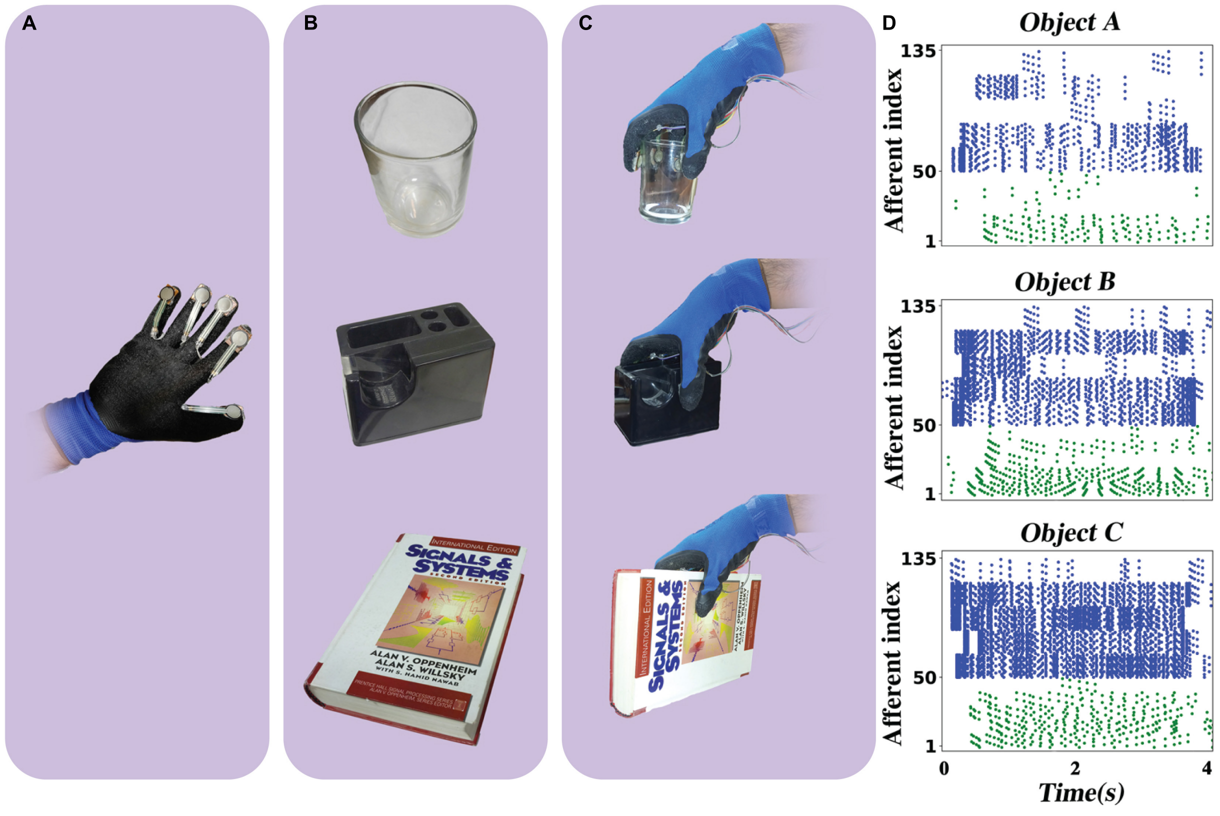


FIGURE 13 | The haptic experiment. **(A)** Five FSR sensors are attached to a glove. **(B)** Three objects: a glass, a tape dispenser, and a book. **(C)** The subject wears the glove and picks up every object and holds it for 3 s and then puts in the place. The subject repeats this experiment for 20 times with three fingers, four fingers, and five fingers, independently. The FSRs send their signal to the ZedBoard where it runs a population of 50 SA-I and 85 FA-I digital afferents. The firing patterns of the 135 artificial tactile afferents are recorded for 180 trials (3 objects, 3 cases, 20 repetitions) to be analyzed by the machine learning algorithms. **(D)** Sample of spike trains with five fingers. Green spikes show the response of the artificial SA-I afferents and blue spikes illustrate the response of artificial FA-I afferents.

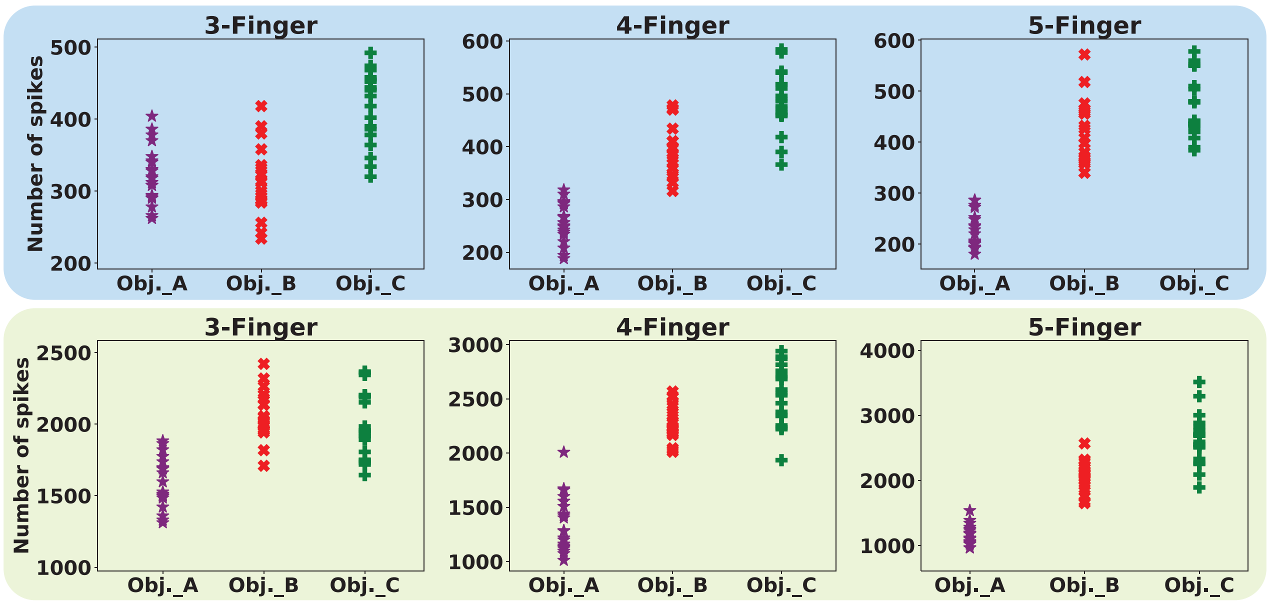


FIGURE 14 | Decoding based on the firing rate paradigm, in which different stimuli elicit a different number of spikes for the same time interval. Upper panels and lower panels indicate spike count for SA-I and FA-I afferents, respectively. Each point indicates one trial. Twenty trials were performed for individual object, which is indicated by a different color.

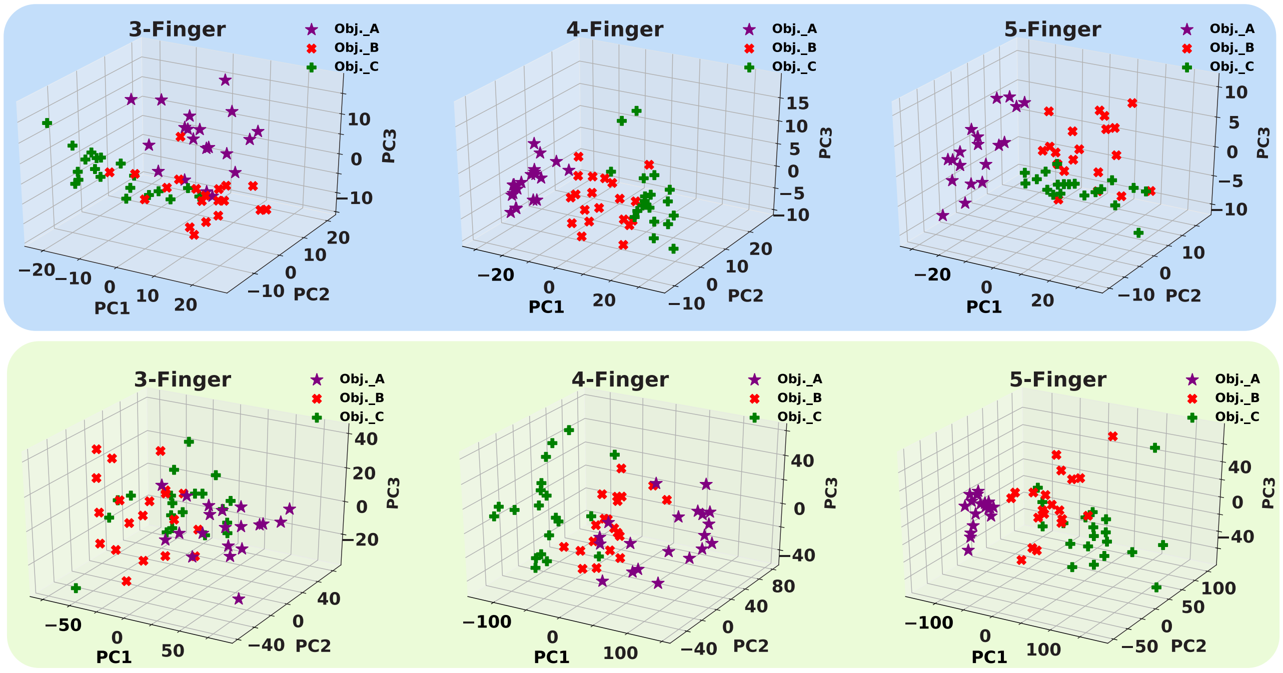


FIGURE 15 | The first three principal components (PCs) obtained from three haptic experiments. Upper panels show feature space for SA-I afferents, and lower panels illustrate feature space for FA-I afferents.

Figure 14 shows the spike count of the population of SA-I and FA-I afferents for three objects and three cases. Each point indicates one trial. Feature space of the first three principle components for all three experiments is illustrated in **Figure 15**.

Next, we report the classification performance of the k-Nearest Neighbor classifier using the obtained artificial spike trains. The classifier has three outputs: objects A, B, and C. The classifier input is the three principal components computed forming the

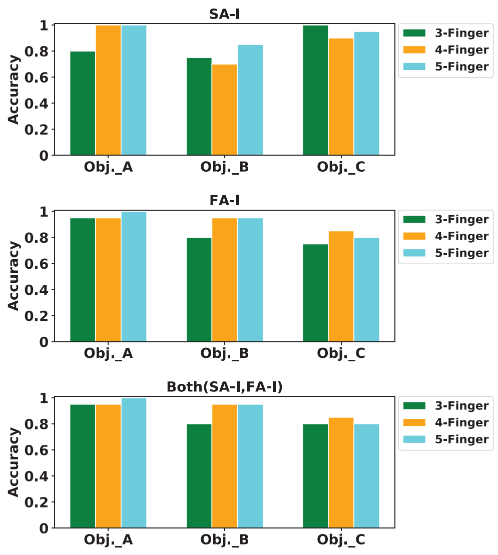


FIGURE 16 | The classification accuracy for k-Nearest Neighbor (kNN) classifier. Classification accuracy for digital SA-I **(top panel)**, for digital FA-I **(middle panel)**, and for both afferents, SA-I and FA-I **(bottom panel)**.

TABLE 5 | Mean and SD of classification accuracy for different experiment.

	3-Finger	4-Finger	5-Finger
SA-I	85% ± 3%	87% ± 8%	93% ± 6%
FA-I	83% ± 5%	92% ± 7%	92% ± 7%
Both (SA-I and FA-I)	85% ± 6%	92% ± 7%	92% ± 7%

TABLE 6 | Parameter values for the spiking models of the tactile afferents used for simulations.

Parameter	Value	Parameter	Value
a	0.02 s^{-1}	b	0.2 s^{-1}
c	−65 mV	d	8 mV
k_1	0.75 s^{-1}	k_2	20 mV s^{-1}
M_1	1 mV^{-1} s^{-1}	M_2	0.0625 s^{-1}
v_{peak}	30 mV	v_{reset}	0 mV
C_m	1 F	τ	1 s
C_{11}	20	C_{12}	960
C_{21}	24	C_{22}	960
C_{31}	0.015625	C_{32}	0.5
C_{41}	1	C_{42}	40

total number of spikes obtained for that stimulus. Different k values from 2 to 8 were tried. However, the results for $k = 5$ were reported in **Figure 16**. The value of k is important as a small k might result to a classifier sensitive to noise samples, and a large k can lead to less distinct boundaries among classes. The k-Nearest Neighbor is a non-parametric classifier that measures the difference between every spike train (ST) and other spike trains. The object was properly classified when the mean difference between the ST and spike trains from the same class was smaller than the mean difference between the ST and spike trains of other classes. This procedure was repeated for every ST obtained from digital afferents.

For classification, 80% of samples were randomly grouped as training set, and the remaining 20% samples were considered as the test set. K-fold cross-validation was also used. Indeed, the data samples are divided into K subsets. Each time, one of these K subsets is used as the validation set and the remaining (K - 1) subsets form a training set. Then, the average error across all K trials for each subset is computed (Hosseini et al., 2007). We used $K = 5$ for cross-validation. The feature vectors must be normalized in order to avoid distortions between features and numerical problems. Finally, the mean and SD of classification accuracy for this haptic experiment (**Figure 13C**) is reported in **Table 5**.

The developed system makes it possible to encode force information by a sequence of spikes, mimicking the neural dynamics of SA-I and FA-I afferents. Indeed, the recorded artificial spike trains from ZedBoard, which runs the SA-I/FA-I digital circuits, carry sufficient information. In this way, the input stimulus is discriminated even using a commercial FSR sensor. This technical approach is an innovative one for manufacturing sensory systems that artificially replicate the SA-I and FA-I firing activities to be employed in the bio-robotic and prosthetic application. The obtained spike trains are diverse and reliable enough to be able to decode the presented stimuli with high accuracy.

CONCLUSION

To obtain better performance and efficacy over traditional methods, recently, there is a tendency toward creating neuromorphic devices to mimic the biological systems. Software simulation and hardware realization of the SA-I and FA-I afferents might be considered as the neuromorphic approaches for restoring tactile feedback in upper limb prostheses. This methodology transmits tactile information more efficient, very similar to the healthy peripheral nervous system, to the next level, which can be the prosthesis controller. In this research, to digitally realize a population of 243 tactile afferents (90 SA-I and 153 FA-I) on FPGA, with emphasis on real-time functionality, a digital circuit was designed using an improved version of the L-QIF neural model. This model has been selected for the highest simplicity and lowest resource consumption of hardware implementation compared to the other model reported in this research. Using an experimental setup, we investigated the performance of the neuromorphic tactile system (comprising the SA-I and FA-I afferents) when it received multiple inputs simultaneously. Using a glove equipped with FSRs, we performed some haptic experiments and then we analyzed the spiking responses measured from the ZedBoard. Applying machine learning algorithm and considering firing rate coding, the picked up object was recognized with high accuracy from the recorded spike trains produced by the artificial tactile afferents.

Although we did not discuss the biological plausibility of the designed digital circuits, it was shown that they functionally follow the physiological observation, which is a basic step for moving forward. It should be mentioned that, whereas FSR transducers are integrated relatively easily with peripheral hardware and software, their application for mimicking mechanoreceptor response is not precise. In addition, a compliant skin-like layer should cover the FSR sensors. Finally, implementing a population of digital afferents might support the possibility for future development of new generation of tactile modules for prosthetic hands to reestablish sensory feedback for amputee. Moreover, the obtained spike trains from digital afferents may be further processed by the next level, which also can be done in hardware. This will make a neuromorphic sensory system for a mobile robot to accomplish various real-world tasks such as texture discrimination and object recognition.

DATA AVAILABILITY STATEMENT

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

AUTHOR CONTRIBUTIONS

NS-N, EI, MA, EF, and CL did conception, design and interpretation of the data, and drafting and revising the manuscript. NS, EI, and MA performed the experiments, acquired the data, and analyzed the data.

FUNDING

This work has received funding from the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No. 785907 (HBP SGA2). MA was supported by the Kermanshah University of Medical Sciences, Kermanshah, Iran.

ACKNOWLEDGMENTS

The authors would like to thank the esteemed reviewers for their insightful comments. NS-N and MA would like to thank Mr. Adel Parvizi-Fard for his valuable assistance with data analysis.

REFERENCES

Ambroise, M., Buccelli, S., Grassia, F., Pirog, A., Bornat, Y., Chiappalone, M., et al. (2017). Biomimetic neural network for modifying biological dynamics during hybrid experiments. *Artif. Life Robot.* 22, 398–403. doi: 10.1007/s10015-017-0366-1

Arthur, J. V., Merolla, P., Akopyan, F., Alvarez-Icaza, R., Cassidy, A., Chandra, S., et al. (2012). "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN)*, (Brisbane), 1–8.

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., and Chandrasekaran, A. R. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 699–716. doi: 10.1109/jproc.2014.2313565

Bologna, L., Pinoteau, J., Passot, J., Garrido, J., Vogel, J., Vidal, E. R., et al. (2013). A closed-loop neurobotic system for fine touch sensing. *J. Neural Eng.* 10:046019. doi: 10.1088/1741-2560/10/4/046019

Bologna, L. L., Pinoteau, J., Brasselet, R., Maggiali, M., and Arleo, A. (2011). Encoding/decoding of first and second order tactile afferents in a neurorobotic application. *J. Physiol. Paris.* 105, 25–35. doi: 10.1016/j.jphysparis.2011.08.002

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Cassidy, A., Andreou, A. G., and Georgiou, J. (2011). "Design of a one million neuron single FPGA neuromorphic system for real-time multimodal scene analysis," in *Information Sciences and Systems (CISS), 2011 45th Annual Conference on*, (Baltimore, MD), 1–6.

Dahiya, R. S., Metta, G., Valle, M., and Sandini, G. (2010). Tactile sensing—from humans to humanoids. *IEEE Trans. Robot.* 26, 1–20. doi: 10.1109/tro.2009.2033627

Dahiya, R. S., Mittendorfer, P., Valle, M., Cheng, G., and Lumelsky, V. J. (2013). Directions toward effective utilization of tactile skin: a review. *IEEE Sens. J.* 13, 4121–4138. doi: 10.1109/jsen.2013.2279056

Dahiya, R. S., and Valle, M. (2012). *Robotic Tactile Sensing: Technologies and System*. Springer: Science & Business Media.

Franceschi, M., Camus, V., Ibrahim, A., Enz, C., and Valle, M. (2017). "Approximate FPGA implementation of CORDIC for tactile data processing using speculative adders," in *2017 IEEE New Generation of Circuits and Systems Conference (NGCAS)*, (Genoa).

Friedl, K. E., Voelker, A. R., Peer, A., and Eliasmith, C. (2016). Human-inspired neurorobotic system for classifying surface textures by touch. *IEEE Robot. Autom. Let.* 1, 516–523. doi: 10.1109/lra.2016.2517213

Grassia, F., Kohno, T., and Levi, T. (2016). Digital hardware implementation of a stochastic two-dimensional neuron model. *J. Physiol. -Paris.* 110, 409–416. doi: 10.1016/j.jphysparis.2017.02.002

Hodgkin, A. L., and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 117, 500-544.

Hosseini, S. M., Amiri, M., Najarian, S., and Dargahi, J. (2007). Application of artificial neural networks for the estimation of tumour characteristics in biological tissues. *Int. J. Med. Robot. Comput. Assist. Surg.* 3, 235–244. doi: 10.1002/rcs.138

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Johansson, R. S., and Flanagan, J. R. (2009). Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nat. Rev. Neurosci.* 10, 345–359. doi: 10.1038/nrn2621

Johansson, R. S., and Vallbo, A. (1979). Tactile sensibility in the human hand: relative and absolute densities of four types of mechanoreceptive units in glabrous skin. *J. Physiol.* 286, 283–300. doi: 10.1113/jphysiol.1979.sp012619

Jörntell, H., Bengtsson, F., Geborek, P., Spanne, A., Terekhov, A. V., and Hayward, V. (2014). Segregation of tactile input features in neurons of the cuneate nucleus. *Neuron* 83, 1444–1452. doi: 10.1016/j.neuron.2014.07.038

Kim, S. S., Sripati, A. P., Vogelstein, R. J., Armiger, R. S., Russell, A. F., and Bensmaia, S. J. (2009). Conveying tactile feedback in sensorized hand neuroprostheses using a biofidelic model of mechanotransduction. *IEEE Trans. Biomed. Circuits Syst.* 3, 398–404. doi: 10.1109/TBCAS.2009.2032396

Lee, D., Lee, G., Kwon, D., Lee, S., Kim, Y., and Kim, J. (2018). "Flexon: a flexible digital neuron for efficient spiking neural network simulations," in *Proceedings - 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture, ISCA 2018*, (Los Angeles, CA).

Lee, W., Cabibihan, J., and Thakor, N. (2013). Bio-mimetic strategies for tactile sensing.in *SENSORS. IEEE* 2013, 1–4.

Lee, W. W., Yu, H., and Thakor, N. V. (2014). "Gait event detection through neuromorphic spike sequence learning," in *Biomedical Robotics and Biomechatronics 2014 5th IEEE RAS & EMBS International Conference on*, 899-904, (Atlanta, GA: IEEE).

Lucarotti, C., Oddo, C. M., Vitiello, N., and Carrozza, M. C. (2013). Synthetic and bio-artificial tactile sensing: a review. *Sensors* 13, 1435–1466. doi: 10.3390/s130201435

McGlone, F., and Reilly, D. (2010). The cutaneous sensory system. *Neurosci. Biobehav. Rev.* 34, 148–159. doi: 10.1016/j.neubiorev.2009.08.004

Misra, J., and Saha, I. (2010). Artificial neural networks in hardware: a survey of two decades of progress. *Neurocomputing* 74, 239–255. doi: 10.1016/j.neucom.2010.03.021

Nanami, T., and Kohno, T. (2016). Simple cortical and thalamic neuron models for digital arithmetic circuit implementation. *Front. Neurosci.* 10:181. doi: 10.3389/fnins.2016.00181

Oddo, C. M., Mazzoni, A., Spanne, A., Enander, J. M., Mogensen, H., Bengtsson, F., et al. (2017). Artificial spatiotemporal touch inputs reveal complementary decoding in neocortical neurons. *Sci. Rep.* 7:45898. doi: 10.1038/srep45898

Oddo, C. M., Raspopovic, S., Artoni, F., Mazzoni, A., Spigler, G., Petrini, F., et al. (2016). Intraneural stimulation elicits discrimination of textural features by artificial fingertip in intact and amputee humans. *Elife* 5:e09148. doi: 10.7554/eLife.09148

Osborn, L., Nguyen, H., Kaliki, R., and Thakor, N. (2017). Prosthesis grip force modulation using neuromorphic tactile sensing. in *Myoelectric Controls Symposium University of New Brunswic* (Baltimore), 188–191.

Osborn, L. E., Dragomir, A., Betthauser, J. L., Hunt, C. L., Nguyen, H. H., Kaliki, R. R., et al. (2018). Prosthesis with neuromorphic multilayered e-dermis perceives touch and pain. *Sci. Robot.* 3:eaat3818. doi: 10.1126/scirobotics.aat3818

Pasluosta, C., Kiele, P., and Stieglitz, T. (2017). Paradigms for restoration of somatosensory feedback via stimulation of the peripheral nervous system. *Clin. Neurophysiol.* 129, 851–862. doi: 10.1016/j.clinph.2017.12.027

Pearson, M., Nibouche, M., Gilhespy, I., Gurney, K., Melhuish, C., Mitchinson, B., et al. (2006). "A hardware based implementation of a tactile sensory system for neuromorphic signal rocessing applications," in *Acoustics, Speech and Signal Processing,. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 2006, IV-IV, (Toulouse).

Pearson, M. J., Mitchinson, B., Sullivan, J. C., Pipe, A. G., and Prescott, T. J. (2011). Biomimetic vibrissal sensing for robots. *Philos. Trans. R. Soc. B Biol. Sci.* 366, 3085–3096. doi: 10.1098/rstb.2011.0164

Pearson, M. J., Pipe, A. G., Mitchinson, B., Gurney, K., Melhuish, C., Gilhespy, I., et al. (2007). Implementing spiking neural networks for real-time signal-processing and control applications: a model-validated FPGA approach. *IEEE Trans. Neural Netwo.* 18, 1472–1487. doi: 10.1109/tnn.2007.891203

Rongala, U. B., Mazzoni, A., Camboni, D., Carrozza, M. C., and Oddo, C. M. (2018). "Neuromorphic artificial sense of touch: bridging robotics and neuroscience," in *Robotics Research*, eds A. Bicchi, and W. Burgard, (Cham: Springer International Publishing), 617–630. doi: 10.1007/978-3-319-60916-4_35

Rongala, U. B., Mazzoni, A., and Oddo, C. M. (2017). Neuromorphic artificial touch for categorization of naturalistic textures. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 819–829. doi: 10.1109/TNNLS.2015.2472477

Roudaut, Y., Lonigro, A., Coste, B., Hao, J., Delmas, P., and Crest, M. (2012). Touch sense: functional organization and molecular determinants of mechanosensitive receptors. *Channels* 6, 234–245. doi: 10.4161/chan.22213

Saal, H. P., and Bensmaia, S. J. (2015). Biomimetic approaches to bionic touch through a peripheral nerve interface. *Neuropsychologia* 79, 344–353. doi: 10.1016/j.neuropsychologia.2015.06.010

Saal, H. P., Delhaye, B. P., Rayhaun, B. C., and Bensmaia, S. J. (2017). Simulating tactile signals from the whole hand with millisecond precision. *Proc. Natl. Acad. Sci. U.S.A.* 114, E5693–E5702. doi: 10.1073/pnas.1704856114

Salimi-Nezhad, N., Amiri, M., Falotico, E., and Laschi, C. (2018). A digital hardware realization for spiking model of cutaneous mechanoreceptor. *Front. Neurosci.* 12:322. doi: 10.3389/fnins.2018.00322

Shlizerman, E., and Holmes, P. (2012). Neural dynamics, bifurcations, and firing rates in a quadratic integrate-and-fire model with a recovery variable. I: deterministic behavio. *Neural Comput.* 24, 2078–2118. doi: 10.1162/NECO_a_00308

Soleimani, H., Ahmadi, A., and Bavandpour, M. (2012). Biologically inspired spiking neurons: piecewise linear models and digital implementation. *IEEE Trans. Circuits Syst. I Regul. Pap.* 59, 2991–3004. doi: 10.1109/tcsi.2012.2206463

Spigler, G., Oddo, C. M., and Carrozza, M. C. (2012). "Soft-neuromorphic artificial touch for applications in neuro-robotics," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, 1913-1918, (Piscataway, NJ).

Tiwana, M. I., Redmond, S. J., and Lovell, N. H. (2012). A review of tactile sensing technologies with applications in biomedical engineering. *Sensors and Actuators A physical.* 179, 17–31. doi: 10.1016/j.sna.2012.02.051

Van Pottelbergh, T., Drion, G., and Sepulchre, R. (2018). Robust modulation of integrate-and-fire models. *Neural Comput.* 30, 987–1011. doi: 10.1162/neco_a_01065

Vreeken, J. (2003). *Spiking Neural Networks, an Introduction*. Utrecht: Utrecht University.

Wang, R. M., Thakur, C. S., and van Schaik, A. (2018). An FPGA-Based massively parallel neuromorphic cortex simulator. *Front. Neurosci.* 12:213. doi: 10.3389/fnins.2018.00213

Yi, Z., and Zhang, Y. (2016). Bio-inspired tactile FA-I spiking generation under sinusoidal stimuli. *J. Bionic Eng.* 13, 612–621. doi: 10.1016/s1672-6529(16)60332-3

Zhengkun, Y., and Yilei, Z. (2017). Recognizing tactile surface roughness with a biomimetic fingertip: a soft neuromorphic approach. *Neurocomputing* 244, 102–111. doi: 10.1016/j.neucom.2017.03.025

Zjajo, A., Hofmann, J., Christiaanse, G. J., van Eijk, M., Smaragdos, G., Strydis, C., et al. (2018). A real-time reconfigurable multichip architecture for large-scale biophysically accurate neuron simulation. *IEEE Trans. Biomed. Circuits Syst.* 12, 326–337. doi: 10.1109/TBCAS.2017.2780287

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.

APPENDIX

Izhikevich Neuron Model (Izh)

Integrate-and-fire (IF) neuron models are popular and simple to simulate, which help to be used in large network computational studies; however, they lack physiological interpretability. In contrast, conductance-based models with high biophysical realism are expensive to simulate since they often have high-dimensional non-linear differential equations. They require the tuning of many parameters and thus preventing their use in large networks.

Izhikevich proposed a model for spiking neuron that combines the response diversity of conductance-based models and the computational efficiency of IF neurons. The Izhikevich model (Izh) is described as follows (Izhikevich, 2003):

$$v' = 0.04v^2 + 5v + 140 - u + C_{11}\frac{I}{C_m} \tag{21}$$

$$u' = a\left(bv - u\right) \tag{22}$$

$$\text{if } v \geq 30 \text{ mV} \quad \rightarrow \quad \text{then} \quad \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{23}$$

v is the membrane potential of the neuron, I is the input current, and u is the membrane recovery variable. Constants a, b, c, and d are the neuron parameters. C_{11} scales the input current. C_m is capacitance value for dimensionality consistency. The parameters values of the Izh model, which were used in this research, are listed in **Table 6**.

Equations 21–23 are used to describe the spiking part of the SA-I model. Similarly, for FA-I model, the following mathematical model is used to obtain the output spike train.

$$v' = 0.04v^2 + 5v + 140 - u + C_{12}\frac{\tau}{C_m}I' \tag{24}$$

$$u' = a\left(bv - u\right) \tag{25}$$

$$\text{if } v \geq 30 \text{ mV} \quad \rightarrow \quad \text{then} \quad \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{26}$$

C_{12} is a constant factor that scales the input and τ is the time constant, and their values were reported in **Table 6**.

Linearized Izhikevich Neuron Model (L-Izh)

One solution to reduce high-cost mathematical operations is *linearization*. We use a piecewise-linear approximation of the Izh, which was presented in Soleimani et al. (2012). This L-Izh is described as follows:

$$v' = k_1\left|v + 62.5\right| - k_2 - u + C_{21}\frac{I}{C_m} \tag{27}$$

$$u' = a\left(bv - u\right) \tag{28}$$

$$\text{if } v \geq 30 \text{ mV} \quad \rightarrow \quad \text{then} \quad \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{29}$$

k_1 and k_2 are the constant values of the linearized model. C_{21} scales the neuron input. For the FA-I model, the L-Izh model can be used as follows:

$$v' = k_1\left|v + 62.5\right| - k_2 - u + C_{22}\frac{\tau}{C_m}I' \tag{30}$$

$$u' = a(bv - u) \tag{31}$$

$$\text{if } v \geq 30 \text{ mV} \quad \rightarrow \quad \text{then} \quad \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{32}$$

C_{22} is the constant coefficient for scaling the input current. The parameter values are listed in **Table 6**.

Quadratic Integrated & Fire Neuron Model (QIF)

Considerable research has been devoted to combine the economy of IF models with the physiological interpretability of conductance-based models. An example is the Quadratic Integrated and Fire (QIF) model that has the interpretation of a mathematical reduction of the conductance-based model of Hodgkin and Huxley (1952) (Van Pottelbergh et al., 2018). Several generalizations of the QIF model have been studied in the literature. Shlizerman and Holmes (2012) presented the QIF model with minimum computations as follows:

$$v' = M_1 v^2 + C_{31} \frac{I}{C_m} \tag{33}$$

$$\text{if } v \geq v_{\text{peak}} \quad \rightarrow \quad \text{then} \quad v = \ v_{\text{reset}} \tag{34}$$

M_1 and C_{31} are the constant coefficients. v_{peak} is the maximum value of membrane voltage, and v_{reset} is the rest membrane potential. All these parameters are reported in **Table 6**. Similarly, for FA-I model, we have:

$$v' = M_1 v^2 + C_{32} \frac{\tau}{C_m} I' \tag{35}$$

$$\text{if } v \geq v_{\text{peak}} \quad \rightarrow \quad \text{then} \quad v = \ v_{\text{reset}} \tag{36}$$

C_{32} scales the model input. Some neuron models such as the QIF model (Benjamin et al., 2014) and adaptive exponential integrate and fire model (AdEx model) (Brette and Gerstner, 2005) do not instantly produce a spike. These neuron models hire alternative non-instant functions, which control the membrane potential once it reaches the threshold voltage (Lee et al., 2018).

Linearized QIF Neuron Model (L-QIF)

Although the QIF model is a simple model, it is possible to use the linear approximation method to obtain a simpler model. Similar to the method used in Soleimani et al. (2012), the QIF model is linearized as follows for SA-I model:

$$v' = M_2 \left| v \right| + C_{41} \frac{I}{C_m} \tag{37}$$

$$\text{if } v \geq v_{\text{peak}} \quad \rightarrow \quad \text{then} \quad v = \ v_{\text{reset}} \tag{38}$$

M_2 and C_{41} are the constant coefficients. The linearized version of the QIF model for FA-I afferent is as follows:

$$v' = M_2 \left| v \right| + C_{42} \frac{\tau}{C_m} I' \tag{39}$$

$$\text{if } v \geq v_{\text{peak}} \quad \rightarrow \quad \text{then} \quad v = \ v_{\text{reset}} \tag{40}$$

C_{42} is the constant parameter. The parameter values reported in **Table 6** are taken from Izhikevich (2003); Shlizerman and Holmes (2012), Soleimani et al. (2012), and Rongala et al. (2018). The M2 value is selected to have minimum mean square error between the QIF spiking model and its linearized version. The parameters in last four rows adjusted by testing various values to obtain appropriate firing rate. Overall, high gain value causes a strong firing rate independent from the stimulus strength, and thus, the temporal structure of spikes is less informative. Conversely, low gain factors initiate low firing rate and accordingly a long latency in spike responses (Oddo et al., 2017). So, it is necessary to have a proper tradeoff.

frontiers in Neuroscience

ORIGINAL RESEARCH
published: 22 January 2020
doi: 10.3389/fnins.2019.01420

An Efficient and Perceptually Motivated Auditory Neural Encoding and Decoding Algorithm for Spiking Neural Networks

Zihan Pan[1], Yansong Chua[2], Jibin Wu[1], Malu Zhang[1], Haizhou Li[1] and Eliathamby Ambikairajah[3]*

[1] Department of Electrical and Computer Engineering, National University of Singapore, Singapore, Singapore, [2] Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore, Singapore, [3] School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW, Australia

OPEN ACCESS

Edited by:
Huajin Tang,
Zhejiang University, China

Reviewed by:
Sadique Sheik,
aiCTX AG, Switzerland
Rui Yan,
Sichuan University, China

****Correspondence:***
Yansong Chua
James4424@gmail.com

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *23 August 2019*
Accepted: *16 December 2019*
Published: *22 January 2020*

Citation:
Pan Z, Chua Y, Wu J, Zhang M, Li H and Ambikairajah E (2020) An Efficient and Perceptually Motivated Auditory Neural Encoding and Decoding Algorithm for Spiking Neural Networks. Front. Neurosci. 13:1420. doi: 10.3389/fnins.2019.01420

The auditory front-end is an integral part of a spiking neural network (SNN) when performing auditory cognitive tasks. It encodes the temporal dynamic stimulus, such as speech and audio, into an efficient, effective and reconstructable spike pattern to facilitate the subsequent processing. However, most of the auditory front-ends in current studies have not made use of recent findings in psychoacoustics and physiology concerning human listening. In this paper, we propose a neural encoding and decoding scheme that is optimized for audio processing. The neural encoding scheme, that we call Biologically plausible Auditory Encoding (BAE), emulates the functions of the perceptual components of the human auditory system, that include the cochlear filter bank, the inner hair cells, auditory masking effects from psychoacoustic models, and the spike neural encoding by the auditory nerve. We evaluate the perceptual quality of the BAE scheme using PESQ; the performance of the BAE based on sound classification and speech recognition experiments. Finally, we also built and published two spike-version of speech datasets: the Spike-TIDIGITS and the Spike-TIMIT, for researchers to use and benchmarking of future SNN research.

Keywords: spiking neural network, neural encoding, auditory perception, spike database, auditory masking effects

INTRODUCTION

The temporal or rate based Spiking Neural Networks (SNN), supported by stronger biological evidence than the conventional artificial neural networks (ANN), represents a promising research direction. Neurons in a SNN communicate using spiking trains that are temporal signals in nature, therefore, making SNN a natural choice for dealing with dynamic signals such as audio, speech, and music.

In the domain of rate-coding, we studied the computational efficiency of SNN (Pan et al., 2019). Recently, further evidence has supported the theory of temporal coding with spike times. To learn a temporal spike pattern, a number of learning rules have been proposed, which include the single-spike Tempotron (Gütig and Sompolinsky, 2006), conductance-based Tempotron (Gütig and Sompolinsky, 2009), the multi-spike learning rule ReSuMe (Ponulak and Kasiński, 2010;

Taherkhani et al., 2015), the multi-layer spike learning rule SpikeProp (Bohte et al., 2002), and the Multi-spike Tempotron (Gütig, 2016), etc. The more recent studies are aggregate-label learning (Gütig, 2016), and a novel probability-based multi-layer SNN learning rule (SLAYER) (Shrestha and Orchard, 2018).

In our research, a question is constantly asked: what are the advantages of SNN over ANN? From the viewpoint of neural encoding, we expect to encode a dynamic stimulus into spike patterns, which was shown to be possible (Maass, 1997; Ghosh-Dastidar and Adeli, 2009). Deep ANNs have benefited from the datasets created in recent years. In the field of image classification, there is ImageNet (Russakovsky et al., 2015); in the field of image detection, there is COCO dataset (Veit et al., 2016); while in the field of Automated Speech Recognition (ASR), there is TIMIT for phonemically and lexically transcribed speech of American English speakers (Garofolo, 1993). With the advent of these datasets, better and faster deep ANNs inevitably follow (Hochreiter and Schmidhuber, 1997; Simonyan and Zisserman, 2014; Redmon et al., 2016). The publicly available datasets become the common platform for technology benchmarking. In the study of neuromorphic computing, there are some datasets such as N-MNIST (Orchard et al., 2015), DVS Gestures (Amir et al., 2017), and N-TIDIGITS (Anumula et al., 2018). They are designed for SNN benchmarking. However, these datasets are relatively small compared with the deep learning datasets.

One may argue that the benchmarking datasets for deep learning may not be suitable for SNN studies. Let us consider image classification as an example. Humans process static images in a similar way as they would process live visual inputs. We note that live visual inputs contain much richer information than 2-D images. When we map (Rueckauer et al., 2017) or quantize (Zhou et al., 2016) static images into spike trains, and compare the performance of an ANN on static images, and a SNN on spike trains, we observe an accuracy drop. One should, however, not hastily conclude that SNNs are inherently poor in image classification as a consequence of event-based activations in SNNs. Rather, the question seems to be: how can one better encode images into spikes that are useful for SNNs, and how can one better use these spikes in an image classification task? For some of the recent image-based neuromorphic datasets, Laxmi et al. (Iyer et al., 2018) has argued that no additional information is encoded in the time domain that is useful for pattern classification. This prompts us to look into the development of event-based datasets that inherently contain spatio-temporal information. On the other hand, a dataset has to be complex enough such that it simulates a real-world problem. There are some datasets that support the learning of temporal patterns (Zhang et al., 2017, 2018, 2019; Wu et al., 2018a), whereby each pattern contains only a single label, such as a sound event or an isolated word. Such datasets are much simpler than those in deep learning studies (Graves et al., 2006), whereby a temporal pattern involves a sequence of labels, such as continuous speech. For SNN study to progress from isolated word recognition toward continuous speech recognition, a continuous speech database is required. In this paper, we would describe how we convert the TIMIT dataset to its event-based equivalent: Spike-TIMIT.

A typical pattern classification task consists of three stages: encoding, feature representation, and classification. The boundaries between each stage are getting less clear in an end-to-end classification neural network. Even then, a good encoding scheme can significantly ease the workload of the subsequent stages in a classification task, for instance, the Mel-Frequency Cepstral Coefficients (MFCC) (Mermelstein, 1976) is still very much in use for automatic speech recognition (ASR). Hence the design of a spiking dataset should consider how the encoding scheme could help reduce the workload of the SNN in a classification task. This cannot be misconstrued as giving the SNN an unfair advantage so long as all SNNs are measured using the same benchmark. The human cochlea performs frequency filtering (Tobias, 2012) while human vision performs orientation discrimination (Appelle, 1972). These all involve encoding schemes to help us better understand our environment. In our earlier work (Pan et al., 2019), on a simple dataset TIDIGITS (Leonard and Doddington, 1993) that contains only single spoken digits, we used a population threshold coding scheme to encode the dataset into events, which we refer to as Spike-TIDIGITS. Using such an encoding scheme, we go on to show that the dataset becomes linearly separable, i.e., the input can be classified based on spike counts alone. This demonstrates that when information is encoded in both the temporal (spike timing) and spatial (which neuron to spike) domain, the encoding scheme is able to project the inputs to a higher dimension, that takes some of the workload off the subsequent feature extraction and classification stages. In the case of Spike-TIDIGITS, the spikes encoded can be directly counted and then classified using a Support Vector Machine (SVM). Using this neural encoding scheme, We further enhance it and then apply it to the TIMIT dataset in this work.

The motivation of this paper is two-fold. Firstly, we believe that we need well-designed spike-encoded datasets that represent the state-of-the-art encoding methodology. With these datasets, one can focus the research on SNN feature representation and classification tasks. Secondly, the datasets should present a challenge in pattern classification, that become the reference benchmark in future SNN studies.

As speech is the most common way of human communication, we are looking into the neural encoding of speech signals in this work. The first question is how best possible to convert speech signals into spikes. There have been many related studies in speech and audio encoding, each of which is optimized for a specific objective, for example, the minimum signal reconstruction error (Loiselle et al., 2005; Dennis et al., 2013; Xiao et al., 2016). However, the speech and audio encoding methods have not taken into consideration the combination of psychoacoustic effects, computational efficiency, and pattern classification performance for neuromorphic implementation. In the SNN applications for speech recognition (Xiao et al., 2016; Darabkh et al., 2018), MFCC (Mermelstein, 1976) are commonly used as the spectral representation in speech recognition. Others have tried to use the biologically plausible cochlear filter bank, but they are either analog filters which are prone to changes in the external environment (Liu and Delbruck, 2010), or yet to be studied in a spike-driven SNN system (Loiselle et al., 2005).

Yang et al. (2016) successfully implements a silicon cochlear for event-driven audio sensing, which has not considered the psychoacoustics of the auditory system.

Considering spectral representation, an important step in neural encoding is to then convert the spectral energy in a perceptual frequency band into a spike train. The most common way is to treat the two-dimensional time-frequency spectrogram as a static image, then converting each "pixel" value into a spike latency time within the framing window size (Wu et al., 2018a), or into the phase of the sub-threshold membrane potential oscillation (Nadasdy, 2009). Such "pixel-conversion" methods do not represent the spatio-temporal dynamics of the auditory signals in the same way as the spike trains in a SNN, therefore, another feature representation step is required, such as the self-organizing map (Wu et al., 2018b), or local spectrogram features (Dennis et al., 2013; Xiao et al., 2016). If the audio encoding is able to capture the spatio-temporal dynamics that are discriminative for classification (Gütig and Sompolinsky, 2009), it is not necessary to encode every speech frame in the front-end, therefore, the spiking rate can be reduced. Finally, it has not been given enough attention as to how to reconstruct a neural encoded speech signal back into its auditory signals for perceptual evaluation. Speech signal reconstruction is a critical task in speech information processing, such as speech synthesis, singing synthesis, and dialogue technology.

To address the need for neuromorphic computing for speech information processing, we propose three criteria for a biologically plausible auditory encoding (BAE) front-end:

(1) Biologically plausible spectral features.
(2) Sparse and energy-efficient spike neural coding scheme.
(3) Friendly for temporal learning algorithms on cognitive tasks.

The fundamental research problem in neural encoding is how to encode the dynamic and continuous speech signals into discrete spike patterns. Spike rate code is thought to be less likely in an auditory system since much evidence suggests otherwise, for example, how bats rely highly on the precise spike timing of their auditory system to locate sound sources by detecting a time difference as short as 5 μs. Latency code and phase code are well supported by neuro-biological observations. However, on its own, they cannot provide an invariant representation of the patterns for a classification task.

To facilitate the processing of an SNN in a cognitive task, neural temporal encoding should not only consider how to encode the stimulus into spikes, but also care about how to represent the invariant features. Just like the auditory and visual sensory representations in the human prefrontal cortex, such representations in the proposed BAE front-end are required in an SNN framework, that can then be implemented with a low-cost neuromorphic solution, that can effectively reduce the processing workload in the subsequent SNN pipeline. A large number of observations in neuroscience support the observation that our auditory sensory neurons encode the input stimulus using threshold crossing events in a population of sensory neurons (Ehret, 1997; Hopfield, 2004). Inspired by these observations, a simple version of threshold coding has been proposed (Gütig and Sompolinsky, 2009), in which a population of encoding neurons with a set of uniformly distributed thresholds encode the spectral energy of different frequency channels into spikes. Such a cross-and-fire mechanism is reminiscent of quantization from the point of view of information coding. In our proposed BAE encoding front-end, such a neural coding scheme is also being incorporated. Further investigation is presented in the "Experiment and Results" section.

Besides effective neural coding representation, an efficient auditory front-end aims to encode acoustic signals into sparse spike patterns, while maintaining sufficient perceptual information. To achieve such a goal, our biological auditory system has provided us a solution best understood as masking effects (Harris and Dallos, 1979; Shinn-Cunningham, 2008). The auditory masking is a complex and yet to be fully understood psychoacoustic phenomenon as some components of the acoustic events are not perceptible in both frequency and time domain (Ambikairajah et al., 1997). From the viewpoint of perceptual coding, these components are regarded as redundancies since they are inaudible. Implementing the masking effects, those inaudible components will be coded with larger quantization noise or not coded at all. Although the mechanism and function of masking are not yet fully understood, its effects have already been successfully exploited in auditory signal compression and coding (Ambikairajah et al., 2001), for efficient information storage, communication, and retrieval. In this paper, we propose a novel idea to apply the auditory masking effects in both frequency and time domain, which we refer to as simultaneous masking and temporal masking, respectively, in our auditory neural encoding front-end so as to reduce the number of encoding spikes. This improves the sparsity and efficiency of our encoding scheme. Given how we address the three optimization criteria of neural encoding, we refer to it as BAE scheme or BAE. Such an auditory encoding front-end also provides an engineering platform to bridge the study of masking effects between psychoacoustics and speech processing.

Our main contributions in this paper are: (1) we emphasize the importance of spike acoustic datasets for SNN research. (2) we propose an integrated auditory neural encoding front-end to further research in SNN-based learning algorithms. With the proposed BAE encoding front-end, the speech or audio datasets can be converted into energy-efficient, information-compact, and well-representative spike patterns for subsequent SNN tasks.

The rest of this paper is organized as follows: in section "Materials and Methods" we discuss the auditory masking effects, and how simultaneous masking in the frequency domain and temporal masking in the time domain for neural encoding of acoustic stimulus is being implemented; the BAE encoding scheme is applied in conjunction with masking to RWCP, TIDIGITS, and TIMIT datasets. In section "Experiment and Results," we describe the details of the resulting spike datasets and evaluate them against their original datasets. We discuss our findings in section "Discussion" and conclude in section "Conclusion."

MATERIALS AND METHODS

Auditory Masking Effects

Most of our speech processing front-ends employ a fixed feature extraction mechanism, such as MFCC, to encode the input signals, whereas the human auditory sensory system ignores some while strongly emphasizes others, commonly referred to as the attention mechanism in psychoacoustics. The auditory masking effects closely emulate this phenomenon (Shinn-Cunningham, 2008).

Auditory masking is a known perceptual property of the human auditory system that occurs whenever the presence of a strong audio signal makes its neighborhood of weaker signals inaudible, both in the frequency and time domain. One of the most notable applications of auditory masking is the MPEG/audio international standard for audio signal compression (Ambikairajah et al., 2001; Fogg et al., 2007). It compresses the audio data by removing the acoustically inaudible elements, or by encoding those parts with less number of bits, due to more tolerance to quantization noise (Ambikairajah et al., 1997). To achieve such a goal, the algorithm is supported by two different kinds of auditory maskings according to the psychoacoustic model (Lagerstrom, 2001):

1. In the frequency domain, two kinds of masking effects are used. Firstly, by allocating the quantization noise in the least sensitive regions of the spectrum, the perceptual distortion caused by quantization is minimized. Secondly, an absolute hearing threshold is exploited, below which the spectral components are entirely removed.
2. In the time domain, the masking effect is applied such that the local peaks of the temporal signals in each frequency band will make their ensuing audio signals inaudible.

Motivated by the above signal compression theory, we propose an auditory masking approach to spike neural encoding, which greatly increases the coding efficiency of the spike patterns, by eliminating those perceptually insignificant spike events. The approach is conceptually consistent with the MPEG-1 layer III signal compression standard (Fogg et al., 2007), with modifications according to the characteristics of spiking neurons.

Simultaneous Masking

The masking effect presented in the frequency domain is referred to as simultaneous masking. According to the MPEG-1 standards, there are two sorts of masking strategies in the frequency domain: the absolute hearing threshold and the frequency masking. The simultaneous masking effects are common in our daily life. For instance, the sensible sound levels of our auditory systems vary in different frequencies, therefore, we can be more sensitive to the sounds in our living environment. This is an evolutionary advantage for survival, in both human beings and animals. Besides the absolute hearing threshold, every acoustic event in the spectrum will also influence the perception of the neighboring frequency components, that is, different levels of tones could contribute to masking effects of other frequency tones. For instance, in a symphony show, the sounds from different musical instruments can be fully or partially masked by each other. As a result, we can enjoy the compositions of various frequency components with rich diversities. Such a psychoacoustic phenomenon is called frequency masking.

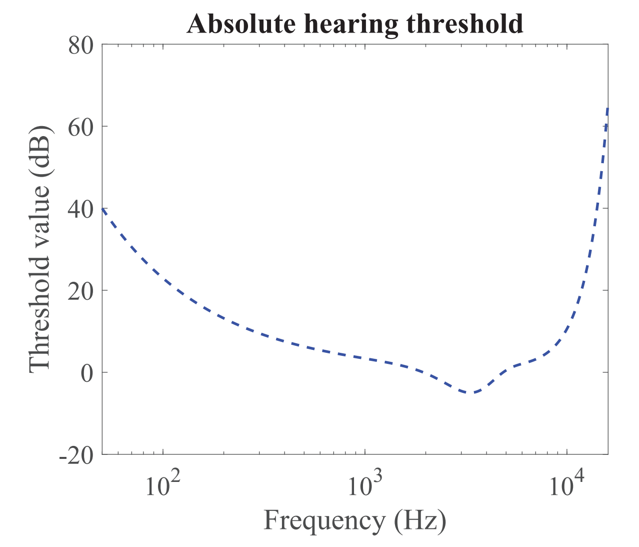


FIGURE 1 | Absolute hearing threshold T_a for the simultaneous masking. Our hearing is more sensitive to the acoustic stimulus around several thousand Hz, which covers the majority of the sounds in our daily life. The sounds below the thresholds are completely inaudible.

Figure 1 illustrates the absolute hearing threshold, T_a, as a function of frequency in Hz. The function is derived from psychoacoustic experiments, in which pure tones continuous in the frequency domain are presented to the test subjects and the minimal audible sound pressure levels (SPL) in dB are recorded. The commonly used function to approximate the threshold is (Ambikairajah et al., 1997):

$$T_a(f) = 3.64 \times \left(\frac{f}{1000}\right)^{-0.8} - 6.5 \times e^{-0.6\left(\frac{f}{1000}-3.3\right)^2} + 0.001 \times \left(\frac{f}{1000}\right)^4 \quad (1)$$

For the frequency masking, in the MPEG-1 standard, some sample pulses under masking thresholds might be partially masked, thus they are encoded by a lower number of bits. However, in the event-based scenario, spike patterns carry no amplitude information, similar to on-off binary values, which means that partial masking can hardly be realized. As such, we have modified the approach such that all components under the frequency masking are fully masked (discarded). Further reconstruction and pattern recognition experiments are necessary to evaluate such an approach. **Figure 2** shows the overall masking thresholds with both masking strategies in the frequency domain. This figure illustrates the simultaneous masking thresholds added to the acoustic events in a spectrogram. The sound signals with different spectral power in different cochlear filter channels will suffer from various masking thresholds.

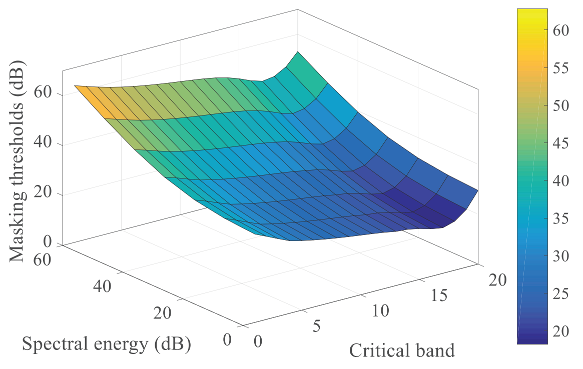


FIGURE 2 | The frequency masking thresholds acting on a maskee (the acoustic events being masked), generated by the acoustic events from the neighboring critical bands, are shown as a surface in a 3-D plot. The acoustic events are referred to as the spectral power of the frames in a spectrogram. The spectral energy axis is the sound level of a maskee; the critical band axis is the frequency bins of the cochlear filter bank, as introduced in section "Spike-TIDIGITS and Spike-TIMIT Databases"; the masking thresholds axis indicates the overall masking levels on the maskees of different sound levels from various critical bands. For example, an acoustic event of 20dB level on the 10th critical band is masked off by the masking threshold of nearly 23dB, which is introduced by the other auditory components of its neighboring frequency channels.

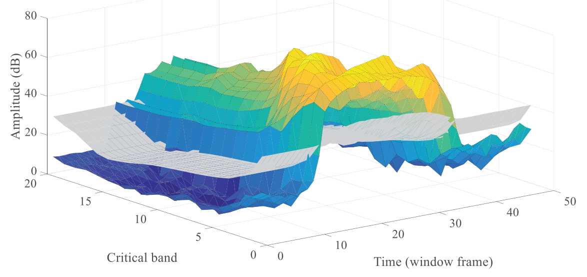


FIGURE 3 | The overall simultaneous masking effects on a speech utterance of "one," in a 3-D spectrogram. Combining the two kinds of masking effects in the frequency domain (refer to **Figures 1**, **2**), the gray surface shows the overall masking thresholds on a speech utterance (the colorful surface). All the spectral energy under the thresholds will be imperceptible.

Figure 3 provides a real-world example of the simultaneous masking. The spectrogram of a speech utterance of "one" from the TIDIGITS dataset is demonstrated in a 3-D plot. The gray surface illustrates the simultaneous masking threshold acting on the spectrogram (colorful surface). By the masking strategy, the acoustic events with spectral energy lower than the threshold surface will be removed. Section "Biologically Plausible Auditory Encoding With Masking Effects" will introduce how to convert the masked spectrogram into a sparse and well-represented spike pattern.

Temporal Masking

Another auditory masking effect is temporal masking in the time domain. Conceptually similar to the frequency masking, a louder sound will mask the perception of the other acoustic components in the time domain. As illustrated in **Figure 4**, the vertical bars represent the signal intensity of short-time frames, that is called acoustic events, along the time axis. A local peak (the first red bar) forms a masking threshold that makes the following events inaudible until the next local peak (the second red bar) exceeds the masking threshold. According to the psychoacoustic studies, the temporal masking threshold is modeled as an exponentially decaying curve (Ambikairajah et al., 2001):

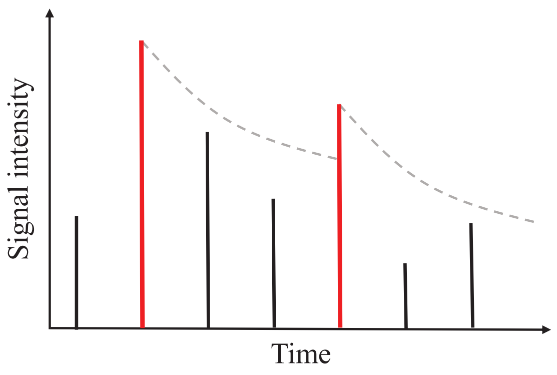


FIGURE 4 | The illustration of temporal masking: each bar represents the acoustic event received by the auditory system. In this paper, acoustic events generally referred to framing spectral power, which are the elements to be parsed to an auditory neural encoding scheme. A local peak event (red bar) forms a masking threshold represented by an exponentially decaying curve. The subsequent events that are weaker than the leading local peak will not be audible until another local peak event exceeds the masking threshold.

$$y(n) = c^n \times p_1 \tag{2}$$

where $y(n)$ denotes the masking threshold level on the n^{th} following an acoustic event; c is the exponential index and p_1 represents the sound level of the local peak as the beginning of the masking. The decaying parameter c is tuned according to the hearing quality.

Auditory Masking Effects in Both Domains

By applying both the simultaneous masking and temporal masking illustrated above, we can remove those imperceptible acoustic events (frames) from the overall spectrogram. Since our goal is to apply the masking effects in the precise timing neural code, we propose the strategy as follows:

1. The spike pattern $\mathbf{P}_{K\times N}(p_{ij})$ is generated from the raw spectrogram $\mathbf{S}_{K\times N}(s_{ij})$ without masking effects, by some temporal neural coding methods, which will be discussed in section "Neural Spike Encoding" Here the index i, j refers to the time-frequency bin in the spectrogram, with i referring to the frequency bin, and j referring to the time frame index. The spike pattern $\mathbf{P}_{K\times N}$ is defined as a matrix that:

$$p_{ij} = \begin{cases} t_f, & \text{if a spike is emitted within the duration of the} \\ & \text{time-frequency bin } i,\ j. \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where t_f is the encoded precise spike timing. As such the spike pattern $\mathbf{P}_{K\times N}\left(p_{ij}\right)$ is a sparse matrix that records the spike timing.

2. According to the spectrogram $\mathbf{S}_{K\times N}\left(s_{ij}\right)$ and the auditory perceptual model, the simultaneous masking threshold matrix $\mathbf{M}_{\text{simultaneous}}\left(m_{ij}^{\text{simultaneous}}\right)$ and the temporal masking threshold matrix $\mathbf{M}_{\text{temporal}}\left(m_{ij}^{\text{temporal}}\right)$ are obtained. The overall masking threshold $\mathbf{M}_{K\times N}\left(m_{ij}\right)$ is defined as follows. It provides a 2-D masking threshold surface that has the same dimensions as the spectrogram.
$$m_{ij} = \max\left\{m_{ij}^{\text{simultaneous}}, m_{ij}^{\text{temporal}}\right\} \tag{4}$$
3. A masking map $\Phi_{K\times N}\left(\phi_{ij}\right)$ is generated, whose dimensions are the same as the spectrogram. The element of the matrix $\Phi_{K\times N}\left(\phi_{ij}\right)$ is defined as:
$$\phi_{ij} = \begin{cases} 1, \text{ if } s_{ij} \geq m_{ij} \\ 0, \text{ if } s_{ij} < m_{ij} \end{cases} \tag{5}$$
where the time-frequency bin i,j is masked with $\phi_{i,j} = 0$ when the frame energy $s_{i,j}$ is less than the masking threshold m_{ij}, otherwise, $\phi_{i,j} = 1$.
4. Apply the masking map matrix $\Phi_{K\times N}\left(\phi_{ij}\right)$ to the encoded pattern $\mathbf{P}_{K\times N}\left(p_{ij}\right)$ to generate a masked spike pattern $\mathbf{P}^{\text{mask}}(p_{ij}^{\text{mask}})$:
$$\mathbf{P}^{\text{mask}} = \mathbf{P}_{K\times N} \circ \Phi_{K\times N} \tag{6}$$
where $\circ$ denotes the Hadamard product. By doing so, those perceptually insignificant spikes are eliminated, thus forming a more compact and sparse spike pattern.

Figure 5 demonstrates the auditory masking effects acting in both the frequency and time domains, on a speech utterance of "one" in the TIDIGITS dataset. The colored surface represents the original spectrogram while the gray areas represent the spectral energy values that are being masked. For TIDIGITS datasets, nearly half of the acoustic events (frames) are removed according to our auditory masking strategy, which corresponds to the 55% removal of PCM pulses in speech coding (Ambikairajah et al., 2001).

Cochlear Filters and Spike Coding

The human auditory system is primarily a frequency analyzer (Tobias, 2012). Many studies have confirmed the existence of the perceptual centre frequencies and equivalent bandwidths. To emulate the working of the human cochlea, several artificial cochlear filter banks have been well studied: GammaTone filter bank (Patterson et al., 1987; Hohmann, 2002), Constant Q Transform-based filter bank (CQT) (Brown, 1991; Brown and Puckette, 1992), Bark-scale filter bank (Smith and Abel, 1999), etc. They share the same idea of logarithm distributed centre frequencies and constant Q factors but slightly differ in the exact parameters. To build the auditory encoding system, we adopt an event-based CQT-based filter bank in the time domain, following our previous work (Pan et al., 2018).

Time-Domain Cochlear Filter Bank

Adopting an event-based approach to emulate the human auditory system, we propose a neuronal implementation of the event-driven cochlear filter bank, of which the computation can be parallelized as follows,

- As illustrated in **Figure 6**, a speech waveform (**Figure 6A**) is filtered by K neurons (**Figure 6B**) where each neuron represents one cochlear filter from a particular frequency bin.
- The weights of each neuron in **Figure 6B** are set as the time-domain impulse response of the corresponding cochlear filter. The computing of a neuron with its input is inherently a time-domain convolution process.
- The output of the filter bank neurons is a K-length vector (**Figure 6C**), where K is the number of filters, for each time step. Since the signal (**Figure 6A**) shifts sample by sample, the width of the output matrix is the same as the length of the input signal. As such, the auditory signal is decomposed into multiple channels in parallel, forming a spectrogram.

Suppose a speech signal $\boldsymbol{x}$ with M samples $\boldsymbol{x} = [x_1, x_2, ..., x_M]$ sampled at 16 kHz. For the k^{th}cochlear filter, the impulse response (wavelet) is a M_k-length vector $\mathbf{F}_k = [F_k(1), F_k(2), ..., F_k(M_k)]$. We note the impulse response $\mathbf{F}_k$ has an infinite window size, however, numerically its amplitude decreases to small values outside an effective window, thus having little influence on the convolution results. As investigated in Pan et al. (2018), we empirically set M_k to an optimal value. So the m^{th} output of the k^{th} cochlear filter neuron is modeled as $y_k(m)$:
$$y_k(m) = \sum_{i=1}^{M_k} \phi_m(i)F_k(i),\ k = 1, 2, ..., K,\ m = 1, 2, ..., M \tag{7}$$
$$\phi_m = [x_m, x_{m+1}, x_{m+2}, ..., x_{m+M_k-1}],\ m \in 1, ..., M \tag{8}$$
ϕ_m is a subset of the input samples within the m^{th} window, whose length is the same as that of the M_k-length wavelet, indicated as the samples between the two arrows in **Figures 6A,B**. The window ϕ_m will move sample by sample, naturally along with the flow of the input signal samples. At each time step, a vector of length K, which is the number of filters, is generated as shown in **Figure 6C**. After M such samples, the final output time-frequency map of the filter bank is a $K \times M$ matrix $\mathbf{Y}_{K\times M}$.

After time-domain cochlear filtering, the $K \times M$ time-frequency map $\mathbf{Y}_{K\times M}$ should be framed, which emulates the signal processing of hair cells in the auditory pathway. For the output waveform from each channel, we apply a framing window of length l (samples) with a step size of $l/2$ and calculate the logarithmic frame energy e of one framing window:
$$e = 10\log\left(\sum_{q=1}^{l} x_q^2\right) \tag{9}$$
where x_q denotes the samples within the l-length window; e is the spectral energy of one frame, hence obtaining the

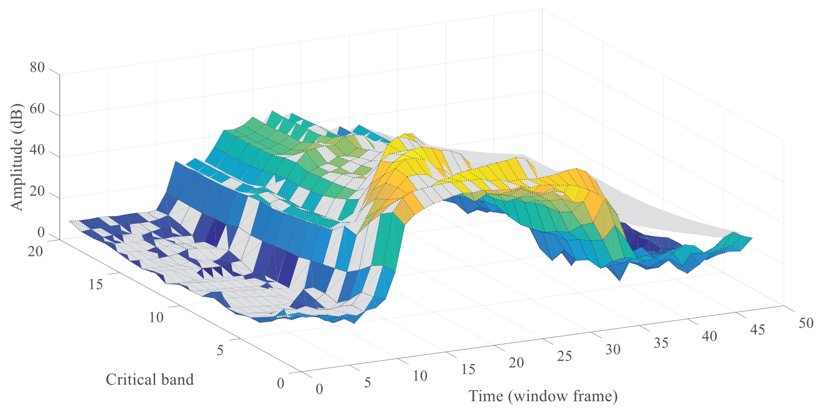


FIGURE 5 | Both the simultaneous and temporal masking effects acting on the 3-D plot spectrogram of a speech utterance of "one." The gray-color shaded parts of the spectrogram are masked.

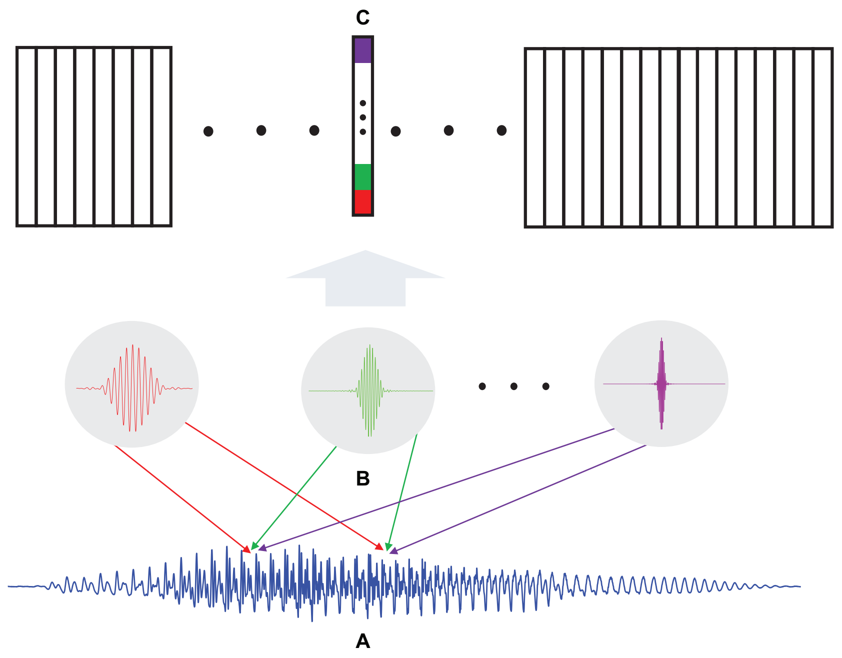


FIGURE 6 | (A) A speech signal of *M* samples; **(B)** Time-domain filter bank with *K* neurons that act as filters; **(C)** The output spectrogram that has $K \times M$ dimension.

time-frequency spectrum $\mathbf{S}_{K \times N}$ (s_{ij}) as indicated in section "Auditory Masking Effects in Both Domains" which will be further encoded into spikes.

Neural Spike Encoding

In the inner ear, the motion of the stereocilia in the inner hair cells is converted into a chemical signal that excites adjacent nerve fibers, generating neural impulses that are then transmitted along the auditory pathway. Similarly, we would like to convert the sub-band framing energy into electrical impulses, or so-called spikes, for the purpose of information encoding and transmission. In the prior work, the temporal dynamic sequences are encoded using several different methods: latency coding (Wu et al., 2018a), phase coding (Arnal and Giraud, 2012; Giraud and Poeppel, 2012), latency population coding (Dean et al., 2005), that are adopted for specific applications. These encoding schemes are not optimized for neuromorphic implementation.

We would like to propose a biologically plausible neural encoding scheme by taking into account the three criteria as defined in section "Introduction." In this section, the particular neural temporal coding scheme, which converts perceptual spectral power to precise spike times, is designed to meet the need of synaptic learning rules in SNNs (Gütig and Sompolinsky, 2006; Ponulak and Kasiński, 2010). As such, the resulting temporal spike patterns are supposed to be friendly toward temporal learning rules.

In our previous work (Pan et al., 2019), two mainstream neural encoding schemes, the single neuron temporal codes (latency coding, phase coding) and the population codes (population latency/phase coding, threshold coding) are compared. It is found that the threshold coding outperforms the other coding schemes in SNN-based pattern recognition tasks. Next are some observations made while comparing threshold coding, and the single neuron temporal coding.

First of all, the single temporal coding scheme, such as latency or phase coding, encodes the spectral power using spike delaying time, or phase-locking time. Suppose a frame of normalized spectral power is e, the n^{th} latency spike timing t_n^f is defined as:

$$t_n^f = (1-e) * T + (n-1) * T = (n-e) * T \tag{10}$$

where T denotes the time duration of the encoding window. For the phase coding, t_n^f is phase-locked to the nearest peak of the sub-threshold membrane oscillation. The spectral power, that represents the amplitude information, e is represented as the relative spike timing $(1-e) * T$ within each window and the number of spikes embedded are in the order n. Unfortunately, the SNN can hardly decode such an encoding scheme without the knowledge of the encoding window boundaries, implicitly provided by the spike order n and window length T. The spatio-temporal spike patterns could not provide such knowledge explicitly to the SNN. On the other hand, in the population code, such as threshold coding, the multiple encoding neurons naturally represent the amplitudes of the spectral power frames, and we only need to represent the temporal information in the spike timing. For example, the spike timing of the n^{th} onset encoding neuron of the threshold code t_f^n is:

$$t_f^n = t_{\text{crossing}} \tag{11}$$

t_{crossing} records the time when the spectral tuning curve from one sub-band crosses the onset threshold θ_n of the n^{th} encoding neuron. In this way, both the temporal and amplitude information is encoded and made known to the SNN, which meets the third criterion mentioned above.

Secondly, coding efficiency, which refers to the average encoding spike rates (number of spikes per second), is also studied in Pan et al. (2019). The threshold code has the least average spike rates among all investigated neural codes. As the threshold code encodes only threshold-crossing events, it is supposed to be the most efficient coding method.

Thirdly, the threshold code promises to be more robust against noise, such as spike jitter. As it encodes the trajectory of the dynamics of the sub-band spectral power, the perturbation of precise spike timing will have less impact on the sequence of encoding neurons.

As such, the threshold code is a promising encoding scheme for temporal sequence recognition tasks (Pan et al., 2019). Further evaluation will be provided later in the experiments. While we note that each neural coding scheme has its own advantages, we focus on how the encoding scheme may help subsequent SNN learning algorithms in a cognitive task in this paper. As such, we adopt the threshold code for all experiments in this paper.

Biologically Plausible Auditory Encoding (BAE) With Masking Effects

We propose a BAE front-end with masking effects as illustrated in **Figure 7**.

Firstly the auditory stimuli are sensed and amplified by the microphone and some peripheral circuits, leading to a digital signal (a). This process corresponds to the pathway of the pinna, external auditory meatus, tympanic membrane, and auditory tube. Then the physically sensed stimuli are filtered by the cochlear filter bank (b), that emulates the cochlear function of frequency analysis. The outputs of the cochlear filter bank are parallel streams of time-domain sub-band (or so-called critical band) signals with psychoacoustic centre frequencies and bandwidths. For the purpose of further neural coding and cognitive tasks, the sub-band signals should be framed as the logarithm-scale energy as per Eq. 9. The output of (c), the raw spectrogram, is then converted into a precise spike pattern. The spectrogram is also being used to calculate the simultaneous and temporal masking thresholds, as in (d) and (e), under which the spikes will be omitted. Finally a sparse, perceptually related, and learnable temporal spike pattern for a learning SNN is generated as shown in (g).

Figure 8 gives an example of the intermediate results at different stages in **Figure 7** for a speech data waveform. **Figures 8A,B** show the raw waveform and the spectrogram of a speech utterance "three" spoken by a male speaker. The spectrogram is further encoded into a raw spike pattern by threshold neural coding. **Figure 8D** is the masking thresholds as formulated in section "Auditory Masking Effects," according to which the raw spike pattern **Figure 8C** is masked and results in a masked spike pattern **Figure 8E**. 50.48% of all spikes are discarded, given by the results in the later experiment section. **Figure 9** further demonstrates the comparison between auditory masked/unmasked spike patterns.

EXPERIMENT AND RESULTS

Spike-TIDIGITS and Spike-TIMIT Databases

The TIDIGITS (Leonard and Doddington, 1993) (LDC Catalog No. LDC93S10) is a speech corpus of spoken digits for speaker-independent speech recognition (Cooke et al., 2001; Tamazin et al., 2019). The speakers are from different genders (male and female), age ranges (adults and children), dialect districts (Boston, Richmond, Lubbock, etc.). As such, the corpus provides sufficiently speaker diversity and becomes one of the common benchmarking datasets. The TIDIGITS has a vocabulary of 11 spoken words of digits. The original database contains both isolated digits and digits sequences. In this work, we only use the isolated digits: each utterance contains one individual spoken digit. In this first attempt, we would like to build a spike-version

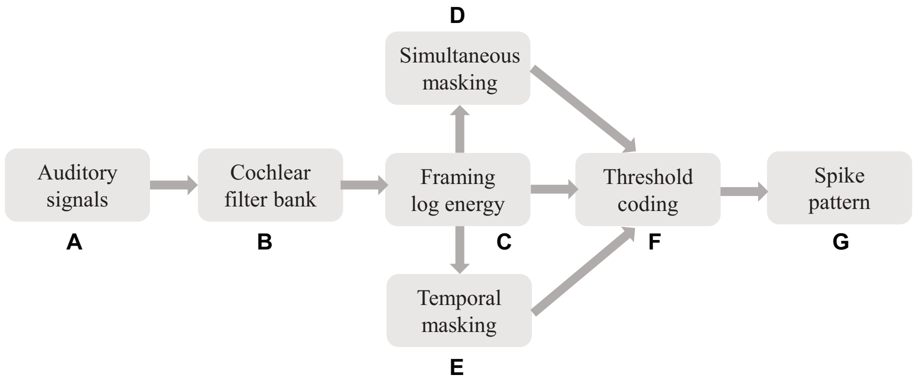


FIGURE 7 | The BAE scheme for temporal learning algorithms in auditory cognitive tasks. The raw auditory signals (a) are filtered by the CQT-based event-driven cochlear filter bank, resulting in a parallel stream of sub-band signals. For each sub-band, the signal is logarithmically framed, which corresponds to the processing in auditory hair cells. The framed spectral signals are then further masked in simultaneous and temporal masking.

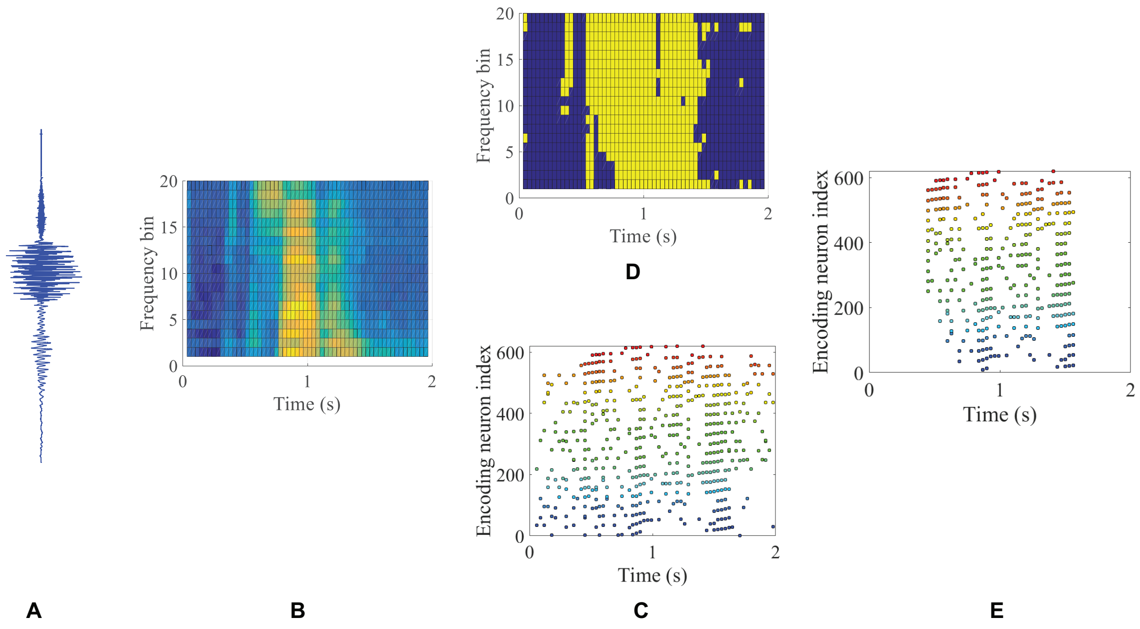


FIGURE 8 | An illustration of the intermediate results in a BAE process. Raw speech signal **(A)** of a speech utterance "three" is filtered and framed into a spectrogram **(B)**, corresponding to the process in **Figure 7** (a) and (c). By applying the neural threshold code, a precise spike pattern **(C)** is generated from the spectrogram. The masking map as described in Eq. 5 is illustrated in **(D)**, where yellow and dark blue color blocks represent the values 1 and 0, respectively. The masking **(D)** is applied to the spike pattern **(C)** and the auditory masked spike pattern is obtained in **(E)**.

speech dataset that contains sufficient diversity and can be immediately used to train an SNN classifier (Pan et al., 2018; Wu et al., 2018a). As each digit is repeated 224 and 226 times, the Spike-TIDIGITS has 224 × 11 = 2464 and 226 × 11 = 2486 isolated digit utterances for the training and testing set, respectively.

The BAE encoder proposed in section "Biologically Plausible Auditory Encoding With Masking Effects" and **Figure 7** is applied as the standard encoding scheme to generate this spike dataset. **Tables 1**, **2** describe the parameters in the encoding process of Spike-TIDIGITS.

Next, we encode one of the most popular speech dataset TIMIT (Garofolo, 1993) into a spike-version, Spike-TIMIT. TIMIT dataset consists of richer acoustic-phonetic content than TIDIGITS (Messaoud and Hamida, 2011). It consists of continuous speech utterances, that are useful for the evaluation of speech coding schemes (Besacier et al., 2000), speech enhancement El-Solh et al. (2007) or ASR systems (Mohamed et al., 2011; Graves et al., 2013). Similar to TIDIGITS, the speakers of TIMIT corpus are from eight different dialect regions in the United States, 438 males and 192 females. There are 4621 and 1679 speech sequences in the training and testing sets. This

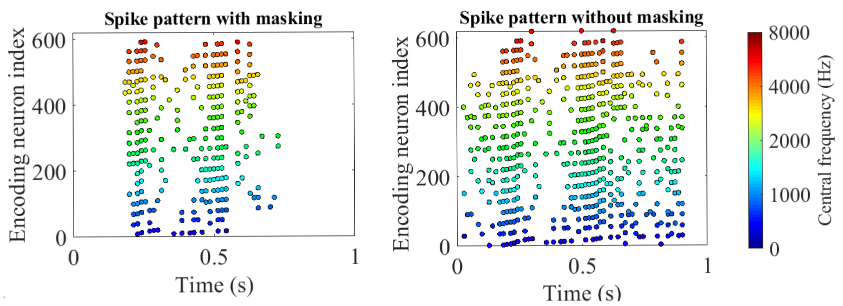


FIGURE 9 | Encoded spike patterns by threshold coding with/without masking. The two spike patterns are encoded from a speech utterance of "five" in the TIDIGITS dataset. The *x*-axis and *y*-axis represent the time and encoding neuron index. The positions of the colorful dots indicate the spike timings of the corresponding encoding neurons. The colors distinguish the centre frequencies of the cochlear filter bank. With auditory masking, the number of spikes reduces by nearly 50%, which are close to the 55% reducing rate of coding pulses as reported in Ambikairajah et al. (1997).

TABLE 1 | Parameters of neural threshold encoding for the speech and audio databases.

Parameter	Value
Window size	30 ms
Stride size	15 ms
Frequency range	[200 Hz, 8000 Hz]
Sampling rate	20 kHz

TABLE 2 | Cochlear filter parameters: we use a total of 20 cochlear filters in the BAE front-end.

Cochlear filter index	Centre frequency (Hz)	Bandwidth (Hz)
1	200.2	69.3
2	238.3	83.0
3	283.2	98.6
4	336.4	117.2
5	400.4	139.6
6	476.1	166.0
7	565.9	197.3
8	672.3	234.4
9	800.8	278.3
10	952.1	331.1
11	1131.3	394.5
12	1345.2	468.8
13	1600.6	557.6
14	1903.3	663.1
15	2263.7	788.1
16	2690.9	937.5
17	3200.2	1114.3
18	3805.7	1325.2
19	4525.9	1576.2
20	8000.5	6949.2

The center frequency and bandwidth of each filter are listed.

corpus has a vocabulary of 6224 words, which is larger than that of TIDIGITS.

Our proposed BAE scheme is next evaluated in the following sections, using both reconstruction and speech pattern recognition experiments.

Audio Reconstruction From Masked Patterns

According to Eq. 5, we adopt the binary auditory mask $\Phi_{K\times N}(\phi_{ij})$ which either fully encodes or ignores an acoustic event. It is suggested in auditory theory (Ambikairajah et al., 1997) that partial masking may exist in the frequency domain, especially in the presence of rich frequency tones. We would like to evaluate the masking effect in the BAE front-end both objectively and subjectively.

We begin by reconstructing the spike trains into speech signals, and then evaluate the speech quality using several objective speech quality measures: Perceptual Evaluation of Speech Quality (PESQ), Root Mean Square Error (RMSE), and Signal to Distortion Ratio (SDR). The PESQ, defined in Beerends et al. (2002) and Rix et al. (2002), is standardized as ITU-T recommendation P.862 for speech quality test methodology (Recommendation P.862, 2001). The core principle of PESQ is the use of the human auditory perception model (Rix et al., 2001) for speech quality assessment. For speech coding, especially the perceptual masking proposed in this paper, the PESQ measure could correctly distinguish between audible and inaudible distortions and thus assess the impact of perceptually masked coding noise. Besides, the PESQ is also used in the assessment of MPEG audio coding where auditory masking is involved. In this paper, the PESQ scores are further converted to MOS-LQO (Mean Opinion Score-Listening Quality Objective) scores ranging from 1 to 5, which are more intuitive for assessing speech quality. The mapping function is obtained from ITU-T Recommendation P.862.1 (ITU-T Recommendation, 2003). **Table 3** defines the MOS scales and their corresponding speech quality subjective descriptions (Recommendation P.800, 1996).

TABLE 3 | Perceptual evaluation of speech quality (MOS) scores and their corresponding perceptual speech quality subjective assessments.

PESQ (MOS) scores	5	4	3	2	1
Speech quality	Excellent	Good	Fair	Poor	Bad

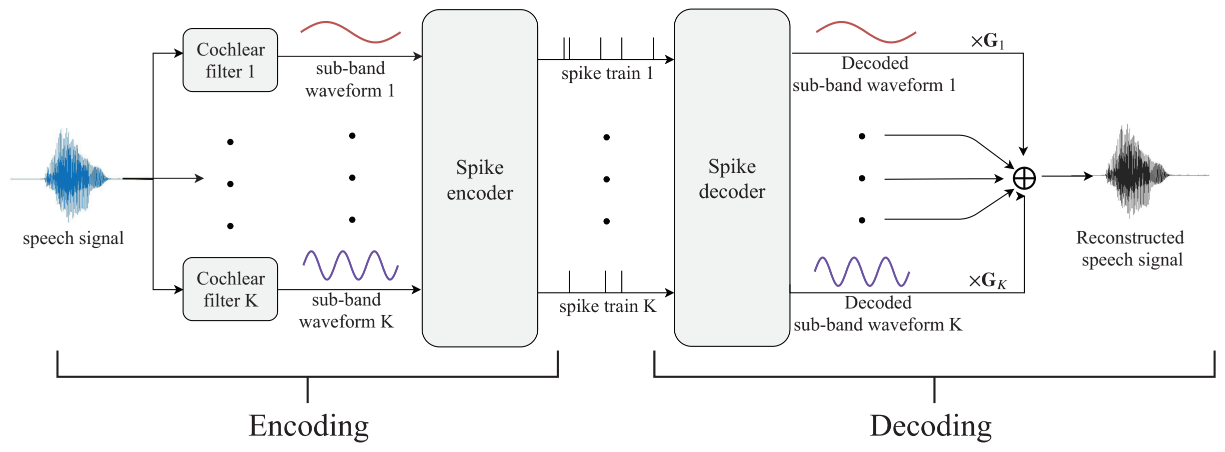


FIGURE 10 | The reconstruction from a spike pattern into a speech signal. Parallel streams of threshold- encoded spike trains that represent the dynamics of multiple frequency channels are first decoded into sub-band digital signals. The sub-band signals are further fed into a series of synthesis filters, which are built inversely from the corresponding analysis cochlear filters as in **Figure 6**. The synthesis filters compensate for the gains from the analysis filters for each frequency bin. Finally, the outputs from the synthesis filter banks sum up to generate the reconstructed speech signal.

TABLE 4 | The objective audio quality measures of the reconstructed audio signals for environmental sounds dataset RWCP.

Reconstructed signal	PESQ	RMSE	SDR (dB)	Reduced rate (%)
$\hat{s}_{\text{raw}}$	4.53	2.79×10^{-4}	38.96	0
$\hat{s}_{\text{mask}}$	4.15	5.67×10^{-4}	33.13	39.38
$\hat{s}_{\text{random}}$	2.81	1.85×10^{-2}	4.95	40.07

The reduced rates refer to the ratio of masked spikes.

TABLE 5 | The objective speech quality measures of the reconstructed speech signals for the spoken digits dataset TIDIGITS.

Reconstructed signal	PESQ	RMSE	SDR (dB)	Reduced rate (%)
$\hat{s}_{\text{raw}}$	4.54	4.78×10^{-4}	34.60	0
$\hat{s}_{\text{mask}}$	4.43	7.49×10^{-4}	29.94	50.48
$\hat{s}_{2\times\text{threshold}}$	3.80	8.50×10^{-3}	5.65	50.80
$\hat{s}_{\text{random}}$	2.92	1.05×10^{-2}	4.76	49.91

The reduced rates refer to the ratio of masked spikes.

TABLE 6 | The objective speech quality measures of the reconstructed speech signals for the continuous and large-vocabulary speech dataset TIMIT.

Reconstructed signal	PESQ	RMSE	SDR (dB)	Reduced rate (%)
$\hat{s}_{\text{raw}}$	4.54	1.23×10^{-4}	42.28	0
$\hat{s}_{\text{mask}}$	4.44	3.10×10^{-4}	34.02	29.33
$\hat{s}_{\text{random}}$	2.35	9.20×10^{-3}	4.83	30.8

The reduced rates refer to the ratio of masked spikes.

Besides PESQ, the RMSE (Eq. 12) and Expand SDR (Eq. 13) measures are also reported, where x_i and $\hat{x}_i$ denote the i^{th} time-domain sample of the original and reconstructed speech signals $\boldsymbol{x}_{1\times M}$ and $\hat{\boldsymbol{x}}_{1\times M}$, respectively.

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(x_i - \hat{x}_i)^2} \tag{12}$$

$$\text{SDR} = 10\log_{10}\left(\frac{\sum_{i=1}^{M}(x_i)^2}{\sum_{i=1}^{M}(x_i - \hat{x}_i)^2}\right) \tag{13}$$

For comparison, we compare three groups of reconstructed speech signals: (1) the reconstructed signal $\hat{\mathbf{s}}_{\text{mask}}$ from spike trains with auditory masking; (2) the reconstructed signal $\hat{\mathbf{s}}_{\text{raw}}$ from raw spike trains without auditory masking; and (3) the reconstructed signal $\hat{\mathbf{s}}_{\text{random}}$ from randomly masked spike trains.

Figure 10 depicts the flowchart of the reconstruction process. The left and right panels represent the spike encoding and decoding processes. The raw speech signals are first decomposed by a series of cochlear analysis filters, generating parallel streams of sub-band signals as in **Figure 7** (b). The 20 sub-band waveforms are encoded into spike times with masking strategies and then decoded back to sub-band speech signals. The reproduced sub-band waveforms 1 to K (20 in this work) are gain-weighted and summed to form the reconstructed speech signal for perceptual quality evaluation. Since the cochlear filters decompose the input signal by various weighting gains in different frequency bands, the weighting gains in the decoding part represent the inverse processing of the cochlear filters.

The audio quality of the three groups of reconstructed signals is compared, as reported in **Tables 4–6**. For a fair comparison, we first simulate a random masking effect by randomly dropping the same amount of spikes as that of the auditory masking. We further simulate a masking effect by doubling the firing thresholds that are used in **Figure 7** (f) purely according to energy level. The reconstructed signals from such elevated thresholds

TABLE 7 | Spiking neural network architectures for RWCP sound classification and TIDIGITS speech recognition tasks.

Dataset and task	Input layer	Output layer	Learning rule
RWCP sound classification	1 × 620 encoding neurons	1 × 10 LIF neurons	Tempotron
TIDIGITS speech recognition	1 × 620 encoding neurons	1 × 11 LIF neurons	MPD-AL

LIF refers to the Leaky-Integrate and Fire spiking neuron.

TABLE 8 | The RWCP classification accuracy for different neural encoding schemes and the average spike rates.

Neural encodings-classifier model	Classification accuracy (%)	# Spikes/ second
BAE-tempotron	99.5	245
Latency coding-tempotron	10.1	1, 598
Phase coding-tempotron	10.1	1, 598
Population coding-tempotron	99.0	4, 627
Threshold coding-tempotron	99.5	557
MFCC-HMM Dennis et al., 2013	90.0	–
LSF-SNN Dennis et al., 2013	98.5	–
LTF-SNN Xiao et al., 2016	97.5	–
DKP-SNN Yao et al., 2019	99.1	–

The average spike rates refer to the average number of spikes per second when the same audio samples are encoded.

are denoted as $\hat{\boldsymbol{s}}_{2 \times \text{threshold}}$. The raw spike patterns without any masking are used as a reference.

The perceptual quality scores of the $\hat{\boldsymbol{s}}_{mask}$ and $\hat{\boldsymbol{s}}_{raw}$ are rather close at a high level of approximately 4.5, which suggests a satisfying subjective quality between "Excellent" and "Good" according to **Table 3**. It is noted that the speech signals with random masking are perceived as "Poor" in quality. Besides the PESQ, the other two measures also lead to the same conclusion. The RMSE of $\hat{\boldsymbol{s}}_{raw}$ and $\hat{\boldsymbol{s}}_{mask}$ are approximately two orders of magnitude larger than that of the $\hat{\boldsymbol{s}}_{random}$; the SDRs also show a great gap.

Sound Classification by SNN for RWCP Dataset

Firstly, the BAE encoding is evaluated by a sound classification task on the RWCP sound scene database (Nakamura et al., 2000), which records the non-speech environmental sounds with rich and diverse frequency components. There are 10 sound categories in this database: bells, bottle, buzzer, cymbals, horn, kara, metal, phone, ring, and whistle. The sound recordings, each clip lasting for several seconds, are encoded into sparse spatio-temporal spike patterns and classified by a supervised learning SNN. We adopt a network structure that is shown in **Table 7** with Tempotron synaptic learning rule (Gütig and Sompolinsky, 2006).

Besides the BAE scheme, we also implement several other neural encoding schemes for the threshold coding (**Figure** 7) (f), such as latency coding (Gollisch and Meister, 2008), phase coding (Giraud and Poeppel, 2012), population coding (Dean et al., 2005), etc. The detailed implementation can be found at (Pan et al., 2019). The classification accuracy, as well as the average spike rates are summarized in **Table 8**.

The results in **Table 8** show that our BAE scheme achieves the best classification accuracy (99.5%) with a spiking rate of 245 spikes per second, across the other commonly used neural encoding schemes. The results suggest that the proposed BAE encoding scheme is both effective in pattern classification, and energy-efficient.

Speech Recognition by SNN for TIDIGITS Dataset

In this section, we evaluate the BAE scheme in a speech recognition task, which also aims to evaluate the coding fidelity of our proposed methodology. The spike patterns encoded from TIDIGITS speech dataset are fed into an SNN, and the outputs correspond to the labels of which spoken digits the patterns are encoded from. The spike learning rule is Membrane Potential Driven Aggregate-Label Learning (MPD-AL) (Zhang et al., 2019). The network structure is given in **Table 7**.

To evaluate the effectiveness and robustness of the BAE front-end, we compare the classification performances between spike patterns with and without auditory masking. Gaussian noise, measured by Signal-to-Noise Ratio (SNR) in dB, is added to the original speech waveforms before the encoding process. **Figure 11** shows the classification accuracy under noisy conditions and in the clean condition. Besides, we also compare our scheme with the other benchmarking results in **Table 9**.

The results show that the pattern classification accuracy for masked patterns is slightly higher than those of the unmasked patterns, under different test conditions. Above all, referring to **Table 5**, our proposed BAE scheme helps to reduce nearly half of the spikes, which is a dramatic improvement in coding efficiency.

Large Vocabulary Speech Recognition for TIMIT Dataset

In section "Biologically Plausible Auditory Encoding With Masking Effects," we present how the TIMIT dataset has been encoded into spike trains, which we henceforth refer to as Spike-TIMIT. We next train a recurrent neural network, the LSTM (Hochreiter and Schmidhuber, 1997) on both the original TIMIT and Spike-TIMIT datasets, with the CTC loss function (Graves et al., 2006). The LSTM architectures and the system performances on various TIMIT datasets are summarized in **Table 10**. The LSTM networks are adopted from Tensorpack[1]. We obtained a PER of 27% and 30%, respectively, for the TIMIT and Spike-TIMIT datasets. For comparison, we also report the accuracy for Spike-TIMIT without masking that shows 28% PER.

[1] Wu Y. et al. (2016). *Tensorpack*. Available at: https://github.com/tensorpack/.

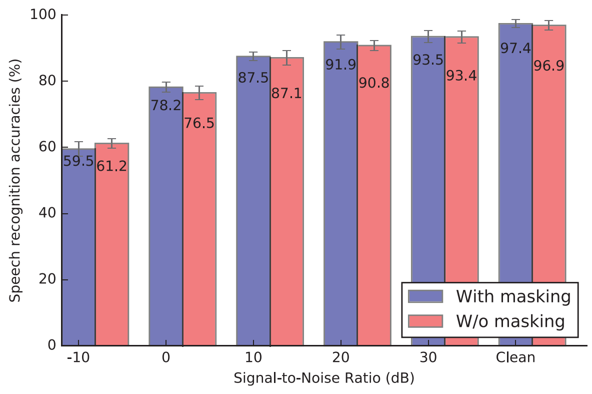


FIGURE 11 | The classification accuracy for the Spike-TIDIGITS dataset under different signal-to-noise ratios, with or without masking effects. The accuracy for the Spike-TIDIGITS with masking effects is slightly higher than that for the Spike-TIDIGITS without masking effects.

TABLE 9 | The TIDIGITS speech recognition accuracy for different neural encoding schemes.

Models of TIDIGITS speech recognition	Classification accuracy (%)
Biologically plausible auditory encoding (BAE)-MPDAL	97.4
Liquid state machine Zhang et al., 2015	92.3
SNN-SVM Tavanaei and Maida, 2017b	91.0
Spiking CNN-HMM Tavanaei and Maida, 2017a	96.0
AER silicon cochlea-SVM Abdollahi and Liu, 2011	95.6
Auditory spectrogram-SVM Abdollahi and Liu, 2011	78.7

We also notice some improvement in accuracy when dropout is introduced for Spike-TIMIT but not for TIMIT. Although the phone error rates are quite close across these datasets, we observe that the Spike-TIMIT derived from our proposed BAE scheme shows the highest spike efficiency (30% spike reduction in **Table 6**), which further improves the energy efficiency.

DISCUSSION

In this paper, we propose a BAE scheme, especially for speech signals. The encoding scheme is inspired by the modeling of the human auditory sensory system, which is composed of spectral analysis, neural spike coding, as well as the psychoacoustic perception model. We adopt three criteria for formulating the auditory encoding scheme.

For the spectral analysis part, a time-domain event-based cochlear filter bank is applied, with the perceptual scale of centre frequencies and bandwidths. The key feature of the spectral analysis is the parallel implementation of time-domain convolution. One of the most important properties of SNN is its asynchronous processing. The parallel implementation makes the neural encoding scheme a friendly frontend for any SNN processing. The neural encoding scheme, the threshold code in our case, helps to generate a sparse and representative spike patterns for efficient computing in the SNN classifier. The threshold code helps in two aspects: firstly it tracks the trajectory of the spectral power tuning curves, which represents the features in the acoustic dynamics; secondly, the threshold code, as a form of population neural code, is able to project the dynamics in the time domain onto the spatial domain, which facilitates the parallel processing of spiking neurons on cognitive tasks (Pan et al., 2019). Another key component of the BAE front-end is the implementation of auditory masking that benefits from findings in human psychoacoustic experiments. The integrated auditory encoding scheme fulfills the three proposed design criteria. We have evaluated our BAE scheme through signal reconstruction and speech recognition experiments giving very promising results. To share our study with the research community, the spike-version of TIDIGITS and TIMIT speech corpus, namely, Spike-TIDIGITS, and Spike-TIMIT, will be made available as benchmarking datasets.

Figure 12 illustrates some interesting findings in our proposed auditory masking strategy. The upper, middle and lower panels of **Figure 12** represent three speech utterances from the TIDIGITS dataset. The first and second column illustrates the encoded spike patterns with and without auditory masking effects. It is apparent that a large number of spikes are removed. The graphs in the third column demonstrate the membrane potential of the output neuron in the trained SNN classifier after being fed with both patterns during the testing phase. For example, the LIF neuron in (**Figure 12C**) responds to the speech utterance of "six." As such, the encoded pattern of spoken "six," as in (**Figures 12A,B**) will trigger the corresponding neuron to fire a spike in the testing phase. The sub-figure (c) demonstrates that though the sub-threshold membrane potentials of masking/unmasking patterns have different trajectories, the two membrane potential curves will exceed the firing threshold (which is 1 in this example) at close timing. Similar results are observed in **Figures 12F,I**. The spike patterns with or without auditory masking lead to similar neuronal responses, either in spiking activities (firing or not) or in membrane potential dynamics, as observed in **Figures 12C,F,I**. It is interesting to observe that auditory masking has little impact on the neuronal dynamics. As a psychoacoustic phenomenon, the auditory masking is always studied using listening tests. It remains unclear how the human auditory system responds to auditory masking. **Figure 12** provides an answer to the same question from an SNN perspective.

The parameters of auditory masking effects in this work, such as the exponential decaying parameter c in Eq. 2, or the cross-channel simultaneous masking thresholds in **Figure 2**, are all derived in the acoustic model of MPEG-1 Layer III standard (Fogg et al., 2007) and tuned according to the particular tasks. However, from a neuroscience point of view, our brain is adaptive to different environments. This suggests that the parameters could be optimized by machine learning methodology, for different tasks and datasets. Also, the threshold neural code, which encodes the dynamics of the spectrum using threshold-crossing events, relies heavily on the choice of thresholds. We use 15 uniformly distributed thresholds for simplicity. We note that the recording of threshold-crossing events is analogous to quantization in

TABLE 10 | Phone error rates (PER) for TIMIT and Spike-TIMIT datasets, using different LSTM structures.

Dataset	Network architecture	PER (%)
TIMIT	1 × 39 – 1 × 512 LSTM – 1 × 512 LSTM – 1 × 62	27
TIMIT	1 × 39 – 1 × 1024 LSTM – 1 × 1024 LSTM – 1 × 62	27
Spike-TIMIT	1 × 620 – Dropout (0.2) – 1 × 512 LSTM – 1 × 512 LSTM – 1 × 62	30
Spike-TIMIT	1 × 620 – Dropout (0.2) – 1 × 1024 LSTM – 1 × 1024 LSTM – 1 × 62	35
Spike-TIMIT-w/o mask	1 × 620 – Dropout (0.2) – 1 × 512 LSTM – 1 × 512 LSTM – 1 × 62	28
Spike-TIMIT-w/o mask	1 × 620 – Dropout (0.2) – 1 × 1024 LSTM – 1 × 1024 LSTM – 1 × 62	30
TIMIT[1]	1 × 39 – 1 × 512 LSTM – 1 × 512 LSTM – 1 × 62	28

[1]*Wu Y. et al. (2016). Tensorpack. Available at: https://github.com/tensorpack/.*

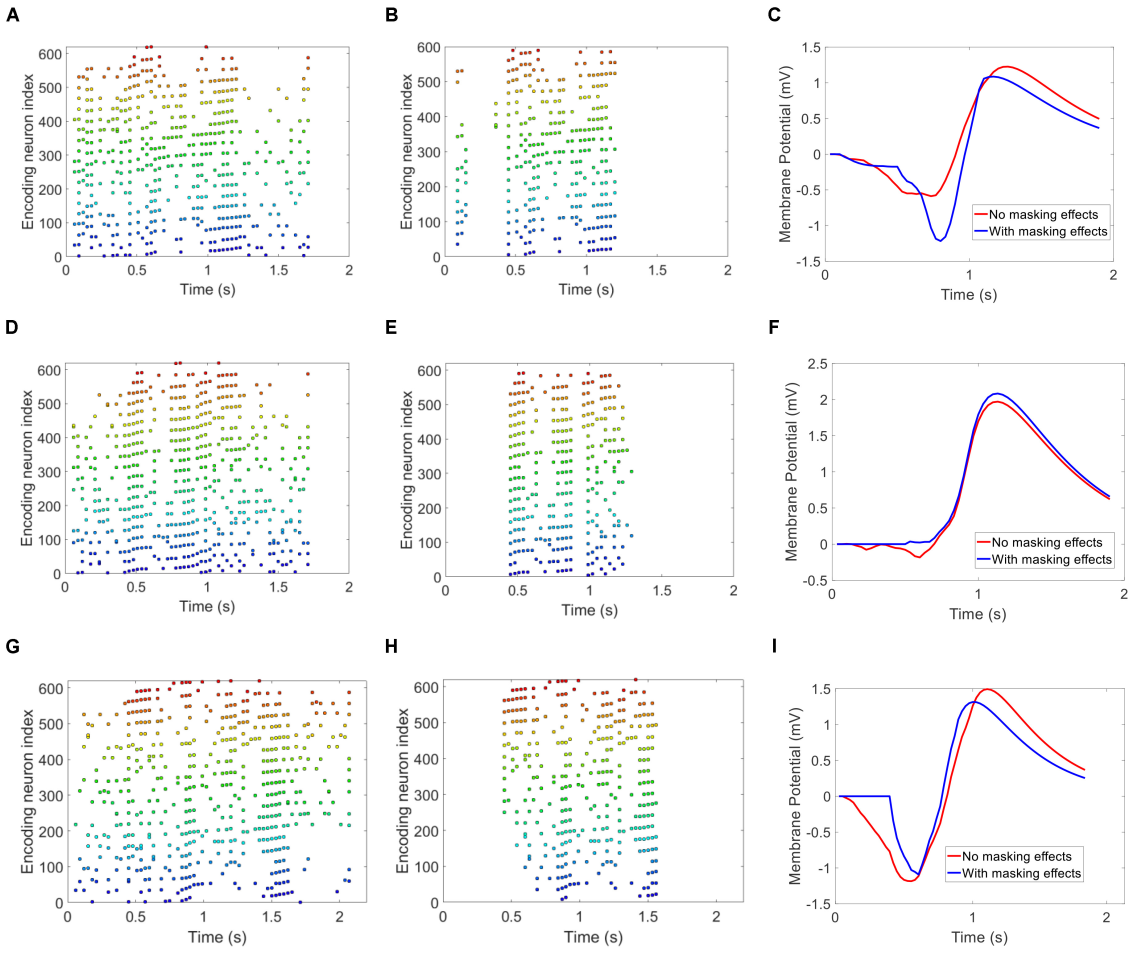


FIGURE 12 | Free membrane potential of trained Leaky-Integrate and Fire neurons, by feeding patterns with and without masking. The upper **(A–C)**, middle **(D–F)**, and lower **(G–I)** panels are for three different speech utterances "six," "seven," and "eight." The spike patterns with or without masking are apparently different, but the output neuron follows similar membrane potential trajectories.

digital coding, that the maximal coding efficiency (maximal information being conveyed constrained by the numbers of neurons or spikes) may be derived using an information-theoretic approach. The Efficient Coding Hypothesis (ECH) (Barlow, 1961; Srinivasan et al., 1982) that describes the link between neural encoding and information theory could provide us the theoretical framework to determine the optimal threshold distribution in the neural threshold code. It may also otherwise be learned using machine learning techniques.

CONCLUSION

Our proposed BAE scheme, motivated by the human auditory sensory system, could encode temporal audio data into spike patterns that are sparse, efficient, and friendly to SNN learning rules. It is both efficient and effective. We use the BAE scheme to encode popular speech datasets, namely, TIDIGITS and TIMIT into their spike versions: Spike-TIDIGITS and Spike-TIMIT. The two spike datasets are to be published as benchmarking datasets, in the hope of improving SNN-based classifiers.

DATA AVAILABILITY STATEMENT

The datasets Spike-TIDIGITS and Spike-TIMIT for this study can be found at https://github.com/pandarialTJU/Biologically-plausible-Auditory-Encoding/tree/master.

AUTHOR CONTRIBUTIONS

ZP performed the experiments and wrote the manuscript. All authors contributed to the experiments design, result interpretation, and writing.

FUNDING

This work was supported by in part by the Programmatic Grant No. A1687b0033 from the Singapore Government's Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain).

REFERENCES

Abdollahi, M. and Liu, S.-C. (2011). Speaker-independent isolated digit recognition using an aer silicon cochlea. *Proceeding of the 2011 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (Piscataway, NJ: IEEE), 269–272

Ambikairajah, E., Davis, A., and Wong, W. (1997). Auditory masking and mpeg-1 audio compression. *Electron. Commun. Eng. J.* 9, 165–175 doi: 10.1049/ecej:19970403

Ambikairajah, E., Epps, J., and Lin, L. (2001). Wideband speech and audio coding using gammatone filter banks. *Proceeding of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing.* (Piscataway, NJ: IEEE) 2, 773–776

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., et al. (2017). A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* (Piscataway, NJ: IEEE) 7243–7252

Anumula, J., Neil, D., Delbruck, T., and Liu, S.-C. (2018). Feature representations for neuromorphic audio spike streams. *Front. Neurosci.* 12:23 doi: 10.3389/fnins.2018.00023

Appelle, S. (1972). Perception and discrimination as a function of stimulus orientation: the" oblique effect" in man and animals. *Psychol. Bull.* 78, 266–278 doi: 10.1037/h0033117

Arnal, L. H. and Giraud, A.-L. (2012). Cortical oscillations and sensory predictions. *Trends Cogn. Sci.* 16, 390–398 doi: 10.1016/j.tics.2012.05.003

Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. *Sens. Commun.* 1, 217–234

Beerends, J. G., Hekstra, A. P., Rix, A. W., and Hollier, M. P. (2002). Perceptual evaluation of speech quality (pesq) the new itu standard for end-to-end speech quality assessment part ii: psychoacoustic model. *J. Audio Eng. Soc.* 50, 765–778

Besacier, L., Grassi, S., Dufaux, A., Ansorge, M., and Pellandini, F. (2000). Gsm speech coding and speaker recognition. In *Proceeding of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Piscataway, NJ: IEEE) 2, II1085-II1088

Bohte, S. M., Kok, J. N., and La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 17–37 doi: 10.1016/s0925-2312(01)00658-0

Brown, J. C. (1991). Calculation of a constant q spectral transform. *J. Acoust. Soc. Am.* 89, 425–434 doi: 10.1121/1.400476

Brown, J. C. and Puckette, M. S. (1992). An efficient algorithm for the calculation of a constant q transform. *J. Acoust. Soc. Am.* 92, 2698–2701 doi: 10.1121/1.404385

Cooke, M., Green, P., Josifovski, L., and Vizinho, A. (2001). Robust automatic speech recognition with missing and unreliable acoustic data. *Speech Commun.* 34, 267–285 doi: 10.1016/s0167-6393(00)00034-0

Darabkh, K. A., Haddad, L., Sweidan, S. Z., Hawa, M., Saifan, R., and Alnabelsi, S. H. (2018). An efficient speech recognition system for arm-disabled students based on isolated words. *Comput. Appl. Eng. Edu.* 26, 285–301 doi: 10.1002/cae.21884

Dean, I., Harper, N. S., and McAlpine, D. (2005). Neural population coding of sound level adapts to stimulus statistics. *Nature Neurosci.* 8, 1684–1689 doi: 10.1038/nn1541

Dennis, J., Yu, Q., Tang, H., Tran, H. D., and Li, H. (2013). Temporal coding of local spectrogram features for robust sound recognition. In *Proceeding of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE)*, (Piscataway, NJ: IEEE) 803–807

Ehret, G. (1997). The auditory cortex. *J. Comp. Physiol. A* 181, 547–557

El-Solh, A., Cuhadar, A., and Goubran, R. A. (2007). Evaluation of speech enhancement techniques for speaker identification in noisy environments. In *Proceeding of the 9th IEEE International Symposium on Multimedia Workshops (ISMW 2007) (IEEE)*, (Piscataway, NJ: IEEE) 235–239

Fogg, C., LeGall, D. J., Mitchell, J. L., and Pennebaker, W. B. (2007). *MPEG Video Compression Standard* (Berlin: Springer Science & Business Media)

Garofolo, J. S. (1993). *Timit Acoustic Phonetic Continuous Speech Corpus.* Philadelphia, PA: Linguistic Data Consortium.

Ghosh-Dastidar, S. and Adeli, H. (2009). Spiking neural networks. *Intl. J. Neural Sys.* 19, 295–308

Giraud, A.-L. and Poeppel, D. (2012). Cortical oscillations and speech processing: emerging computational principles and operations. *Nat. Neurosci.* 15, 511–5187 doi: 10.1038/nn.3063

Gollisch, T. and Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. *science* 319, 1108–1111 doi: 10.1126/science.1149639

Graves, A., Fernàndez, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning* (New York,NY: ACM), 369–376

Graves, A., Mohamed, A.-R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceeding of the 2013 IEEE international conference on acoustics, speech and signal processing* (Piscataway, NJ: IEEE), 6645–6649

Gütig, R. (2016). Spiking neurons can discover predictive features by aggregate-label learning. *Science* 351:aab4113 doi: 10.1126/science.aab4113

Gütig, R. and Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing–based decisions. *Nat. Neurosci.* 9, 420–428 doi: 10.1038/nn1643

Gütig, R. and Sompolinsky, H. (2009). Time-warp–invariant neuronal processing. *PLoS Biol.* 7:e1000141 doi: 10.1371/journal.pbio.1000141

Harris, D. M. and Dallos, P. (1979). Forward masking of auditory nerve fiber responses. *J. Neurophysiol.* 42, 1083–1107 doi: 10.1152/jn.1979.42.4.1083

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation 9, 1735–1780

Hohmann, V. (2002). Frequency analysis and synthesis using a gammatone filterbank. *Acta Acust. United Acust.* 88, 433–442

Hopfield, J. (2004). Encoding for computation: recognizing brief dynamical patterns by exploiting effects of weak rhythms on action-potential timing. *Proc. Natl. Acad Sci.* 101, 6255–6260 doi: 10.1073/pnas.0401125101

ITU-T Recommendation, (2003). *862.1: Mapping Function for Transforming p. 862 Raw Result Scores to Mos-Lqo*. Geneva: International Telecommunication Union.

Iyer, L. R., Chua, Y., and Li, H. (2018). Is neuromorphic mnist neuromorphic? analyzing the discriminative power of neuromorphic datasets in the time domain. *arXiv* [Preprint] 1807.01013

Lagerstrom, K. (2001). *Design and Implementation of An Mpeg-1 Layer iii Audio Decoder*. Gothenburg: Chalmers University of Technology.

Leonard, R. G. and Doddington, G. (1993). *Tidigits ldc93s10*. Philadelphia, PA: Linguistic Data Consortium.

Liu, S.-C. and Delbruck, T. (2010). Neuromorphic sensory systems. *Curr. Opin. Neurobiol.* 20, 288–295 doi: 10.1016/j.conb.2010.03.007

Loiselle, S., Rouat, J., Pressnitzer, D., and Thorpe, S. (2005). Exploration of rank order coding with spiking neural networks for speech recognition. In *Proceeding of the Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on (IEEE)*, (Piscataway, NJ: IEEE) 4, 2076–2080

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Netw.* 10, 1659–1671 doi: 10.1016/s0893-6080(97)00011-7

Mermelstein, P. (1976). Distance measures for speech recognition, psychological and instrumental. *Pattern Recognit. Artif. Intell.* 116, 374–388

Messaoud, Z. B. and Hamida, A. B. (2011). Combining formant frequency based on variable order lpc coding with acoustic features for timit phone recognition. *Int. J. Speech Technol.* 14:393 doi: 10.1007/s10772-011-9119-z

Mohamed, A.-R., Dahl, G. E., and Hinton, G. (2011). Acoustic modeling using deep belief networks. *Proceeding of the IEEE transactions on audio, speech, and language processing* (Piscataway, NJ: IEEE) 20, 14–22 doi: 10.1109/tasl.2011.2109382

Nadasdy, Z. (2009). Information encoding and reconstruction from the phase of action potentials. *Front. Sys. Neurosci.* 3:6 doi: 10.3389/neuro.06.006.2009

Nakamura, S., Hiyane, K., Asano, F., Nishiura, T., and Yamada, T. (2000). Acoustical sound database in real environments for sound scene understanding and hands-free speech recognition. In *Proceedings of the Second International Conference on Language Resources and Evaluation* France: LREC

Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* 9:437 doi: 10.3389/fnins.2015.00437

Pan, Z., Li, H., Wu, J., and Chua, Y. (2018). An event-based cochlear filter temporal encoding scheme for speech signals. In *Proceeding of the 2018 International Joint Conference on Neural Networks (IJCNN)* (Piscataway, NJ: IEEE), 1–8

Pan, Z., Wu, J., Chua, Y., Zhang, M., and Li, H. (2019). Neural population coding for effective temporal classification. In *Proceeding of the 2019 International Joint Conference on Neural Networks (IJCNN)* (Piscataway, NJ: IEEE), 1–8

Patterson, R., Nimmo-Smith, I., Holdsworth, J., and Rice, P. (1987). An efficient auditory filterbank based on the gammatone function. In *Proceeding of the a meeting of the IOC Speech Group on Auditory Modelling at RSRE*. (Malvern: RSRE).

Ponulak, F. and Kasiński, A. (2010). Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural Comput.* 22, 467–510 doi: 10.1162/neco.2009.11-08-901

Recommendation P.800 (1996). *P. 800: Methods for subjective determination of transmission quality*. Geneva: International Telecommunication Union 22.

Recommendation P.862 (2001). Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. Geneva: International Telecommunication Union.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (Piscataway, NJ: IEEE). 779–788

Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P. (2001). Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *Proceeding of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)* (Piscataway, NJ: IEEE) 2, 749–752

Rix, A. W., Hollier, M. P., Hekstra, A. P., and Beerends, J. G. (2002). Perceptual evaluation of speech quality (pesq) the new itu standard for end-to-end speech quality assessment part i–time-delay compensation. *J. Audio Eng. Soc.* 50, 755–764

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11:682 doi: 10.3389/fnins.2017.00682

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* 115, 211–252. doi: 10.1007/s11263-015-0816-y

Shinn-Cunningham, B. G. (2008). Object-based auditory and visual attention. *Trends Cogn. Sci.* 12, 182–186 doi: 10.1016/j.tics.2008.02.003

Shrestha, S. B. and Orchard, G. (2018). Slayer: spike layer error reassignment in time. In *Proceeding of the Advances in Neural Information Processing Systems*, Montreal, 1412–1421

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv* [Preprint] *1409.1556*

Smith, J. O. and Abel, J. S. (1999). Bark and erb bilinear transforms. *IEEE Trans. Speech Audio Process.* 7, 697–708 doi: 10.1109/89.799695

Srinivasan, M. V., Laughlin, S. B., and Dubs, A. (1982). Predictive coding: a fresh view of inhibition in the retina. *Proc. R. Soc. Lond. Series B. Biol. Sci.* 216, 427–459 doi: 10.1098/rspb.1982.0085

Taherkhani, A., Belatreche, A., Li, Y., and Maguire, L. P. (2015). Dl-resume: A delay learning-based remote supervised method for spiking neurons. *IEEE Trans. Neural Netw. and Learn. Syst.* 26, 3137–3149 doi: 10.1109/TNNLS.2015.2404938

Tamazin, M., Gouda, A., and Khedr, M. (2019). Enhanced automatic speech recognition system based on enhancing power-normalized cepstral coefficients. *Appl. Sci.* 9, 2166 doi: 10.3390/app9102166

Tavanaei, A. and Maida, A. (2017a). Bio-inspired multi-layer spiking neural network extracts discriminative features from speech signals. In *International Conference on Neural Information Processing* (Berlin: Springer), 899–908 doi: 10.1007/978-3-319-70136-3_95

Tavanaei, A. and Maida, A. S. (2017b). A spiking network that learns to extract spike signatures from speech signals. *Neurocomputing* 240, 191–199 doi: 10.1016/j.neucom.2017.01.088

Tobias, J. (2012). *Foundations of Modern Auditory Theory*, Amsterdam: Elsevier.

Veit, A., Matera, T., Neumann, L., Matas, J., and Belongie, S. (2016). Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv* [Preprint] 1601.07140

Wu, J., Chua, Y., and Li, H. (2018a). A biologically plausible speech recognition framework based on spiking neural networks. In *Proceeding of the 2018 International Joint Conference on Neural Networks (IJCNN)*, (Piscataway, NJ: IEEE), 1–8

Wu, J., Chua, Y., Zhang, M., Li, H., and Tan, K. C. (2018b). A spiking neural network framework for robust sound classification. *Front. Neurosci.* 12:836. doi: 10.3389/fnins.2018.00836

Xiao, R., Yan, R., Tang, H., and Tan, K. C. (2016). A spiking neural network model for sound recognition. In *Proceeding of the International Conference on Cognitive Systems and Signal Processing* (Berlin: Springer), 584–594 doi: 10.1007/978-981-10-5230-9_57

Yang, M., Chien, C.-H., Delbruck, T., and Liu, S.-C. (2016). A 0.5 v 55μw 64×2 channel binaural silicon cochlea for event-driven stereo-audio sensing. *IEEE J. Solid State Circuits* 51, 2554–2569 doi: 10.1109/jssc.2016.2604285

Yao, Y., Yu, Q., Wang, L., and Dang, J. (2019). A spiking neural network with distributed keypoint encoding for robust sound recognition. In *Proceeding of the 2019 International Joint Conference on Neural Networks (IJCNN)* (Piscataway, NJ: IEEE), 1–8

Zhang, M., Qu, H., Belatreche, A., Chen, Y., and Yi, Z. (2018). A highly effective and robust membrane potential-driven supervised learning method for spiking neurons. *IEEE Transactions on Neural Networks and Learning Systems* (Piscataway, NJ: IEEE) 1–15

Zhang, M., Qu, H., Belatreche, A., and Xie, X. (2017). Empd: An efficient membrane potential driven supervised learning algorithm for spiking neurons. *IEEE Trans. Cogn. Dev. Sys.* 10, 151–162 doi: 10.1109/tcds.2017.2651943

Zhang, M., Wu, J., Chua, Y., Luo, X., Pan, Z., Liu, D., et al. (2019). Mpd-al: an efficient membrane potential driven aggregate-label learning algorithm for spiking neurons. *Proc. AAAI Conf. Artif. Intell.* 33, 1327–1334 doi: 10.1609/aaai.v33i01.33011327

Zhang, Y., Li, P., Jin, Y., and Choe, Y. (2015). A digital liquid state machine with biologically inspired learning and its application to speech recognition. *IEEE Trans. Neural Netw. Learn. Sys.* 26, 2635–2649 doi: 10.1109/TNNLS.2015.2388544

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. (2016). Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv*[Preprint] 1606.06160

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.

frontiers in Neuroscience

ORIGINAL RESEARCH
published: 28 January 2020
doi: 10.3389/fnins.2020.00007

Controlled Forgetting: Targeted Stimulation and Dopaminergic Plasticity Modulation for Unsupervised Lifelong Learning in Spiking Neural Networks

Jason M. Allred and Kaushik Roy*

Nanoelectronics Research Laboratory, Electrical and Computer Engineering Department, Purdue University, West Lafayette, IN, United States

OPEN ACCESS

Edited by:
Michael Pfeiffer,
Bosch Center for Artificial Intelligence, Germany

Reviewed by:
David Kappel,
Dresden University of Technology, Germany
Saeed Reza Kheradpisheh,
Shahid Beheshti University, Iran

****Correspondence:***
Jason M. Allred
allredj@purdue.edu

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *20 August 2019*
Accepted: *07 January 2020*
Published: *28 January 2020*

Citation:
Allred JM and Roy K (2020) Controlled Forgetting: Targeted Stimulation and Dopaminergic Plasticity Modulation for Unsupervised Lifelong Learning in Spiking Neural Networks. Front. Neurosci. 14:7. doi: 10.3389/fnins.2020.00007

Stochastic gradient descent requires that training samples be drawn from a uniformly random distribution of the data. For a deployed system that must learn online from an uncontrolled and unknown environment, the ordering of input samples often fails to meet this criterion, making lifelong learning a difficult challenge. We exploit the locality of the unsupervised Spike Timing Dependent Plasticity (STDP) learning rule to target local representations in a Spiking Neural Network (SNN) to adapt to novel information while protecting essential information in the remainder of the SNN from catastrophic forgetting. In our Controlled Forgetting Networks (CFNs), novel information triggers stimulated firing and heterogeneously modulated plasticity, inspired by biological dopamine signals, to cause rapid and isolated adaptation in the synapses of neurons associated with outlier information. This targeting controls the forgetting process in a way that reduces the degradation of accuracy for older tasks while learning new tasks. Our experimental results on the MNIST dataset validate the capability of CFNs to learn successfully over time from an unknown, changing environment, achieving 95.24% accuracy, which we believe is the best unsupervised accuracy ever achieved by a fixed-size, single-layer SNN on a completely disjoint MNIST dataset.

Keywords: lifelong learning, continual learning, catastrophic forgetting, controlled forgetting, dopaminergic learning, Spiking Neural Networks, Spike Timing Dependent Plasticity, stability-plasticity dilemma

1. INTRODUCTION

Artificial neural networks have enabled computing systems to successfully perform tasks previously out of reach for traditional computing, such as image and audio classification. These networks, however, are typically trained offline and do not update during deployed inference. One of the current obstacles preventing fully autonomous, unsupervised learning in dynamic environments while maintaining efficiency is the *stability-plasticity dilemma*, or the challenge of ensuring that the system can continue to quickly and successfully learn from and adapt to its current environment while simultaneously retaining and applying essential knowledge from previous environments (Grossberg, 1987).

There have been a handful of terms used in literature to describe the process of learning from data that is temporally distributed inhomogeneously, such as the terms incremental learning, sequential learning, continual learning, and lifelong learning. In this work, we will use the term "lifelong learning." *Lifelong learning* is the process of successfully learning from new data while retaining useful knowledge from previously encountered data that is statistically different, often with the goal of sequentially learning differing tasks while retaining the capability to perform previously learned tasks without requiring retraining on data for older tasks. When traditional artificial neural networks are presented with changing data distributions, more rigid parameters interfere with adaption, while more flexibility causes the system to fail to retain important older information, a problem called *catastrophic interference* or *catastrophic forgetting*. Biological neuronal systems dont seem to suffer from this dilemma. We take inspiration from the brain to help overcome this obstacle.

To avoid catastrophic forgetting, important information from older data must be protected while new information is learned from novel data. Non-local learning rules may not provide such isolation. Localized learning, on the other hand, may provide the desired segmentation while also being able to perform unsupervised learning, which is critical for lifelong learning in unknown environments. Spike Timing Dependent Plasticity (STDP) is a localized biological Hebbian learning process where a synaptic weight's adjustment is a function of the timing of the *spikes*, or firing events, of its locally connected pre- and post-synaptic neurons. Spiking Neural Networks (SNNs), which have been explored for their potential energy advantages due to sparse computing (Han et al., 2018), have been shown to perform successful unsupervised clustering tasks with STDP (Diehl and Cook, 2015).

However, even though STDP learning is localized, it is still susceptible to catastrophic forgetting because the algorithms that employ STDP are traditionally designed for randomized input ordering. Certain features, such as homeostasis, attempt to distribute the effect of input groupings globally in order to benefit from the full network. Without a temporally uniform distribution of classes, traditional STDP algorithms still lose important older information, which is either replaced by or corrupted with information from newer samples (Allred and Roy, 2016).

We present a new learning paradigm, inspired by the dopamine signals in mammalian brains that non-uniformly, or heterogeneously modulate synaptic plasticity. We create Controlled Forgetting Networks (CFNs) that address the stability-plasticity dilemma with rapid/local learning from new information, rather than the traditional gradual/global approach to learning. Our approach allows fixed-size CFNs to successfully perform unsupervised learning of sequentially presented tasks without catastrophically forgetting older tasks.

Many recent papers have tackled the challenge of lifelong learning without catastrophic forgetting, but they are not designed to target the goal of this paper, which is autonomous learning on a deployed neuromorphic system. This goal requires real-time unsupervised learning, energy efficiency, and fixed network resources. Wysoski et al. (2006), Srivastava et al. (2013), Wang et al. (2014), Wang et al. (2015), Rusu et al. (2016), Fernando et al. (2017), Kirkpatrick et al. (2017), Lee et al. (2017), Aljundi et al. (2018), Li and Hoiem (2018), Bashivan et al. (2019) and Du et al. (2019) all employ supervised or reinforcement learning methods, in some way provide the network with the knowledge of when a task change occurs, or provide access to previous samples for retraining. For example, the work by Aljundi et al. (2018) requires that the system be allowed a parameter-"importance update" period on the older task(s) before proceeding to a new task. Similarly, Panda et al. (2018) requires that samples from earlier distributions be presented in disproportionately larger quantities than later distributions to avoid catastrophic forgetting, which would require knowledge of a task change. Additionally, Srivastava et al. (2013), Rusu et al. (2016), Fernando et al. (2017), Kirkpatrick et al. (2017), Lee et al. (2017), Li and Hoiem (2018) and Rios and Itti (2018) are also not applicable to localized learning rules that may be employed on spiking networks. And Wysoski et al. (2006), Dhoble et al. (2012), and Wang et al. (2017) are morphological systems that do not work with static-sized networks, which would exclude them from direct mapping onto physical hardware implementations.

2. MATERIALS AND METHODS

2.1. The Challenge of Lifelong Learning

Backpropagation has proven a successful learning algorithm for deep neural networks. The accuracy of this approach depends on proper stochastic gradient descent or SGD, also known as incremental gradient descent, in which many small, global adjustments to network weights are performed while iterating over samples from a training dataset. These samples, however, must be drawn from a random distribution of the dataset—hence the name "stochastic" gradient descent—intermixing the classes so that each class can affect the direction of descent for correct error minimization throughout the entire training process.

The need to draw training samples from a random distribution is an obstacle for on-line learning, especially when the system encounters novel data. Backpropagation in an on-line system for real-time learning proves difficult when the input from the environment is uncontrolled and unknown. With traditional SGD, the system typically has three choices to attempt learning from novel data: (1) train normally on inputs in the order seen; (2) periodically go offline and retrain from an updated dataset; (3) maintain an online storage of previous samples to intermix with the new samples, providing a simulated random sampling. The latter two choices are costly and inhibit real-time learning, while the first catastrophically violates SGD.

2.1.1. Catastrophic Forgetting Due to Global Interference

If a uniformly randomized order is not provided, e.g., samples are grouped by class and classes are presented sequentially to the network, then the gradient descent followed by latter samples will likely disagree with the direction from previous samples. This conflict causes the network to fail to reach an error minimum that respects older tasks, as at each period of time in the training process the network essentially attempts to globally optimize for only the current tasks, agnostic as to whether or not that particular direction increases the error for older tasks. Latter samples erase or corrupt the information learned from previous samples, causing catastrophic forgetting.

One of the largest underlying causes of catastrophic forgetting in backpropagation algorithms is the reliance on a global error. Calculating weight updates from the current sample's global error means that the current sample may globally affect network weights. Biological neuronal learning, on the other hand, appears to be significantly localized, with synaptic weight updates being a function of local activity, causing different regions to be responsible for different tasks. While distributed representations promote generalization in neural networks, rapid learning of novel information may not require significant modifications to low-level distributed representations in a sufficiently trained network. It has been shown that the IT cortex contains a large-scale spatial organization, or "shape map," that remains significantly stable over time (Op de Beeck et al., 2007), even while learning novel information. Lee and DiCarlo (2019) have shown that the stable earlier levels of the visual cortex are capable of representing the generic structure and composition of never-before-seen inputs with an already-learned understanding of the physical world that remains constant through the remainder of life–for example, an understanding of lines, edges, curves, and colors at the lowest levels and an understanding of rotations, shading, and physical properties at subsequent levels. Thus, it is likely that lifelong learning need only occur in the last one or two layers of a neural network, where local learning may sufficiently classify from a read-out of the higher-dimensional generalizations that have been learned previously.

2.1.2. Catastrophic Forgetting in Localized Learning Due to Homeostasis

Many leading STDP-trained SNNs employ adaptive thresholding, in which a neuron's firing threshold increases each time it fires and otherwise decays, preventing specific neurons from dominating the receptive field. Adaptive thresholding helps achieve homeostasis by distributing the firing activity between neurons. However, adaptive thresholding assumes a temporally random distribution of input samples and often causes catastrophic interference when the environment changes (Allred and Roy, 2016). For lifelong learning, adaptive thresholding must be modified to account for long-term variations in spiking activity that would occur when processing temporally variant input distributions.

2.1.3. The Need for Forgetting

For successful lifelong learning, there must be network resources available to learn new information. In a deployed system with finite resources, some forgetting of older knowledge is required to make room for information from new data. As mentioned earlier, there are morphological systems that logically grow the network to accommodate new information, even employing pruning techniques when necessary if the network grows too large. However, for our goal of deployed learning on neuromorphic hardware, inserting and removing physical components of the network is not an option, and existing network components must be re-purposed to learn a new task when network capacity is reached, requiring some forgetting.

Additionally, in some cases, forgetting may actually be beneficial. Forgetting outlier data can improve generalizations, and forgetting stale data can allow the system to adapt to a changing environment if new information directly contradicts older information. Because some forgetting must occur, we seek to control the forgetting process to protect the most vital information, minimizing accuracy loss.

2.2. Controlled Forgetting With Dopaminergic Learning

The stability-plasticity dilemma can be addressed by allowing for dynamic, heterogeneously modulated plasticity. Consider the example of unsupervised clustering where neurons are trained to center on input clusters (see **Figure 1**). Temporarily making the synaptic weights of some neurons more plastic while keeping the weights of other neurons more rigid can allow for isolated adaptation by the plastic parameters while protecting the information associated with the rigid parameters. The challenge then becomes how to dynamically control the plasticity and for which parameters.

STDP embeds local, generalized representations of correlated inputs within the synaptic weights of individual neurons. Lateral inhibition between neurons, similar to the architecture in Diehl and Cook (2015), creates competition that prevents multiple

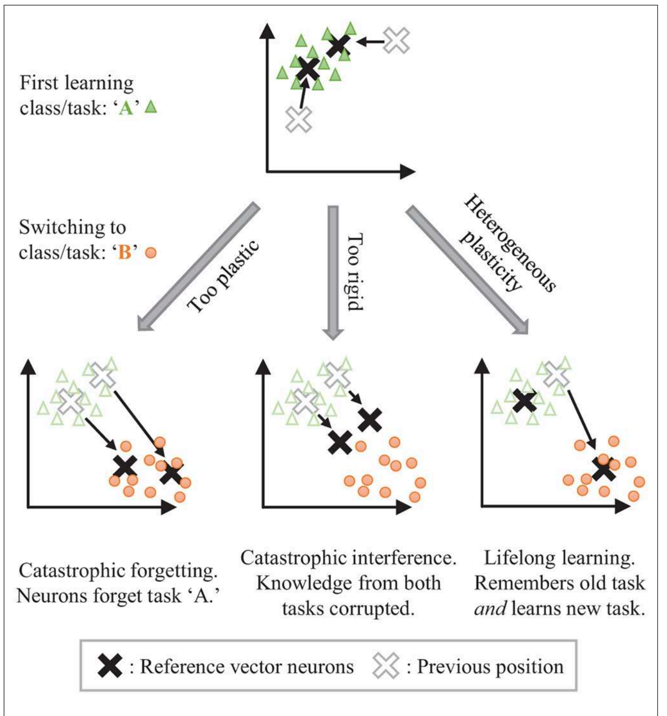

First learning class/task: 'A' ▲
Switching to class/task: 'B' ●
Too plastic
Too rigid
Heterogeneous plasticity
Catastrophic forgetting. Neurons forget task 'A.'
Catastrophic interference. Knowledge from both tasks corrupted.
Lifelong learning. Remembers old task *and* learns new task.
✖ : Reference vector neurons
✖ : Previous position

FIGURE 1 | The stability-plasticity dilemma in unsupervised clustering. Lifelong learning is achieved with a strategic heterogeneous modulation of synaptic plasticity.

neurons from learning the same information. We seek to control the forgetting process by harnessing the segmentation of localized and distinct representations that are created by STDP with competition. Interference from novel information may be isolated by stimulating specific network elements to adapt to that information, protecting the remainder of the network from change. The forgetting caused by this interference may be minimized and controlled by targeting network elements associated with less useful information. We draw on inspiration from biology to heterogeneously modulate STDP learning to perform such isolated adaptation, creating Controlled Forgetting Networks (CFNs).

2.2.1. Biologically Inspired Dopaminergic Plasticity Modulation

Dopamine acts as a neuromodulator which gates synaptic plasticity. Dopamine signals are most commonly thought of as reward signals. In addition, though, dopamine releases are also associated with encountering novel data, which allows the brain to quickly adapt to new information (Frémaux and Gerstner, 2016). We adopt this concept of novelty-induced plasticity modulation for our goal of local, rapid adaptation. We mimic a novelty-induced dopamine release by including a *dopaminergic neuron* at a given layer of a CFN (see **Figure 2**). We discuss how to identify novel information in an STDP-trained SNN, how the dopaminergic neuron is designed to fire under those conditions, and how the dopaminergic neuron modulates plasticity.

In STDP-trained SNNs, the weight vectors stabilize on the radial center of seen input clusters and are more likely to fire for inputs to which they are *angularly closer*–meaning inputs where the angle between the input vector and the weight vector are smaller for a given vector magnitude (see sections 2.3.4.1, 2.4.2.4). In other words, a sample from an unseen distribution will be less likely to induce firing than a sample from a learned distribution. Thus, when an input sample results in little-to-no firing activity at a given layer of neurons, we may assume that it contains information novel to that layer. (Data in Figure S3 in the **Supplementary Material** validate this assumption, showing that a dopaminergic neuron designed to fire under these conditions is indeed triggered more frequently whenever the system switches to an unseen class and otherwise sees a reduction in triggered dopaminergic activity as the new class is learned over time. See **Supplementary Section S3** for more details).

We design the dopaminergic neuron with a resting potential higher than its firing potential, giving it a self-firing property. It is additionally suppressed via inhibitory connections from the other neurons in its layer so that it only spikes when they do not. This setup allows the dopaminergic neuron to fire only when novel information is detected.

When it fires, the dopaminergic neuron enhances plasticity by temporarily boosting the learning rate of the other neurons in its layer all the way to one while simultaneously stimulating firing in those other neurons via excitatory synaptic connections that we are calling *dopaminergic weights*. Because of the lateral inhibition discussed previously, once one of the stimulated neurons fires, it prevents or reduces the probability of the other neighboring neurons from firing. A neuron with a boosted learning rate then resets its learning rate the next time it fires or receives an inhibitory signal from a neighboring neuron, indicating that one of its neighbors has fired. Thus, while the dopamine signal is sent to many neurons, only the first neuron(s) to fire undergo the enhanced plasticity, creating heterogeneous plasticity and allowing the dopamine signal to perform an isolated targeting for local, rapid adaptation rather than global interference. Temporarily modulating the learning rate to the full value allows the first neuron that responds during a dopamine release to undergo a one-shot rapid learning of the current, novel sample and be "reassigned" without corruption from its old weight values. Then the learning rate is reset, allowing the representation to generalize with traditional, gradual weight changes. **Figure 3** presents an example of the dopaminergic neuron in operation. The dopaminergic neuron fires for novel representations and does not fire if an input is similar to one already seen.

Due to the rapid learning that occurs in the presence of dopamine and the lack of traditional homeostatic threshold dynamics, we also modify the STDP learning rule for improved stability, discussed later in more detail in section 2.4.2.2.

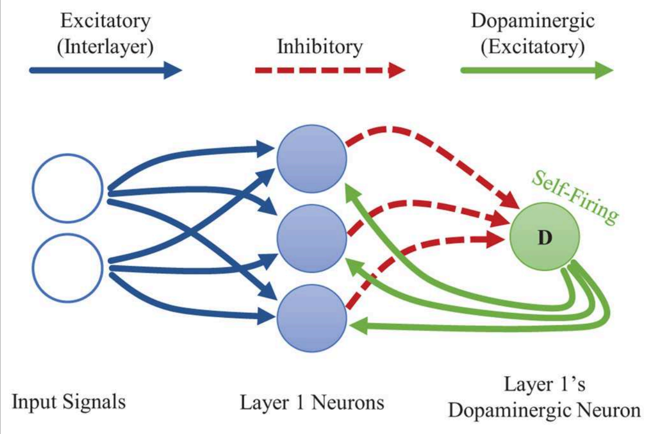


FIGURE 2 | Single-layer CFN architecture. The dopaminergic neuron fires when the other neurons on its layer are *not* firing, often a sign of novel information. The firing of the dopaminergic neuron stimulates firing in the other neurons while temporarily enhancing plasticity. The stimulation signals from the dopaminergic neuron are weighted to provide heterogeneous, targeted stimulation. The other neurons within a layer each have additional laterally inhibitory connections for competition (not shown here).

2.2.2. Targeted Stimulation for Controlled Forgetting via Trained Dopaminergic Weights

We have addressed how to make the forgetting process rapid and local in order to reduce interference between old and new information. However, we must also control the specific locality of the forgetting so as to maintain high accuracy for previous tasks. A uniform stimulation would cause the neuron that is angularly closest to the input to fire first and adapt to the novel information, independent of how useful that neuron is for previous tasks. When the network is refining representations that have already been seen, adjusting the closest weight vector is appropriate to promote generalization. However, when novel

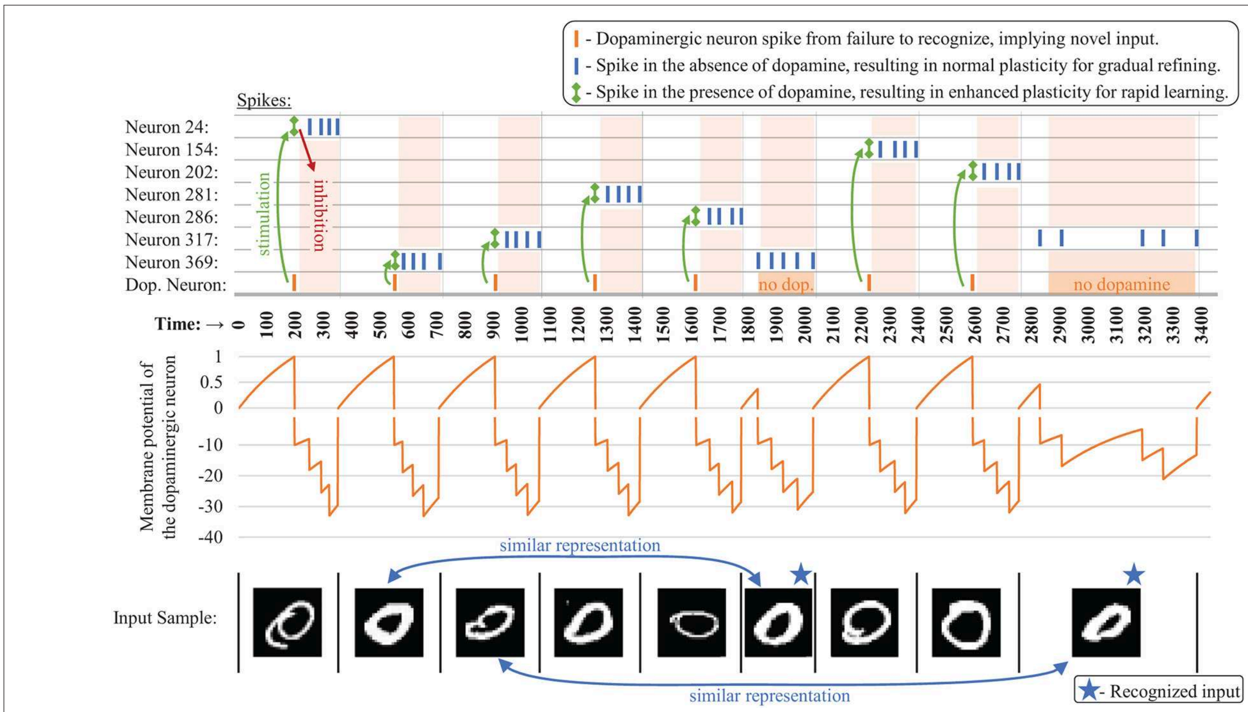


FIGURE 3 | Example spiking activity **(top)** showing the interaction between the dopaminergic neuron and the other neurons during training for the first several samples **(bottom)** for a CFN of 400 neurons. Only neurons that fired during this small time interval are shown. The membrane potential of the dopaminergic neuron is shown **(middle)**, demonstrating its self-firing property unless inhibited, with a firing threshold at 1 (A break is shown in the graph at zero indicating a different scale for the positive and negative values).

data in presented in high-dimensional space, the closest neuron is more likely to be one that has already learned a distribution from a previous class rather than an unused neuron that is completely uncorrelated. Thus the stimulation must be controlled to avoid overwriting the most essential information from previous tasks. We provide this control by heterogeneously stimulating the other neurons to fire during the release of dopamine via the excitatory dopaminergic weights. Training these weights allows specific neurons to be targeted to undergo forgetting and re-learning.

To minimize accuracy degradation caused by forgetting, we would ideally like to forget outlier or stale information rather than commonly-used or recent information that may be essential for returning to previous tasks, applying knowledge from old tasks to new tasks, or generalizing the rapidly learned novel information. As a proxy for this categorization, we target neurons with low overall firing frequency (outliers) or less recent firing activity (stale). Considering firing age over firing frequency is a tunable parameter that controls how much if any preference should be given to more recent tasks. For the experiments in this paper, we consider all tasks as equally important no matter how recently seen, so we target neurons with low firing frequency.

For these purposes, we enact a simple local learning rule: a dopaminergic weight depresses each time its post-synaptic neuron fires. This rule causes a dopaminergic weight to be smaller when the post-synaptic neuron it is targeting has a higher firing rate, and vice versa. To maintain positive values, the depressions are proportional to the current value, causing an exponential decay. Otherwise, the dopaminergic weights experience a gradual potentiation. Potentiation must occur to prevent the weights from tending toward zero with differences between weights too small to distinguish on implementations with finite precision. The rate of potentiation is irrelevant in our setup as long as it is the same for all dopaminergic weights in the layer, maintaining their relative values, because the dopaminergic neuron continues to send the dopaminergic signal until one of the other neurons in the layer fires. For the experiments in this work, we effect this potentiation by L2-normalizing the fan-out vector of dopaminergic weights after a depression.

2.3. Models

In this subsection, we describe the input, synapse, and neuron models and associated probability distributions that are useful in selecting the appropriate hyperparameters for unsupervised lifelong learning.

2.3.1. Input Encoding

Input samples are encoded as Poisson spike trains, following the mathematical model of a Poisson point process (PPP), where the spike rate λ_i of an input neuron is proportional to the pixel intensity of input i. Thus, the number of spikes in a given time window follows the distribution of a Poisson random variable with an expectation proportional to the input value. For perception tasks on static images, there is no temporal information in a single sample, and thus rate encoding is one of the most common encoding methods for SNN image perception implementations as it maintains statistical independence between individual input spikes, which is useful for the computationally less expensive one-sided STDP curve, discussed later in section 2.4.2[1].

Each spike is modeled as a time-shifted delta function. The precise time of the *kth* most recent spike from input i is represented as t_{ik}. Being a PPP, the timing between two sequential spikes on a given input channel are drawn from an exponential random distribution, also with rate λ_i. The time passed since the *kth* most recent spike from i at time t is represented as $t_{|ik|} = t - t_{ik}$ and follows the distribution of a gamma random variable $T_{|ik|} \sim gamma(\alpha = k; \beta = \lambda_i)$. The vector of all input rates for each dimension of the given sample is represented as $\vec{\lambda}$.

2.3.2. Synapse Model

We model the synaptic connections between neurons as a multiplicative weight which is applied to the delta spike from its pre-synaptic neuron and then added to the membrane potential of its post-synaptic neuron, creating a exponential kernel response[2]. We represent the weight of the synapse connecting input i to neuron j as w_{ij} and the vector of all inputs to neuron j as $\vec{w}_j$.

2.3.3. Spiking Neuron Model

We use the common Leaky-Integrate-and-Fire (LIF) neuron model, in which a neuron's membrane potential v_{mem} undergoes a continuous decay according to the differential equation in (1), where τ_{mem} is the membrane decay constant and v_{rest} is the resting potential. The membrane potential is also potentiated or depressed by incoming excitatory or inhibitory signals, respectively. If the membrane potential reaches or surpasses the neuron's firing threshold v_{th} then the neuron fires, producing an output spike and resetting its potential to v_{reset}. Without loss of generality, we set v_{rest} to zero as a reference voltage. For model and evaluation simplicity, we also set v_{reset} to zero and have no refractory periods.

$$\dot{v}_{mem} = \frac{-(v_{mem} - v_{rest})}{\tau_{mem}} \tag{1}$$

2.3.4. Membrane Potential Distribution

To estimate the relative firing distributions of competing LIF neurons, it is useful to understand the distribution of their membrane potentials. Assuming a firing event has yet to occur, the effect of a Poisson spike train on a neuron's membrane potential with exponential leakage may be viewed as a shot-noise process (Hohn and Burkitt, 2001). A Poisson spike train from input i is the summation of many spikes represented as delta functions:

$$N_i = \sum_k \delta_{T_{|ik|}} \tag{2}$$

[1]Other input encodings that use time-encoding such as rank-order may provide energy efficiency improvements but at the moment provide no obvious benefit in addressing catastrophic forgetting and are thus beyond the scope of this work.

[2]Non-instantaneous potentiation kernels, such as the alpha response, are beyond the scope of this work due to the added difficulty to event-driven simulation.

This stochastic process produces the following pre-firing membrane potential induced on neuron j by the spike train from input i:

$$V_{ij}(t) = \int f_{ij}(t)\, N(dt) = \sum_k f_{ij}(t - T_k), \tag{3}$$

where $f_{ij}(t) = w_{ij} e^{-t/\tau_{mem}}$. The Laplace transform of this shot-noise process is:

$$\mathcal{L}(\theta) = E[e^{-\theta V_{ij}(t)}] = e^{g(\theta)} \tag{4}$$

where $g(\theta) = \lambda_i \int_0^t (e^{-\theta f_{ij}(v)} - 1)dv$.

2.3.4.1. Mean pre-firing membrane potential

The 1st moment, which is the mean pre-firing potential caused by input channel i, is given by:

$$\begin{aligned} E[V_{ij}(t)] &= -\left[\frac{d\mathcal{L}(\theta)}{d\theta}\right]_{\theta=0} = -\left[\frac{de^{g(\theta)}}{d\theta}\right]_{\theta=0} \\ &= -\left[e^{g(\theta)}\right]_{\theta=0}\left[\frac{dg(\theta)}{d\theta}\right]_{\theta=0} \\ &= -\lambda_i\left[\int_0^t (-f_{ij}(v)e^{-\theta f_{ij}(v)})dv\right]_{\theta=0} \\ &= \lambda_i \int_0^t f_{ij}(v)dv = \lambda_i w_{ij}\tau_{mem}(1 - e^{-t/\tau_{mem}}) \end{aligned} \tag{5}$$

For all inputs, represented as the rate vector $\vec{\lambda}$, the mean combined pre-firing potential of neuron j is:

$$\begin{aligned} E[V_j(t)] &= \tau_{mem}\sum_i \lambda_i w_{ij}(1 - e^{-t/\tau_{mem}}) \\ &= \tau_{mem}(\vec{w}_j \bullet \vec{\lambda})(1 - e^{-t/\tau_{mem}}) \end{aligned} \tag{6}$$

In steady-state this converges to: $\tau_{mem}(\vec{w}_j \bullet \vec{\lambda})$, which is important for discussions later in sections 2.4.2, 2.4.3.1.

2.3.4.2. Variance of pre-firing membrane potential

Continuing to the second moment, we can calculate the variance of the pre-firing membrane potential that is induced on neuron j by incoming spikes received from input i:

$$\begin{aligned} Var(V_{ij}(t)) &= E[V_{ij}(t)^2] - E[V_{ij}(t)]^2 \\ &= \left[\frac{d^2\mathcal{L}(\theta)}{d\theta^2}\right]_{\theta=0} - E[V_{ij}(t)]^2 \\ &= [e^{g(\theta)}(g'(\theta)^2 + g''(\theta))]_{\theta=0} - E[V_{ij}(t)]^2 \\ &= E[V_{ij}(t)]^2 + \lambda_i\int_0^t f_{ij}(v)^2 dv - E[V_{ij}(t)]^2 \\ &= \frac{1}{2}\lambda_i \tau_{mem} w_{ij}^2(1 - e^{-2t\tau_{mem}}) \end{aligned} \tag{7}$$

The combined variance of the potential induced by all inputs is:

$$\begin{aligned} Var(V_j(t)) &= \frac{1}{2}\tau_{mem}\sum_i \lambda_i w_{ij}^2(1 - e^{-2t\tau_{mem}}) \\ &= \frac{1}{2}\tau_{mem}(\vec{\lambda} \bullet \vec{w}_j^{\circ 2})(1 - e^{-2t\tau_{mem}}) \end{aligned} \tag{8}$$

where $\vec{w}_j^{\circ 2}$ represents the Hadamard square of the weight vector. This equation is important for discussions later in section 2.4.2.2.

2.4. Experimental Methodology

To evaluate the effectiveness of our proposed lifelong learning approach, we simulated CFNs on the MNIST dataset (Lecun et al., 1998) on network sizes of 400, 900, 1,600, 2,500, 3,600, 4,900, and 6,400 excitatory neurons, each for five different seeds. We compare the CFNs that have dopaminergic neurons to the same setups without dopaminergic neurons, both with and without homeostasis from adaptive thresholding. This section details the experimental methodology of the simulations for the CFNs and the comparison networks.

2.4.1. Simulation Setup

Each network was evaluated on the MNIST dataset for two different scenarios: (1) *interleaved* classes where classes are distributed uniform randomly, providing the network with samples from each class throughout the entire training process; and (2) *disjoint* classes where all samples from one digit are presented before moving to the next digit and never returning to previous digits after changing classes. The first scenario is meant to represent traditional offline training in which all training data is already available. The second scenario is meant to test lifelong learning by representing a changing environment with samples presented in the worst-case possible ordering–entirely sequential. In both scenarios, class labels are not provided during training. This means that the network receives no external indication of when a task/digit change occurs in the disjoint scenario.

Other than sequentializing the MNIST dataset in the disjoint scenario, our training and testing procedure follows closely with that of Diehl and Cook (2015), who demonstrated competitive unsupervised STDP training on MNIST in the traditional interleaved scenario.

2.4.1.1. Training process

In both training scenarios, samples are presented one-by-one to the network. For the current sample, input neurons fire at the sample rate until the system registers at least five output spikes, as followed by Diehl and Cook, which is generally enough to confidently identify the input in view of the stochasticity in the SNN.

In contrast to Diehl and Cook, if a given sample does not produce enough output spikes we do not continue to increase the input firing rate during training on the CFNs, since the dopaminergic neuron takes care of stimulating neurons in the absence of a good match. After five output spikes are registered, all membrane potentials are reset to avoid one sample interfering with the next, and then the next sample is presented. Details of the STDP learning rule implementation are provided in section 2.4.2.

2.4.1.2. Testing process

In the disjoint scenario, we measure effective lifelong learning over time by evaluating each network after each task change to determine its current accuracy for all classes seen up to that point. Networks in the interleaved scenario are only evaluated at the end of the training process. During evaluation we pause learning and freeze network parameters to prevent samples from older classes or samples from the testing set from affecting the network.

As training is performed entirely without supervision and without knowledge of a task change, the final network outputs must be assigned class labels for evaluation. While the network is frozen, label assignment is done by inference on the training set, followed by evaluation on the testing set. The MNIST dataset is already highly clustered in its input space, and therefore a supervised linear classifier is already capable of competitive accuracy. Because of this, no final linear readout classification layer is added to avoid the label assignment process acting as a traditional supervised linear classifier. Instead, following the unsupervised evaluation method of Diehl and Cook, each trained neuron is *directly* assigned a class label and no linear combination of these neuron outputs is performed. Rather, the class decision is winner-take-all, choosing the class of the neuron that spiked the most for that sample. As a control, we also perform this same label assignment and evaluation process on networks with randomized weights, also averaged over five seeds, to compare with the accuracy achievable solely by this label assignment process.

With frozen parameters, dopaminergic adaptation does not occur during label assignment and testing set evaluation. Instead, for inference a poorly-recognized input is assigned the class of the closest trained neuron by continuing to increase input firing rates until a sufficient response is recorded as is done in the other networks.

2.4.1.3. Event-driving computation

Using exponential kernels, we treat spikes as inducing instantaneous voltage potentiations in the respective post-synaptic neuron membranes with exponential decay. As such, neurons only fire upon receiving an incoming spike and will not fire between incoming spikes, with the exception of the dopaminergic neurons which are handled separately. This allows us to emulate the networks using purely event-driven computation rather than breaking time into discrete time steps and updating neurons states at each time step. Because we encoded input spike trains as Poisson point processes, the time between spikes is an exponential random variable with $\lambda_i = input_i$. Therefore, rather than incrementing time in fixed intervals, we calculate the time until the next input spike arrival and decay all the traces and membrane potentials according to that time interval before processing that input spike.

The dopaminergic neurons are an exception, as they fire in the *absence* of input spikes. Therefore, before processing an input spike, we first check to see if the dopaminergic neuron would have fired earlier, in which case, it is processed at its respective time interval first.

2.4.2. STDP Learning

STDP's Hebbian learning rule involves potentiation or depression of a synaptic weight based on the timing of firing events.

This section details how it is implemented and modified for these experiments.

2.4.2.1. One-sided STDP

As the input information in our system is encoded only in the spike rate, we can employ the computationally less-expensive one-sided version of STDP, evaluated at the post-synaptic firing event:

$$\Delta w = \alpha(pre - offset) \tag{9}$$

where α is the learning rate, *pre* is a trace of pre-synaptic firing events, and *offset* is the value to which the pre-synaptic traces are compared, determining potentiation or depression.

The *pre* trace follows a similar distribution as the membrane potential (see section 2.3.4), only with a different time constant and without being weighted by the synapse, and so its expected value is also proportional to the input spike rate (e.g., $E[pre_i] = \lambda_i \tau_{pre}$). Correlated potentiations in the direction of $\vec{pre}$ therefore provide Hebbian learning by angularly migrating $\vec{w}$ toward the angle of the input vector $\vec{\lambda}$. Anti-Hebbian depression reduces weights from uncorrelated inputs and is provided by subtracting the *offset* term for one-sided STPD rather than performing additional weight processing at pre-synaptic firing events.

2.4.2.2. Stabilizing STDP

Typically, *offset* is a constant value identical across all dimensions and can be thought of as a scaled ones vector, applying uniform anti-Hebbian depression. Such uniform depression does not, however, create a weight change in exactly the direction desired (see **Figure 4**) and causes instability in the STDP learning rule. This instability is usually controlled by adaptive thresholding and weight capping via exponential weight-dependence.

However, our CFNs with rapid one-shot dopaminergic learning of novel inputs cannot use such gradual approaches to stabilize. We provide the required stability to this STDP learning rule by correcting the direction of the weight change. Rather than a constant offset, we dynamically tie *offset* to the current weight value, which is an adaptation based on Oja's rule (Oja, 1982). To place *pre* and the weight on the same scale, we scale the pre-synaptic trace by the inverse of its decay rate τ_{pre}, changing (9) to:

$$\Delta w = \alpha\left(\frac{pre}{\tau_{pre}} - w\right) \tag{10}$$

This corrected weight change allows our CFNs to rapidly and accurately capture information from novel inputs during dopaminergic learning and otherwise gradually stabilize on the center of the cluster of input samples for which it has fired. Additionally, the stochasticity of the presynaptic trace can allow the values of some dimensions to significantly overshoot or undershoot their mean. Because of the rapid learning in the presence of dopamine, we capped each individual weight between 0 and 0.2 before normalization to prevent the outliers from distorting the normalization.

2.4.2.3. Modulating STDP

Dopaminergic modulation of plasticity is implemented by dynamically changing the learning rate α. During normal operation, α is set to 0.01 for gradual generalizing refinement of the synaptic weights. When the dopaminergic neuron fires, α is temporarily set to one for the reasons discussed in section 2.2.1.

2.4.2.4. Normalization

The MNIST dataset is a magnitude insensitive dataset, meaning that increasing or decreasing the intensity of a sample does not alter its class and that angular distance is more important than Euclidean distance. As given in (6), the mean pre-firing potential of a spiking neuron is proportional to the L2-norm of its weight vector and also to the L2-norm of the input rate vector. Although a larger mean pre-firing potential does not always correspond to a larger firing rate due to differing variances caused by the Hadamard square of the weight vector as shown in (8), the correlation between $E[V]$ and the firing rate sufficiently holds for datasets like MNIST with inputs of large enough dimensions and fairly comparable input sparsity between samples.

As such, for a given input and assuming equal weight vector magnitudes, the neuron that is angularly closest to the input will be more likely to fire, allowing for unsupervised Hebbian learning by training neurons on correlated inputs. Therefore, we L2-normalize each neuron's weight vector, and for the same reason the input rate vectors are also L2-normalized. Weight normalization has recently been shown to occur in biology (El-Boustani et al., 2018) and may still be considered a localized function, as the processing can occur at the post-synaptic neuron to which all the weights in a given weight vector are directly connected.

2.4.3. Timing and Time Constants

As our evaluations and simulations are purely event-driven, the concept of discrete computational time steps is not applicable. Timing parameters are thus purely relative. Therefore, without loss of generality, the L2 normalized input rate vectors were defined as having an L2 rate magnitude of one spike per time unit, and all other timing values are relative to that. This subsection discusses the timing values used in the simulations.

2.4.3.1. Membrane decay time constant

According to Equation (6), the expected value of the membrane potential saturates in time according to $(1 - e^{-t/\tau})$. A smaller τ results in a faster convergence to the steady state, or, equivalently, fewer input spikes to converge. E.g., in five time constants, the expected potential reaches over 99% of is steady-state value. However, using (8), the steady state standard deviation of the potential in proportion to the mean decreases as the decay rate increases:

$$\frac{\sqrt{Var(V)}}{E[V]} \propto \frac{1}{\sqrt{\tau}} \tag{11}$$

Thus, a larger membrane decay constant is better for proper discrimination between two differing inputs, but increases the number of computations. For the L2-normalized MNIST

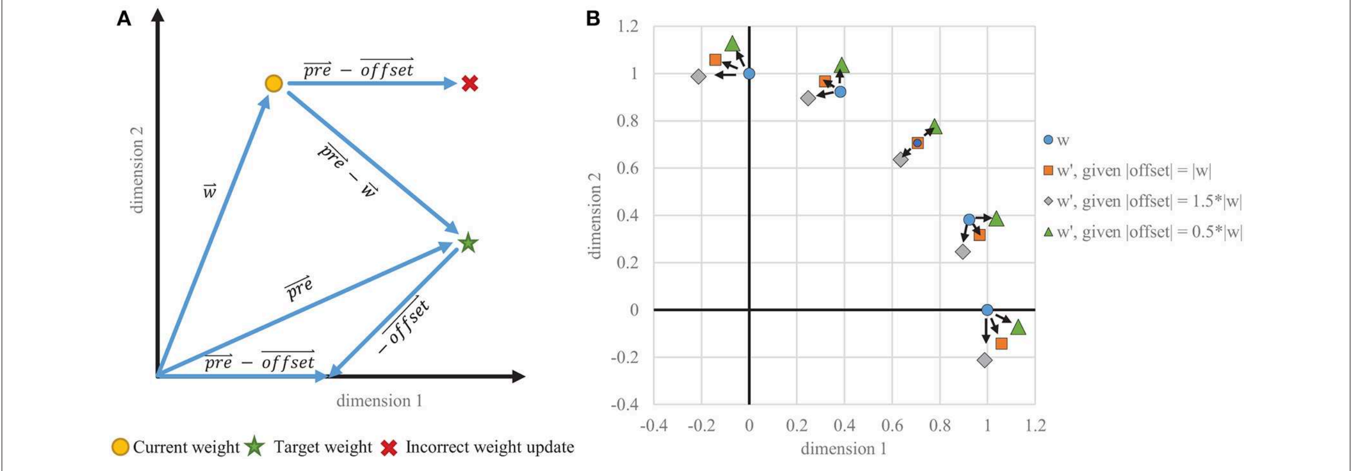


FIGURE 4 | Instability of one-sided STDP. **(A)** Example vectors showing how a static offset does not result in a correct weight change. The goal is to migrate $\vec{w}$ toward the $\vec{pre}$ trace, which is proportional to the input $\vec{\lambda}$. The $\overrightarrow{offset}$ vector that is subtracted from $\vec{pre}$ must be dynamically tied in each dimension to $\vec{w}$, rather than being the same in every dimension. **(B)** Weight change results for various starting positions where the target vector is equal to the current weight vector, which would ideally result in no weight change. With a static offset in each dimension, even scaled to the appropriate magnitude, the weight vectors do not stabilize on the target and instead migrate toward the axes, creating binarized weights when capped at zero.

dataset with 784 input dimensions, the angular distances between samples of differing classes are close enough to require at least 10 to 15 normalized time units for τ_{mem} in order to successfully establish a firing threshold that can discriminate between classes, and so τ_{mem} was set to 15 time units.

2.4.3.2. Time to recognize

A τ_{mem} of 15 still produces enough variance according to (8) that two to three time constants (between 30 and 45 time units) is on average sufficient time for the potential to rise above its steady-state mean. As mentioned earlier, we identify successful recognition of an input sample after registering five output spikes. Therefore, a total of 150–225 time units was generally sufficient to produce five sequential firing events in a reference vector neuron with a center close to the input.

In our simulations, we found little accuracy change by adjusting this hyperparameter within this range as long as the threshold voltage was appropriately tuned, so we fixed the time to recognize at 200 normalized time units for each simulation. We tuned the dopaminergic neuron to fire after those 200 time units unless it has been otherwise inhibited as discussed in section 2.2. Specifically, with v_{reset} set to zero as a reference voltage, the dopaminergic neuron's firing threshold v_{th} was set to one with a resting voltage set higher at two, causing the membrane potential to rise until it fires. Setting its rising time constant to $\frac{200}{ln(2/1)}$ then meets this objective. **Figure 3** shows the membrane potential of the dopaminergic neuron during simulation for the first several samples as an example of its operation over time.

We also set τ_{pre} to the same timing value of 200 time units to capture as much of the input train as possible because of the rapid one-shot dopaminergic learning of novel samples.

2.4.4. Determining v_{th} Without Adaptive Thresholding

As discussed in section 2.1.2, adaptive thresholding for homeostasis can interfere with lifelong learning on changing input distributions by temporally and spatially distributing the firing activity. Long-term adaptive thresholding may still be used with controlled forgetting if properly tuned, but our proposed method of enhanced plasticity and stimulated firing of infrequently-firing neurons is itself a form of deliberate, controlled homeostasis. Therefore, for a more accurate evaluation of the CFNs, we do not have the CFNs employ any adaptive thresholding–having static thresholds instead. With normalized weight vectors and input vectors, the larger the ratio $v_{th} : E[V(t)]$ the closer the input rate vector must be angularly to the weight vector to produce a given firing probability. Determining the proper v_{th} without dynamic adaptation, therefore, depends on the tightness of the clustering in the dataset. With this context, we included v_{th} in our hyper-parameter search, discussed next.

2.4.5. Hyper-Parameter Sweep

SNNs are known to be highly sensitive to hyper-parameters, especially during unsupervised learning without error signals to provide dynamic corrections. We perform a small search in the hyper-parameter space, adjusting v_{th} and the number of training epochs. Results from this search are shown in **Table 1**, with hyperparameters resulting in the best accuracy highlighted for each size. Good machine learning practice requires that we choose the system parameters based only on the training set, so only training set accuracy results are shown here. Testing accuracy results are discussed later in the section 3. A similar hyper-parameter sweep was performed for the non-dopaminergic SNNs that also do not have homeostatic adaptive thresholding, as well as for the SNNs with randomized weight vectors.

TABLE 1 | Training accuracy results of hyper-parameter sweep for each network size across both v_{th} and number of training epochs per task.

		# of training epochs per task			
Neurons	v_{th}	**1 (%)**	**5 (%)**	**10 (%)**	**20 (%)**
400	13.50	87.63	83.82	79.23	74.15
	13.75	87.63	84.07	78.34	75.18
	14.00	86.64	82.49	77.11	75.20
	14.25	85.50	83.59	75.75	75.06
900	13.50	89.78	91.07	89.36	85.31
	13.75	89.11	90.91	89.81	86.24
	14.00	87.83	91.47	89.71	84.75
	14.25	86.82	91.46	89.94	84.14
1,600	13.50	91.54	92.42	92.21	91.35
	13.75	91.24	92.87	92.48	91.40
	14.00	90.08	93.34	92.20	91.30
	14.25	88.48	93.06	92.85	91.74
2,500	13.50	93.15	93.62	93.46	93.13
	13.75	92.82	93.65	94.06	93.20
	14.00	91.80	93.37	94.28	93.53
	14.25	90.06	93.18	94.04	93.49
3,600	13.50	93.88	94.09	94.04	93.80
	13.75	93.90	94.12	94.48	94.52
	14.00	93.31	94.02	94.53	94.52
	14.25	92.33	93.27	94.40	94.27
4,900	13.50	94.51	94.91	94.67	94.77
	13.75	95.00	94.92	94.82	95.09
	14.00	94.61	94.85	94.97	95.21
	14.25	93.59	93.82	94.54	95.29
6,400	13.50	95.39	95.25	95.28	95.25
	13.75	95.42	95.55	95.39	95.68
	14.00	95.33	95.59	95.42	95.79
	14.25	94.79	94.99	95.21	95.88

Highlighted cells are best configuration for each size.

2.4.5.1. Neuron firing thresholds, v_{th}

Based on the discussion above, v_{th} should be close to but slightly less than τ_{mem} in voltage units, which is set to 15 time units. For MNIST, we initially found that if v_{th} is much less than 13.5, a neuron may too likely fire for samples from other classes, while if v_{th} is much higher than 14.25, a neuron may not fire for very close samples, even different stochastic instances of the same sample. We therefore tested each setup with four different threshold values in this range: 13.5, 13.75, 14.0, and 14.25. Smaller networks require each individual neuron to capture a larger subset of input samples, generally requiring slightly lower thresholds than those in larger networks.

2.4.5.2. Number of training epochs

Larger networks can capture representations that are less common but still useful. As such, for larger networks more epochs within a class are required before proceeding to subsequent tasks in order to refine the less common representations. For smaller networks, on the other hand, more epochs may reinforce less useful outliers, making it more difficult to make room for subsequent tasks.

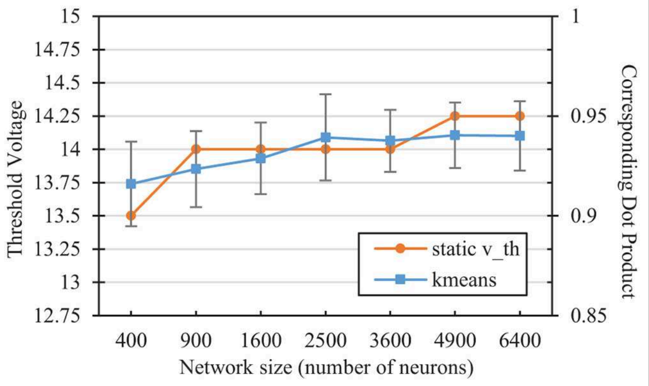


FIGURE 5 | Comparison of the static v_{th} selected in the hyperparameter sweep with the corresponding dot product of the nearest training error in a kmeans network of the same size. The kmeans error bars represent two standard deviations over 100 trials each.

2.4.6. Comparison of $E[V(t)]$ at v_{th} With K-Means Clustering Angular Error

We can compare the v_{th} values selected in the hyper-parameter search with the mean angular distance to a neuron's weight vector that would on average result in a membrane potential equal to that threshold. Performing a simple k-means clustering on the L2-normalized MNIST dataset yields information on the relative desired scope of each reference vector, depending on the number of reference vector neurons. **Figure 5** shows the dot product associated with the angular distance of the closest training sample/reference vector pair from differing classes for each network size after k-means clustering. The figure also shows the average membrane potential of a spiking neuron corresponding to these angles. For SNNs, neurons that are able to fire for samples that are further away than these angles are thus more likely to fire for samples of the wrong class. As the number of reference vector neurons increases, the portion of the input space per neuron decreases, improving accuracy by allowing each individual neuron to be more restrictive in its angular scope, which is relatively similar to those associated with the v_{th} values selected in the hyper-parameter sweep.

3. RESULTS

In this section, we present the results of simulating the CFNs and the non-dopamine comparison networks for the various sizes in both the interleaved classes scenario and the fully disjoint classes scenario. We present both the combined accuracy and the per digit accuracy, with final results and (in the disjoint scenario) results throughout the attempted lifelong learning process.

3.1. Combined, Across-Task Accuracy Results

Figure 6 shows the final combined, across-task classification accuracy of the CFNs and comparison networks for both the interleaved scenario and the disjoint scenario for all network sizes. The comparison with Diehl and Cook (2015) is provided for the network sizes for which results were published (400, 1,600, and 6,400). In the fully disjoint scenario, the 6,400 CFN achieves on average 95.24% classification accuracy across all digits, compared to 32.97% for a non-dopamine SNN without homeostasis, 61.95% accuracy for a non-dopamine SNN with homeostasis, and 53.30% accuracy for an SNN with random weights (Per-neuron activity statistics are available in the **Supplementary Material**).

Figure 7 shows the combined, across-task accuracy over time for the CFNs and comparison networks for network sizes of 1,600 and 6,400 neurons. (CFN results for the other sizes are available in the **Supplementary Material**). The combined, across-task accuracy over time is defined as classification accuracy on the portion of the testing set consisting of all previously-seen classes, up to and including the current task. For the 6,400 size, the CFN incurs its largest accuracy drop at the last stage, adding digit '9,' dropping 1.06 percentage points. In comparison, at that size the non-dopamine SNN without homeostasis incurs a 34.41 percentage point drop when adding digit '2,' the non-dopamine SNN with homeostasis incurs a 10.41 percentage point drop adding digit '9,' and the SNN with random weights incurs an 11.82 percentage point drop adding digit '2.'

3.2. Per-Digit Accuracy Results

Figure 8 shows the final accuracy of each individual task/digit by the end of the training process for 6400 neurons, comparing the distribution of accuracy across tasks for the CFNs in both the interleaved and disjoint scenarios, as well as with both the non-dopamine SNNs in the disjoint scenario and the randomized weights. In the disjoint scenario, the CFN's final worst performing class is digit '9' at 91.18% accuracy, which is also the worst performing class in the interleaved scenario at 93.60% accuracy. In comparison, for the other networks in the disjoint scenario, the final worst performing class is digit '8' at 38.81% accuracy for the non-dopamine SNN with homeostasis; digits '5,' '7,' and '8' tied at 0.00% accuracy for the non-dopamine SNN without homeostasis; and digit '8' at 33.37% accuracy for the SNN with random weights.

Figure 9 shows the per-digit accuracy over time for each network of 6,400 neurons. (Per-digit false positives over time are provided in the **Supplementary Material**). The CFN incurred its largest per-digit accuracy drop for digit '4' after adding digit '9,' decreasing 3.89 percentage points for digit '4' during that task change. In comparison, the non-dopamine SNN with homeostasis incurred a 23.07 percentage point drop for digit '4' at that same transition; the non-dopamine SNN without homeostasis incurred a 69.52 percentage point drop for digit '1' after adding digit '7,' and the SNN with random weights incurred a 10.98 percentage point drop in accuracy for digit '4' when adding digit '9.'

4. DISCUSSION

In this section, using a qualitative analysis we discuss reasons why the non-dopamine SNNs failed at lifelong learning in the disjoint scenario and how the CFNs avoided those failures. We also discuss the expected sequential penalty and graceful degradation of accuracy.

4.1. A Qualitative Analysis

In these fully-connected one-layer SNNs, each neurons weight vector can be viewed as a reference vector that captures a specific input representation, ideally successfully generalized. As such, we may qualitatively observe the success of dopaminergic learning over time by viewing these representations. For a better visual demonstration of the disjoint scenario, we show the weights of the networks for the first four digits '0' through '3' in **Figure 10**, with 100 neurons arranged in a 10x10 grid.

Note that in the CFN case (**Figure 10A**) there are two very distinct categories of representations. The digit representations that appear to have a more consistent pixel intensity and a more consistent line width and curvature are generalized representations refined by many similar samples in a cluster. On the other hand, the digit representations that appear less defined and with more irregularity in pixel intensity are outlier representations from only one or a few samples. Notice that the digit representations that are preserved from one task to another are the useful generalizations rather than the outliers, which on the other hand are the first to be overwritten when space for a new task is required. In addition, the representations that are preserved from previous tasks experience very little and infrequent corruption during later learning stages. The dopamine signals are able to successfully replace old information with new information without interference and while maintaining accuracy because of the targeted localization.

In contrast, we can visually see the failure of the non-dopamine SNNs in the disjoint scenario. In the network without homeostasis (**Figure 10B**) we see that only a few neurons experienced any learning. Without homeostasis the neurons that fired first migrated closer to the input distributions and dominated the firing activity. Even when the input distribution changed between tasks, the already used neurons were closer to the new distributions than the unused neurons with random weight vectors. Continuing the reuse the same neurons caused the SNN to overwrite and forget previous tasks.

Next, in the network with homeostatic adaptive thresholding (**Figure 10C**), we see a better use of network resources from the distributed firing activity. But without targeted dopaminergic modulation homeostasis distributes the learning for a new task over all the neurons previously used in earlier tasks. Even when the learning per-digit is reduced (**Figure 10D**), the activity for the new tasks are still globally distributed by the adaptive thresholding, causing corruption between tasks.

The CFNs with dopaminergic learning avoid globally distributing firing activity during a single task by not having

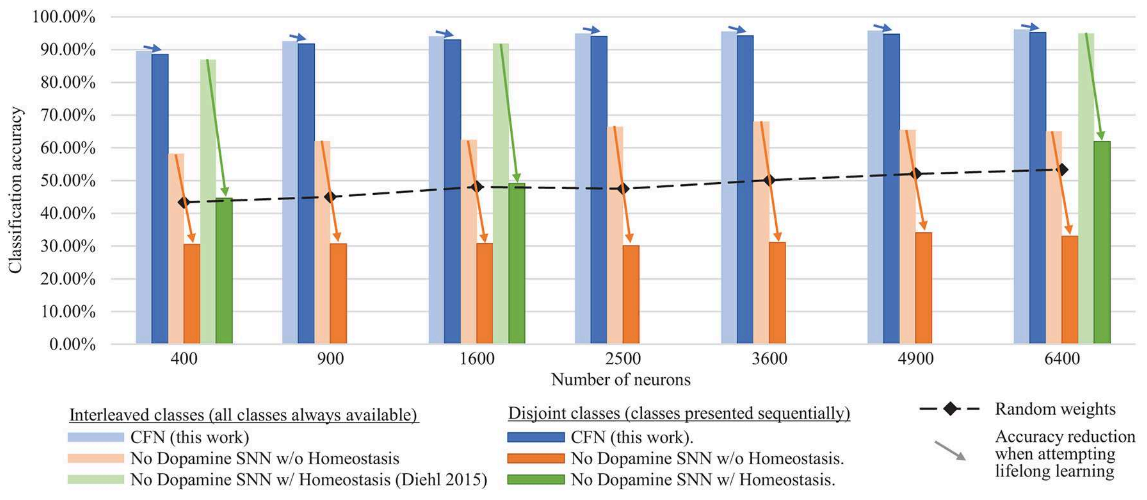


FIGURE 6 | Final classification accuracy at various sizes of the CFNs compared to SNNs without dopamine. Accuracy is shown for both the interleaved class scenario and the disjoint class scenario, showing the resulting accuracy reduction by sequentializing the classes. CFNs show average over five seeds.

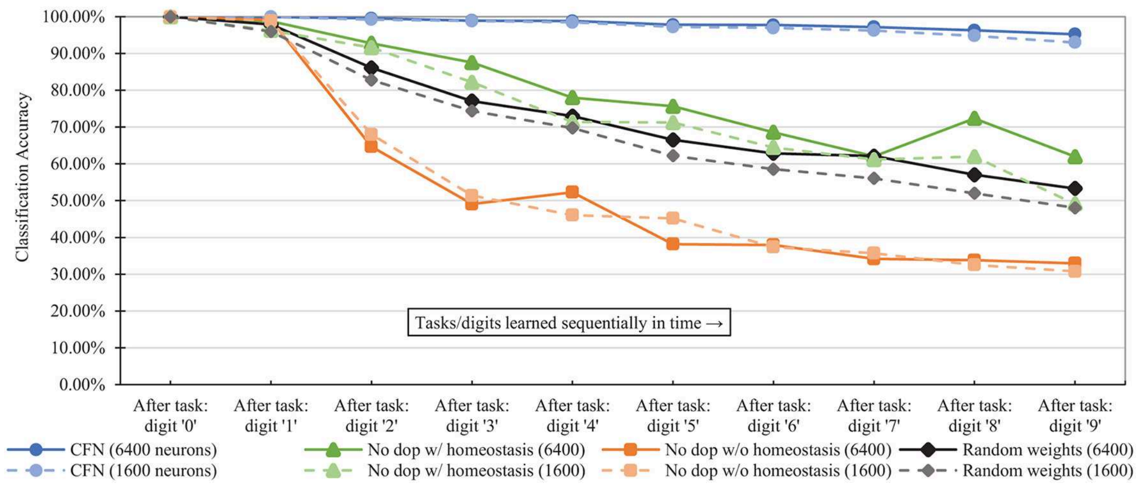


FIGURE 7 | Classification accuracy over time at each stage of the learning process (i.e., after each new task/digit) in the disjoint scenario, comparing the proposed CFNs to SNNs without dopamine and to the randomized weight control. Accuracy is for all previous tasks, up to and including the current task. CFN results are averaged over five seeds.

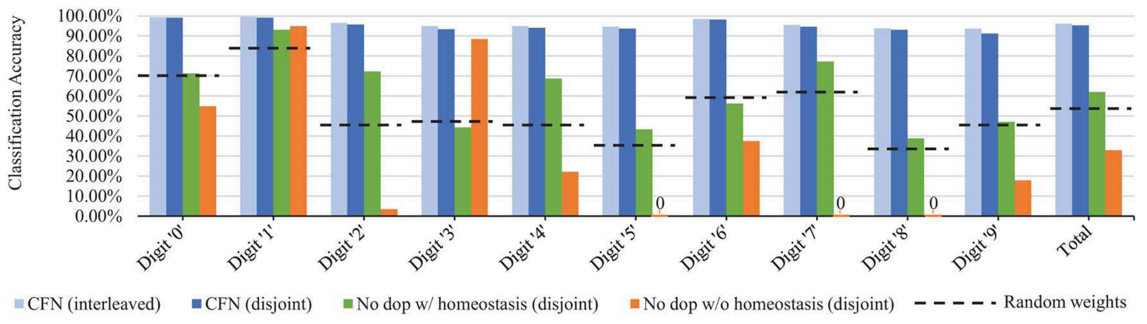


FIGURE 8 | Final per-digit accuracy (size 6,400), comparing interleaved CFN accuracy to the disjoint CFN accuracy. Also showing failure for individual digits in the disjoint scenario for SNNs without dopamine.

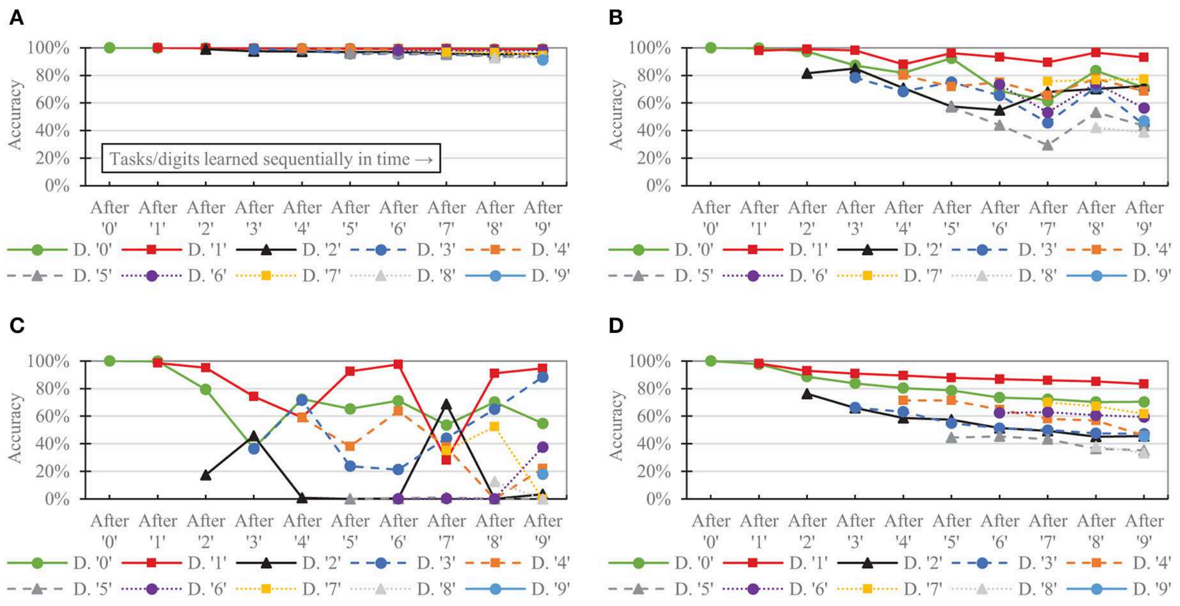


FIGURE 9 | Per-task/digit classification accuracy as new tasks/digits are added over time for the following networks, all of size 6,400: **(A)** the proposed CFN **(B)** no dopamine SNN with homeostasis **(C)** no dopamine SNN without homeostasis, **(D)** SNN with random weights.

traditional homeostatic adaptive thresholding. In addition, the CFNs avoid continuing to reuse the same neurons by proactively identifying novel data and targeting specific neurons to learn the novel data, preserving essential information from previous tasks.

We note that for the failed networks where older classes are entirely overwritten by new classes, the networks still report some, albeit poor, accuracy for the forgotten tasks. This is because the varied intra-class distributions can still be somewhat useful at differentiating inter-class distributions. For this purpose, the accuracy comparisons to the SNNs with random weights are essential at identifying catastrophic forgetting, indicating that around 40–50% is a failure baseline for unsupervised learning using SNNs of these sizes on the MNIST dataset.

4.2. The Expected "Sequential Penalty"

We see that the CFNs in the disjoint scenario perform on par with the interleaved scenario, averaging only a 1.04% accuracy reduction across all sizes. This penalty is expected due to sequentializing the tasks. In fact, such a penalty may be impossible to completely avoid, as the interleaved scenario provides more information to the network throughout training by providing all distributions up front, whereas the disjoint scenario never provides an opportunity to temporally overlap learning of different distributions. Even so, the sequential penalty for the CFNs is minimal, and may be acceptable given the systems avoidance of catastrophic failure in the disjoint scenario. In fact, even with this penalty, the 6400 neuron CFN achieves a respectable 95.24% test accuracy after lifelong learning, which we believe is the best unsupervised accuracy ever achieved by a fixed-size, single-layer SNN on a completely disjoint MNIST dataset. The CFNs in the disjoint scenario even outperform (Diehl and Cook, 2015) in all cases for which they provide results, even though that work is in the interleaved scenario.

4.3. Graceful Degradation Instead of Catastrophic Forgetting

Controlled forgetting allows the network to gracefully degrade its accuracy in exchange for the ability to learn new tasks with limited resources, rather than failing. The true success of a lifelong learning system is shown not just by the final accuracy, but also by its performance throughout the training process and across training tasks. Notice how in **Figure 8** while the system expectedly performs better for some tasks rather than others, there is no single task for which the system fails; i.e., the sequential penalty is spread between tasks. In fact, the lifelong system performs best at the same tasks (digits '0,' '1,' and '6') and worst at the same tasks (digits '8' and '9') that the offline/non-lifelong system does.

We believe that this type of approach with modulated plasticity and targeted stimulation can be useful for allowing deployed systems to gracefully adapt to changing environments rather than failing to adapt or requiring frequent offline retraining.

4.4. Future Work

We expect that a deeper network will improve accuracy beyond that of these results and allow for learning of more complicated datasets. As mentioned earlier, in a deeper network, it may be that only the last few layers would require lifelong learning, performing a readout from a liquid state

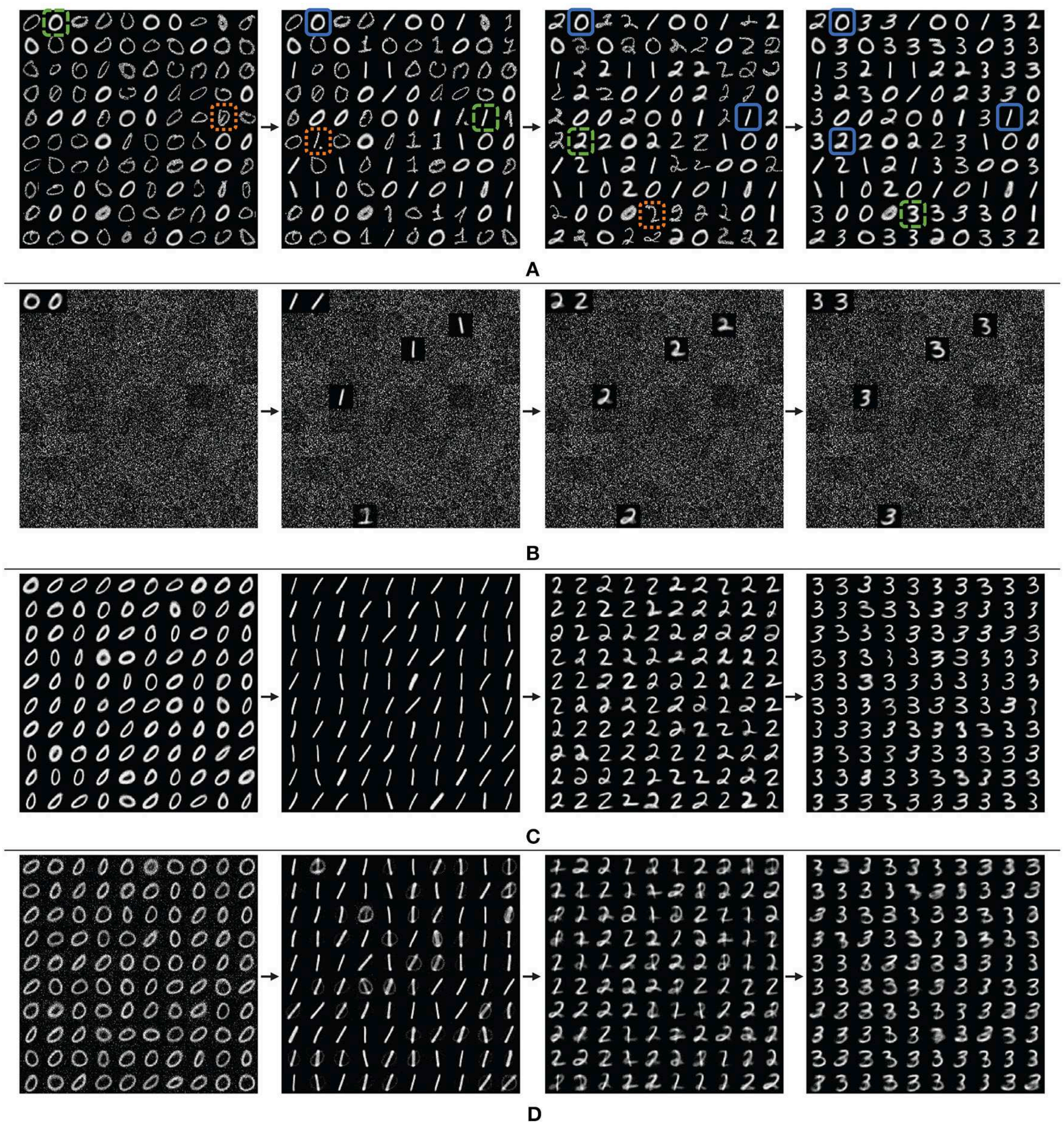


FIGURE 10 | Grid view of the weight vectors of reference neurons over time, showing the first four digits, learning ‘0’ through ‘3’ for **(A)** the proposed CFN, **(B)** a non-dopamine SNN without homeostasis, and **(C)** a non-dopamine SNN with homeostasis, each with 400 neurons, although only the 100 top-firing neurons are shown for space. For the CFN, digits highlighted in dashed green are examples of successfully learned generalized representations. Digits highlighted in dotted orange are examples of outlier representations. Digits highlighted in solid blue are examples of representations preserved from previous tasks. Also shown is **(D)** another non-dopamine SNN with homeostasis, but with reduced learning on each digit, showing catastrophic interference between classes causing corruption.

machine or a fixed feed forward network sufficiently pre-trained on low-level representations. We also plan to evaluate this method on time-encoded signals to improve sparsity and energy efficiency. Further, we hope to explore other dopaminergic weight adjustment policies that have a higher time-dependence or weight policies with habituation, such as in Panda et al. (2018), in order to allow for operation in an environment of changing priorities, and not just temporally separated tasks.

4.5. Conclusion

We presented a biologically-inspired dopaminergic modulation of synaptic plasticity to exploit STDP locality. Trained stimulation during the presentation of novel inputs allows the system to quickly perform isolated adaptation to new information while preserving useful information from previous tasks. This method of controlled forgetting successfully achieves lifelong learning. Our Controlled Forgetting Networks show only a slight reduction in accuracy when given the worst possible class ordering, i.e., completely sequential without revisiting previous classes, while successfully avoiding catastrophic forgetting.

DATA AVAILABILITY STATEMENT

The MNIST dataset used in this study can be found at http://yann.lecun.com/exdb/mnist.

AUTHOR CONTRIBUTIONS

JA wrote the paper and performed the simulations. Both JA and KR helped with developing the concepts, conceiving the experiments, and writing the paper.

FUNDING

This work was supported in part by C-BRIC, a JUMP center sponsored by the Semiconductor Research Corporation and DARPA, and by the National Science Foundation, Intel Corporation, and the Vannevar Bush Fellowship. The authors declare that this study received funding from Intel and the Semiconductor Research Corporation. The funders were not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

ACKNOWLEDGMENTS

The authors would also like to thank Professor Jonathon Peterson and Dr. Gerard (Rod) Rinkus of Purdue University for helpful discussions.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2020.00007/full#supplementary-material

REFERENCES

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). "Memory aware synapses: Learning what (not) to forget," in *Computer Vision – ECCV 2018*, eds V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss (Cham: Springer International Publishing), 144–161.

Allred, J. M., and Roy, K. (2016). "Unsupervised incremental stdp learning using forced firing of dormant or idle neurons," in *2016 International Joint Conference on Neural Networks (IJCNN)* (Vancouver, BC), 2492–2499. doi: 10.1109/IJCNN.2016.7727509

Bashivan, P., Schrimpf, M., Ajemian, R., Rish, I., Riemer, M., and Tu, Y. (2019). Continual learning with self-organizing maps. *arXiv:1904.09330 [Preprint]*. Available online at: https://arxiv.org/abs/1904.09330

Dhoble, K., Nuntalid, N., Indiveri, G., and Kasabov, N. (2012). "Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning," in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane, QLD), 1–7. doi: 10.1109/IJCNN.2012.6252439

Diehl, P., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9:99. doi: 10.3389/fncom.2015.00099

Du, X., Charan, G., Liu, F., and Cao, Y. (2019). Single-net continual learning with progressive segmented training (PST). *arXiv:1905.11550 [Preprint]*. Available online at: https://arxiv.org/abs/1905.11550

El-Boustani, S., Ip, J. P. K., Breton-Provencher, V., Knott, G. W., Okuno, H., Bito, H., et al. (2018). Locally coordinated synaptic plasticity of visual cortex neurons *in vivo*. *Science* 360, 1349–1354. doi: 10.1126/science.aao0862

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., et al. (2017). PathNet: evolution channels gradient descent in super neural networks. *arXiv:1701.08734 [Preprint]*. Available online at: https://arxiv.org/abs/1701.08734

Frmaux, N., and Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circ.* 9:85. doi: 10.3389/fncir.2015.00085

Grossberg, S. (1987). Competitive learning: from interactive activation to adaptive resonance. *Cogn. Sci.* 11, 23–63. doi: 10.1111/j.1551-6708.1987.tb00862.x

Han, B., Ankit, A., Sengupta, A., and Roy, K. (2018). Cross-layer design exploration for energy-quality tradeoffs in spiking and non-spiking deep artificial neural networks. *IEEE Trans. Multi Scale Comput. Syst.* 4, 613–623. doi: 10.1109/TMSCS.2017.2737625

Hohn, N., and Burkitt, A. (2001). Shot noise in the leaky integrate-and-fire neuron. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* 63:031902. doi: 10.1103/PhysRevE.63.031902

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U.S.A.* 114, 3521–3526. doi: 10.1073/pnas.1611835114

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lee, M. J., and DiCarlo, J. J. (2019). Comparing novel object learning in humans, models, and monkeys. *J. Vis.* 19:114b. doi: 10.1167/19.10.114b

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. (2017). "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in Neural Information Processing Systems 30*, eds I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Long Beach, CA: Curran Associates, Inc.), 4652–4662.

Li, Z., and Hoiem, D. (2018). Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 2935–2947. doi: 10.1109/TPAMI.2017.2773081

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *J. Math. Biol.* 15, 267–273. doi: 10.1007/BF00275687

Op de Beeck, H. P., Deutsch, J. A., Vanduffel, W., Kanwisher, N. G., and DiCarlo, J. J. (2007). A stable topography of selectivity for unfamiliar shape classes in monkey inferior temporal cortex. *Cereb. Cortex* 18, 1676–1694. doi: 10.1093/cercor/bhm196

Panda, P., Allred, J. M., Ramanathan, S., and Roy, K. (2018). Asp: learning to forget with adaptive synaptic plasticity in spiking neural networks. *IEEE J. Emerg. Select. Top. Circ. Syst.* 8, 51–64. doi: 10.1109/JETCAS.2017.2769684

Rios, A., and Itti, L. (2018). Closed-loop memory GAN for continual learning. *arXiv:1811.01146 [Preprint]*. Available online at: https://arxiv.org/abs/1811.01146

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., et al. (2016). Progressive neural networks. *arXiv:1606.04671 [Preprint]*. Available online at: https://arxiv.org/abs/1606.04671

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J. (2013). "Compete to compute," in *Advances in Neural Information Processing Systems 26*, eds C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Lake Tahoe, NV: Curran Associates, Inc.), 2310–2318.

Wang, J., Belatreche, A., Maguire, L., and McGinnity, T. M. (2014). An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing* 144, 526–536. doi: 10.1016/j.neucom.2014.04.017

Wang, J., Belatreche, A., Maguire, L., and McGinnity, T. M. (2015). "Dynamically evolving spiking neural network for pattern recognition," in *2015 International Joint Conference on Neural Networks (IJCNN)* (Killarney), 1–8. doi: 10.1109/IJCNN.2015.7280649

Wang, J., Belatreche, A., Maguire, L. P., and McGinnity, T. M. (2017). Spiketemp: an enhanced rank-order-based learning approach for spiking neural networks with adaptive structure. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 30–43. doi: 10.1109/TNNLS.2015.2501322

Wysoski, S. G., Benuskova, L., and Kasabov, N. (2006). "On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition," in *Proceedings of the 16th International Conference on Artificial Neural Networks - Volume Part I, ICANN'06* (Berlin; Heidelberg. Springer-Verlag), 61–70.

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.

frontiers in Neuroscience

ORIGINAL RESEARCH
published: 28 February 2020
doi: 10.3389/fnins.2020.00119

Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures

Chankyu Lee[*†], *Syed Shakib Sarwar*[*†], *Priyadarshini Panda, Gopalakrishnan Srinivasan and Kaushik Roy*

Nanoelectronics Research Laboratory, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, United States

OPEN ACCESS

Edited by:
Hesham Mostafa,
University of California, San Diego, United States

Reviewed by:
Bodo Rückauer,
ETH Zürich, Switzerland
Eric Hunsberger,
University of Waterloo, Canada
David Kappel,
Dresden University of Technology, Germany

Correspondence:
Chankyu Lee
lee2216@purdue.edu
Syed Shakib Sarwar
sarwar@purdue.edu

[†]*These authors have contributed equally to this work*

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *11 September 2019*
Accepted: *30 January 2020*
Published: *28 February 2020*

Citation:
Lee C, Sarwar SS, Panda P, Srinivasan G and Roy K (2020) Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures. Front. Neurosci. 14:119. doi: 10.3389/fnins.2020.00119

Spiking Neural Networks (SNNs) have recently emerged as a prominent neural computing paradigm. However, the typical shallow SNN architectures have limited capacity for expressing complex representations while training deep SNNs using input spikes has not been successful so far. Diverse methods have been proposed to get around this issue such as converting off-the-shelf trained deep Artificial Neural Networks (ANNs) to SNNs. However, the ANN-SNN conversion scheme fails to capture the temporal dynamics of a spiking system. On the other hand, it is still a difficult problem to directly train deep SNNs using input spike events due to the discontinuous, non-differentiable nature of the spike generation function. To overcome this problem, we propose an approximate derivative method that accounts for the leaky behavior of LIF neurons. This method enables training deep convolutional SNNs directly (with input spike events) using spike-based backpropagation. Our experiments show the effectiveness of the proposed spike-based learning on deep networks (VGG and Residual architectures) by achieving the best classification accuracies in MNIST, SVHN, and CIFAR-10 datasets compared to other SNNs trained with a spike-based learning. Moreover, we analyze sparse event-based computations to demonstrate the efficacy of the proposed SNN training method for inference operation in the spiking domain.

Keywords: spiking neural network, convolutional neural network, spike-based learning rule, gradient descent backpropagation, leaky integrate and fire neuron

1. INTRODUCTION

Over the last few years, deep learning has made tremendous progress and has become a prevalent tool for performing various cognitive tasks such as object detection, speech recognition, and reasoning. Various deep learning techniques (LeCun et al., 1998; Srivastava et al., 2014; Ioffe and Szegedy, 2015) enable the effective optimization of deep ANNs by constructing multiple levels of feature hierarchies and show remarkable results, which occasionally outperform human-level performance (Krizhevsky et al., 2012; He et al., 2016; Silver et al., 2016). To harness the deep learning capabilities in ubiquitous environments, it is necessary to deploy deep learning not only on large-scale computers, but also on edge devices (e.g., phone, tablet, smartwatch, robot, etc.). However, the ever-growing complexity of deep neural networks together with the explosion in the amount of data to be processed, place significant energy demands on current computing platforms.

Spiking Neural Network (SNN) is one of the leading candidates for overcoming the constraints of neural computing and to efficiently harness the machine learning algorithm in real-life (or mobile) applications. The concepts of SNN, which is often regarded as the 3rd generation neural network (Maass, 1997), are inspired by the biological neuronal mechanisms (Hodgkin and Huxley, 1952; Dayan and Abbott, 2001; Izhikevich, 2003; Brette and Gerstner, 2005) that can efficiently process discrete spatio-temporal events (spikes). The Leaky Integrate and Fire (LIF) neuron is the simple first-order phenomenological spiking neuron model, which can be characterized by the internal state, called membrane potential. The membrane potential integrates the inputs over time and generates an output spike whenever it overcomes the neuronal firing threshold. Recently, specialized hardwares (Merolla et al., 2014; Ankit et al., 2017; Davies et al., 2018) have been developed to exploit this event-based, asynchronous signaling/processing scheme. They are promising for achieving ultra-low power intelligent processing of streaming spatiotemporal data, and especially in deep hierarchical networks, as it has been observed in SNN models that the number of spikes, and thus the amount of computation, decreases significantly at deeper layers (Rueckauer et al., 2017; Sengupta et al., 2019).

We can divide SNNs into two broad classes: (a) converted SNNs and (b) SNNs derived by direct spike-based training. The former one is SNNs converted from the trained ANN for the efficient event-based inference (ANN-SNN conversion) (Cao et al., 2015; Hunsberger and Eliasmith, 2015; Diehl et al., 2016; Rueckauer et al., 2017; Sengupta et al., 2019). The main advantage is that it uses state-of-the-art (SOTA), optimization-based, ANN training methods and therefore achieves SOTA classification performance. For instance, the specialized SNN hardwares [such as SpiNNaker (Furber et al., 2013), IBM TrueNorth (Merolla et al., 2014) have exhibited greatly improved power efficiency as well as the state-of-the-art performance for the inference. The signals used in such training are real-valued and are naturally viewable as representing spike rate (frequency). The problem is that reliably estimating frequencies requires non-trivial passage of time. On the other hand, SNNs derived by direct spike-based training also involves some hurdles. Direct spike-based training methods can be divided into two classes: (i) non-optimization-based, mostly unsupervised, approaches involving only signals local to the synapse, e.g., the times of pre- and post-synaptic spikes, as in Spike-Timing-Dependent-Plasticity (STDP); and (ii) optimization-based, mostly supervised, approaches involving a global objective, e.g., loss, function. STDP-trained two-layer network (consisting of 6,400 output neurons) has been shown to achieve 95% classification accuracy on MNIST dataset (Diehl and Cook, 2015). However, a shallow network structure limits the expressive power (Brader et al., 2007; Zhao et al., 2015; Srinivasan et al., 2018a,b) and may not scale well to larger problem sizes. While efficient feature extraction has been demonstrated using layer-wise STDP learning in deep convolutional SNNs (Kheradpisheh et al., 2016; Lee et al., 2019b), ANN models trained with standard backpropagation (BP) (Rumelhart et al., 1985) still achieve significantly better classification performance. These considerations have inspired the search for spike-based versions of BP, which requires finding a differentiable surrogate of the spiking unit's activation function. Bohte et al. (2002) and Lee et al. (2016) defined such a surrogate in terms of a unit's membrane potential. Besides, Panda and Roy (2016) applies BP-based supervised training for the classifier after layer-by-layer autoencoder-based training of the feature extractor. By combining layer-wise STDP-based unsupervised and supervised spike-based BP, Lee et al. (2018) showed improved robustness, generalization ability, and faster convergence. In this paper, we take these prior works forward to effectively train very deep SNNs using end-to-end spike-based BP learning.

The main contributions of our work are as follows. First, we develop a spike-based supervised gradient descent BP algorithm that employs an approximate (pseudo) derivative for LIF neuronal function. In addition, we leverage the key idea of the successful deep ANN models such as LeNet5 (LeCun et al., 1998), VGG (Simonyan and Zisserman, 2014), and ResNet (He et al., 2016) for efficiently constructing deep convolutional SNN architectures. We also adapt the dropout (Srivastava et al., 2014) technique to better regularize deep SNN training. Next, we demonstrate the effectiveness of our methodology for visual recognition tasks on standard character and object datasets (MNIST, SVHN, CIFAR-10) and a neuromorphic dataset (N-MNIST). To the best of our knowledge, this work achieves the best classification accuracy in MNIST, SVHN, and CIFAR-10 datasets among other spike-based learning methodologies. Last, we expand our efforts to quantify and analyze the advantages of a spike-based BP algorithm compared to ANN-SNN conversion techniques in terms of inference time and energy consumption.

The rest of the paper is organized as follows. In section 2.1, we provide the background on fundamental components and architectures of deep convolutional SNNs. In section 2.2.1, we detail the spike-based gradient descent BP learning algorithm. In section 2.2.2, we describe our spiking version of the dropout technique. In section 3.1–3.2, we describe the experiments and report the simulation results, which validate the efficacy of spike-based BP training for MNIST, SVHN, CIFAR-10, and N-MNIST datasets. In section 4.1, we discuss the proposed algorithm in comparison to relevant works. In section 4.2–4.4, we analyze the spike activity, inference speedup and complexity reduction of directly trained SNNs and ANN-SNN converted networks. Finally, we summarize and conclude the paper in section 5.

2. MATERIALS AND METHODS

2.1. The Components and Architecture of Spiking Neural Network

2.1.1. Spiking Neural Network Components

Leaky-Integrate-and-Fire (LIF) neurons (Dayan and Abbott, 2001) and plastic synapses are fundamental and biologically plausible computational elements for emulating the dynamics of SNNs. The sub-threshold dynamics of a LIF spiking neuron can be formulated as

$$\tau_m \frac{dV_{mem}}{dt} = -V_{mem} + I(t) \tag{1}$$

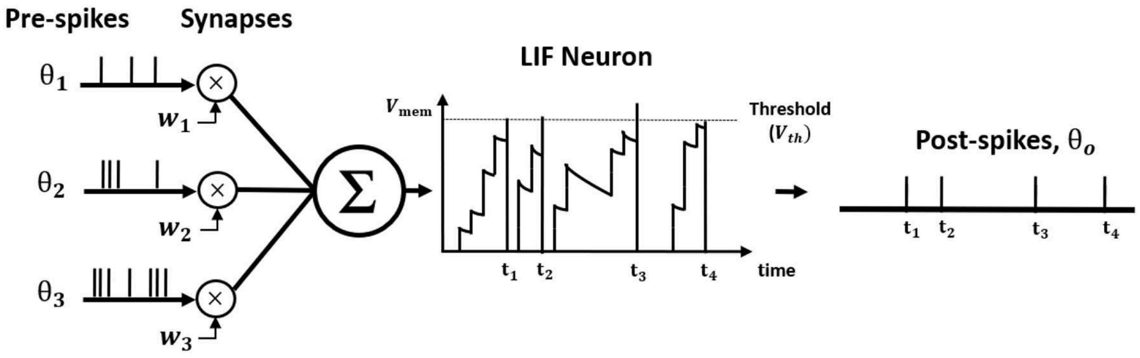


FIGURE 1 | The illustration of Leaky Integrate and Fire (LIF) neuron dynamics. The pre-spikes are modulated by the synaptic weight to be integrated as the current influx in the membrane potential that decays exponentially. Whenever the membrane potential crosses the firing threshold, the post-neuron fires a post-spike and resets the membrane potential.

where V_{mem} is the post-neuronal membrane potential and τ_m is the time constant for membrane potential decay. The input current, $I(t)$, is defined as the weighted summation of pre-spikes at each time step as given below.

$$I(t) = \sum_{i=1}^{n^l} (w_i \sum_k \theta_i(t - t_k)) \tag{2}$$

where n^l indicates the number of pre-synaptic weights, w_i is the synaptic weight connecting ith pre-neuron to post-neuron. $\theta_i(t - t_k)$ is a spike event from ith pre-neuron at time t_k, which can be formulated as a Kronecker delta function as follows,

$$\theta(t - t_k) = \begin{cases} 1, & \text{if } t = t_k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where t_k is the time instant that kth spike occurred. **Figure 1** illustrates LIF neuronal dynamics. The impact of each pre-spike, $\theta_i(t-t_k)$, is modulated by the corresponding synaptic weight (w_i) to generate a current influx to the post-neuron. Note, the units do not have bias term. The input current is integrated into the post-neuronal membrane potential (V_{mem}) that leaks exponentially over time with time constant (τ_m). When the membrane potential exceeds a threshold (V_{th}), the neuron generates a spike and resets its membrane potential to initial value. The **Table 1** lists the annotations used in Equations (1–27).

2.1.2. Deep Convolutional Spiking Neural Network

2.1.2.1. Building blocks

In this work, we develop a training methodology for convolutional SNN models that consist of an input layer followed by intermediate hidden layers and a final classification layer. In the input layer, the pixel images are encoded as Poisson-distributed spike trains where the probability of spike generation is proportional to the pixel intensity. The hidden layers consist of multiple convolutional (C) and spatial-pooling (P) layers, which are often arranged in an alternating manner. These convolutional (C) and spatial-pooling (P) layers represent the intermediate stages of feature extractor. The spikes from the feature extractor are combined to generate a one-dimensional vector input for the fully-connected (FC) layers to produce the final classification. The convolutional and fully-connected layers contain trainable parameters (i.e., synaptic weights) while the spatial-pooling layers are fixed *a priori*. Through the training procedure, weight kernels in the convolutional layers can encode the feature representations of the input patterns at multiple hierarchical levels. **Figure 2A** shows the simplified operational example of a convolutional layer consisting of LIF neurons over three time steps (assuming 2-D input and 2-D weight kernel). On each time step, each neuron convolves its input spikes with the weight kernel to compute its input current, which is integrated into its membrane potential, V_{mem}. If $V_{mem} > V_{th}$, the neuron spikes and its V_{mem} is set to 0. Otherwise, V_{mem} is considered as residue in the next time step while leaking in the current time step. **Figure 2B** shows the simplified operation of a pooling layer, which reduces the dimensionality from the previous convolutional layer while retaining spatial (topological) information.

TABLE 1 | List of notations.

Notations	Meaning
θ	Spike event
x	The sum of pre-spike events over time
w	Synaptic weight
V_{mem}	Membrane potential
V_{th}	Neuronal firing threshold
I	Input current at each time step
net	Total current influx over time
a	Activation of spiking neuron
E	Loss function
δ	Error gradient

The two major operations used for pooling are max and average. Both have been used for SNNs, e.g., max-pooling (Rueckauer et al., 2017) and average-pooling (Cao et al., 2015;

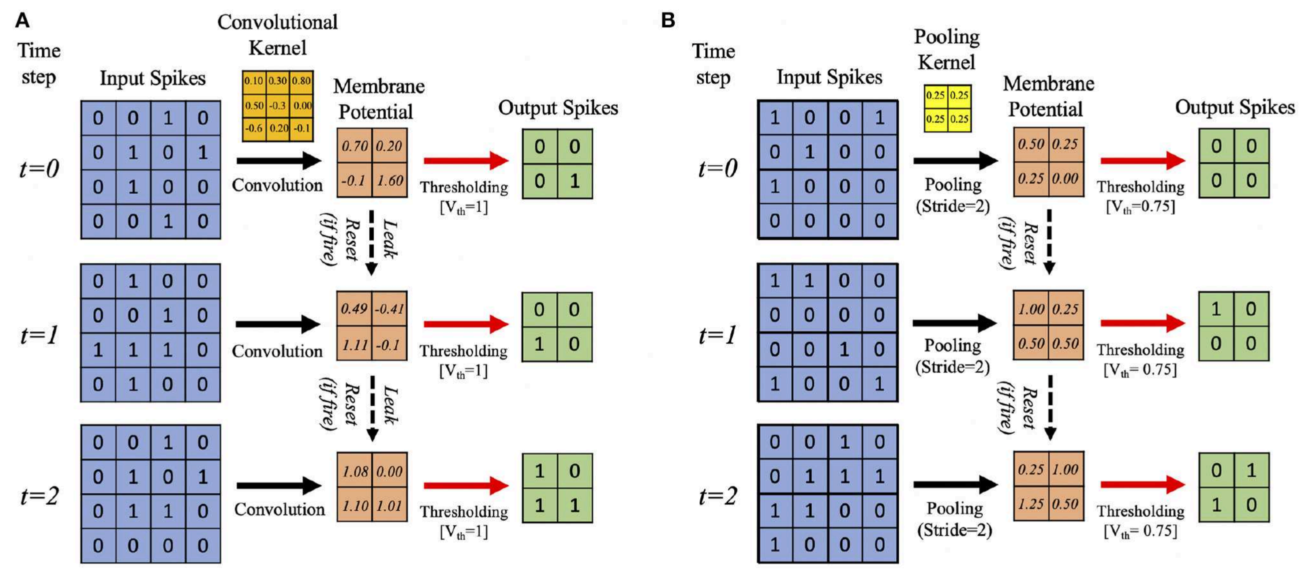


FIGURE 2 | Illustration of the simplified operational example of **(A)** convolutional, **(B)** spatial-pooling layers (assuming 2-D input and 2-D weight kernel) over three time steps. At each time step, the input spikes are convolved with the weight kernel to generate the current influx, which is accumulated in the post-neuron's membrane potential, V_{mem}. Whenever the membrane potential exceeds the firing threshold (V_{th}), the post-neuron in the output feature map spikes and V_{mem} resets. Otherwise, V_{mem} is considered as residue in the next time step while leaking in the current time step. For spatial-pooling, the kernel weights are fixed, and there is no membrane potential leak.

Diehl et al., 2015). We use average-pooling due to its simplicity. In the case of SNNs, an additional thresholding is used after averaging to generate output spikes. For instance, a fixed 2×2 kernel (each having a weight of 0.25) strides through a convolutional feature map without overlapping and fires an output spike at the corresponding location in the pooled feature map only if the sum of the weighted spikes of the 4 inputs within the kernel window exceeds a designated firing threshold (set to 0.75 in this work). Otherwise, the membrane potential remains as a residue in the next time step. **Figure 2B** shows an example spatial-pooling operation over three time steps (assuming 2-D input and 2-D weight kernel). The average-pooling threshold need to be carefully set so that spike propagation is not disrupted due to the pooling. If the threshold is too low, there will be too many spikes, which can cause loss of spatial location of the feature that was extracted from the previous layer. If the threshold is too high, there will not be enough spike propagation to the deeper layers.

2.1.2.2. Deep convolutional SNN architecture: VGG and residual SNNs

Deep networks are essential for recognizing intricate input patterns so that they can effectively learn hierarchical representations. To that effect, we investigate popular deep neural network architectures such as VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016) in order to build deep SNN architectures. VGG (Simonyan and Zisserman, 2014) was one of the first neural networks, which used the idea of using small (3×3) convolutional kernels uniformly throughout the network. Using small kernels enables effective stacking of convolutional layers while minimizing the number of parameters in deep networks. In this work, we build deep convolutional SNNs (containing more than 5 trainable layers) using "Spiking VGG Block's," which contain stacks of convolutional layers using small (3×3) kernels. **Figure 3A** shows a "Spiking VGG block" containing two stacked convolutional layers, each followed by a LIF neuronal layer. The convolutional layer box contains the synaptic connectivity, and the LIF neuronal box contains the activation units. Next, ResNet (He et al., 2016) introduced the skip connections throughout the network that had considerable successes in enabling successful training of significantly deeper networks. In particular, ResNet addresses the degradation (of training accuracy) problem (He et al., 2016) that occurs while increasing the number of layers in the standard feedforward neural network. We employ the concept of skip connection to construct deep residual SNNs with 7–11 trainable layers. **Figure 3B** shows a "Spiking Residual Block" containing non-residual and residual paths. The non-residual path consists of two convolutional layers with an intermediate LIF neuronal layer. The residual path (skip connection) is composed of the identity mapping when the number of input and output feature maps are the same, and 1×1 convolutional kernels when the number of input and output feature maps differ. The outputs of both the non-residual and residual paths are integrated to the membrane potential in the last LIF neuronal activation layer (LIF Neuron 2 in **Figure 3B**) to generate output spikes from the "Spiking Residual Block." Within the feature extractor, a "Spiking VGG Block" or "Spiking Residual Block" is often followed by an average-pooling layer. Note, in some "Spiking Residual Blocks," the last convolutional and residual connections employ convolution with a stride of 2 to incorporate the functionality of the spatial-pooling layers. At the end of the feature extractor,

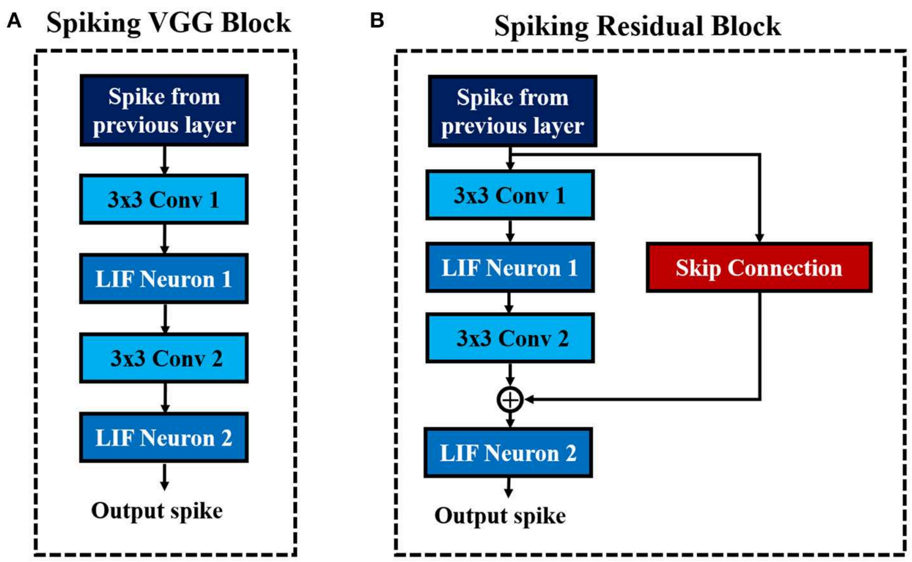


FIGURE 3 | The basic building blocks of the described convolutional SNN architectures. **(A)** Spiking VGG Block. **(B)** Spiking ResNet Block.

extracted features from the last average-pooling layer are fed to a fully-connected layer as a 1-D vector input for inference.

2.2. Supervised Training of Deep Spiking Neural Network

2.2.1. Spike-Based Gradient Descent Backpropagation Algorithm

The spike-based BP algorithm in SNN is adapted from standard BP (Rumelhart et al., 1985) in the ANN domain. In standard BP, the network parameters are iteratively updated in a direction to minimize the difference between the final outputs of the network and target labels. The standard BP algorithm achieves this goal by back-propagating the output error through the hidden layers using gradient descent. However, the major difference between ANNs and SNNs is the dynamics of neuronal output. An artificial neuron (such as *sigmoid*, *tanh*, or *ReLU*) communicates via continuous values whereas a spiking neuron generates binary spike outputs over time. In SNNs, spatiotemporal spike trains are fed to the network as inputs. Accordingly, the outputs of spiking neuron are spike events, which are also discrete over time. Hence, the standard BP algorithm is incompatible with training SNNs, as it can not back-propagate the gradient through a non-differentiable spike generation function. In this work, we formulate an approximate derivative for LIF neuron activation, making gradient descent possible. We derive a spike-based BP algorithm that is capable of learning spatiotemporal patterns in spike-trains. The spike-based BP can be divided into three phases, forward propagation, backward propagation and weight update, which we describe in the following sections.

2.2.1.1. Forward propagation

In forward propagation, spike trains representing input patterns are presented to the network for estimating the network outputs. To generate the spike inputs, the input pixel values are converted to Poisson-distributed spike trains and delivered to the network. The input spikes are multiplied with synaptic weights to produce an input current that accumulates in the membrane potential of post neurons as in Equation (1). Whenever its membrane potential exceeds a neuronal firing threshold, the post-neuron generates an output spike and resets. Otherwise, the membrane potential decays exponentially over time. The neurons of every layer (excluding output layer) carry out this process successively based on the weighted spikes received from the preceding layer. Over time, the total weighted summation of the pre-spike trains (i.e., *net*) is described as follows,

$$net_j^l(t) = \sum_{i=1}^{n^{l-1}} (w_{ij}^{l-1} x_i^{l-1}(t)), \; where \; x_i^{l-1}(t) = \sum_t \sum_k \theta_i^{l-1}(t - t_k) \tag{4}$$

where $net_j^l(t)$ represents the total current influx integrated to the membrane potential of jth post-neuron in layer l over the time t, n^{l-1} is the number of pre-neurons in layer l-1 and $x_i^{l-1}(t)$ denotes the sum of spike train ($t_k \leq t$) from ith pre-neuron over time t. The sum of post-spike trains ($t_k \leq t$) is represented by $a_j^l(t)$ for the jth post-neuron.

$$a_j^l(t) = \sum_t \sum_k \theta_j^l(t - t_k) \tag{5}$$

Clearly, the sum of post-spike train ($a^l(t)$) is equivalent to the sum of pre-spike train ($x^l(t)$) for the next layer. On the other hand, the neuronal firing threshold of the final classification layer is set to a very high value so that final output neurons do not spike. In the final layer, the weighted pre-spikes are accumulated in the membrane potential while decaying over time. At the last time step, the accumulated membrane potential is divided by the number of total time steps (T) in order to quantify the output distribution (*output*) as presented by Equation (6).

$$output = \frac{V^L_{mem}(T)}{number\ of\ timesteps} \tag{6}$$

2.2.1.2. Backward propagation and weight update

Next, we describe the backward propagation for the proposed spike-based backpropagation algorithm. After the forward propagation, the loss function is measured as a difference between target labels and outputs predicted by the network. Then, the gradients of the loss function are estimated at the final layer. The gradients are propagated backward all the way down to the input layer through the hidden layers using recursive chain rule, as formulated in Equation (7). The following Equations (7–27) and **Figure 4** describe the detailed steps for obtaining the partial derivatives of (final) output error with respect to weight parameters.

The prediction error of each output neuron is evaluated by comparing the output distribution (*output*) with the desired target label (*label*) of the presented input spike trains, as shown in Equation (8). The corresponding loss function (E in Equation, 9) is defined as the sum of squared (final prediction) error over all output neurons. To calculate the $\frac{\partial E}{\partial a_{LIF}}$ and $\frac{\partial a_{LIF}}{\partial net}$ terms in Equation (7), we need a defined activation function and a method to differentiate the activation function of a LIF neuron.

$$\frac{\partial E}{\partial w^l} = \frac{\partial E}{\partial a_{LIF}} \frac{\partial a_{LIF}}{\partial net} \frac{\partial net}{\partial w^l} \tag{7}$$

$$Final\ output\ error,\ e_j = output_j - label_j \tag{8}$$

$$Loss\ function,\ E = \frac{1}{2}\sum_{j=1}^{n^L} {e_j}^2 \tag{9}$$

In SNN, the "activation function" indicates the relationship between the weighted summation of pre-spike inputs and post-neuronal outputs over time. In forward propagation, we have different types of neuronal activation for the final layer and hidden layers. Hence, the estimation of neuronal activations and their derivatives are different for the final layer and hidden layers. For the final layer, the value of *output* in Equation (6) is used as the neuronal activation (a_{LIF}) while considering the discontinuities at spike time instant as noise. Hence, $\frac{\partial E}{\partial output}$ is equal to the final output error, as calculated in Equation (10).

$$\frac{\partial E}{\partial output} = \frac{\partial}{\partial output}\frac{1}{2}(output - label)^2 = output - label = e \tag{10}$$

During back-propagating phase, we consider the leak statistics of membrane potential in the final layer neurons as noise. This allows us to approximate the accumulated membrane potential value for a given neuron as equivalent to the total input current (i.e., *net*) received by the neuron over the forward time duration (T) ($V^L_{mem,j}(T) \approx \sum_{i=1}^{n^{L-1}}(w_{ij}x_i(T)) = net^L_j(T)$). Therefore, the derivative of post-neuronal activation with respect to *net* for final layer ($\frac{\partial output}{\partial net} \equiv \frac{\partial V^L_{mem}(T)/T}{\partial net} = \frac{\partial net^L(T)/T}{\partial net} = \frac{1}{T}$) is calculated as $\frac{1}{T}$ for the final layer.

For the hidden layers, we have post-spike trains as the neuronal outputs. The spike generation function is non-differentiable since it creates a discontinuity (because of step jump) at the time instance of firing. Hence, we introduce a pseudo derivative method for LIF neuronal activation ($a'_{LIF}(net)$) for the hidden layers, for back-propagating the output error via the chain rule. The purpose of deriving $a'_{LIF}(net)$ is to approximately estimate the $\frac{\partial a_{LIF}}{\partial net}$ term in Equation (7) for the hidden layers only. To obtain this pseudo derivative of LIF neuronal activation with respect to total input current (i.e., *net*), we make the following approximations. We first estimate the derivative of an "Integrate and Fire" (IF) neuron's activation. Next, with the derivative of IF neuron's activation, we estimate a leak correctional term to compensate for the leaky effect of membrane potential in LIF activation. Finally, we obtain an approximate derivative for LIF neuronal activation as a combination of two estimations (i.e., derivative for IF neuron and approximated leak compensation derivative). If a hidden neuron does not fire any spike, the derivative of corresponding neuronal activation is set to zero.

The spike generation function of IF neuron is a hard threshold function that generates the output signal as either +1 or 0. The IF neuron fires a post-spike whenever the input currents accumulated in membrane potential exceed the firing threshold (note, in case of IF neuron, there is no leak in the membrane potential). Hence, the membrane potential of a post-neuron at time instant *t* can be written as,

$$V_{mem}(t) \approx \sum_{i=1}^{n}(w_i x_i(t)) - V_{th}a_{IF}(t) \tag{11}$$

where n denotes the number of pre-neurons, $x_i(t)$ is the sum of spike events from ith pre-neuron over time t (defined in Equation, 4) and $a_{IF}(t)$ represents the sum of post-spike trains over time t (defined in Equation 5). In Equation (11), $\sum_{i=1}^{n}(w_i x_i(t))$ accounts for the integration behavior and $V_{th}a_{IF}(t)$ accounts for the fire/reset behavior of the membrane potential dynamics. If we assume V_{mem} as zero (using small signal approximation), the activation of IF neuron ($a_{IF}(t)$) can be formulated as the Equation (12). Then, by differentiating it with respect to *net* (in Equation, 13), the derivative of IF neuronal activation can be approximated as a linear function with slope of $\frac{1}{V_{th}}$ as the straight-through estimation (Bengio et al., 2013).

$$a_{IF}(t) \approx \frac{1}{V_{th}}\sum_{i=1}^{n}(w_i x_i(t)) = \frac{1}{V_{th}}net(t) \tag{12}$$

$$\frac{\partial a_{IF}}{\partial net} \approx \frac{1}{V_{th}}1 = \frac{1}{V_{th}} \tag{13}$$

The spike generation function of both the IF and LIF neuron models are the same, namely the hard threshold function.

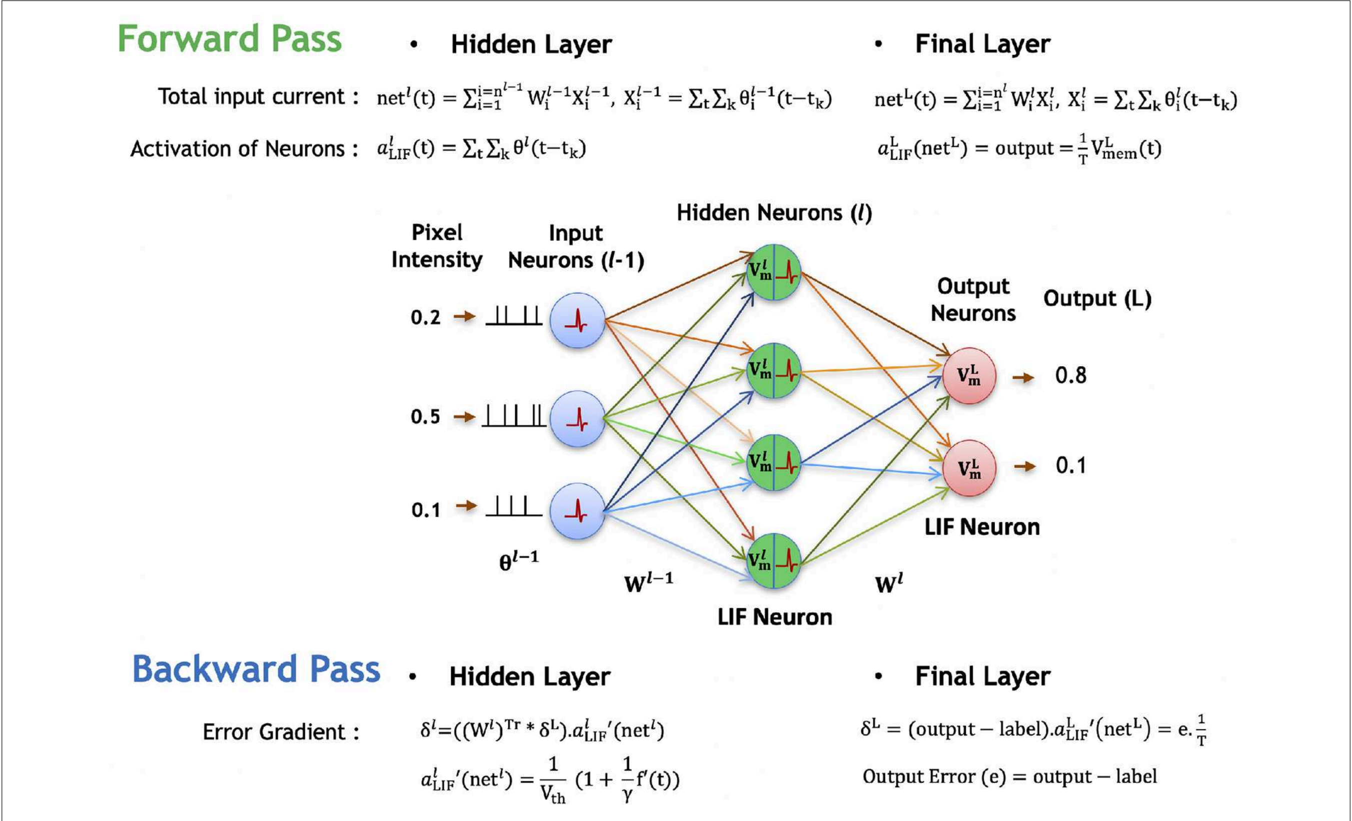

Forward Pass
• Hidden Layer
• Final Layer
Total input current : $\mathrm{net}^l(t) = \sum_{i=1}^{i=n^{l-1}} W_i^{l-1} X_i^{l-1},\ X_i^{l-1} = \sum_t \sum_k \theta_i^{l-1}(t-t_k)$
$\mathrm{net}^L(t) = \sum_{i=1}^{i=n^l} W_i^l X_i^l,\ X_i^l = \sum_t \sum_k \theta_i^l(t-t_k)$
Activation of Neurons : $a_{LIF}^l(t) = \sum_t \sum_k \theta^l(t-t_k)$
$a_{LIF}^L(\mathrm{net}^L) = \mathrm{output} = \frac{1}{T} V_{mem}^L(t)$
Pixel Intensity
Input Neurons (l-1)
Hidden Neurons (l)
Output Neurons
Output (L)
0.2
0.5
0.1
0.8
0.1
θ^{l-1}
W^{l-1}
W^l
LIF Neuron
LIF Neuron
Backward Pass
• Hidden Layer
• Final Layer
Error Gradient : $\delta^l = ((W^l)^{Tr} * \delta^L).a_{LIF}^l{}'(\mathrm{net}^l)$
$a_{LIF}^l{}'(\mathrm{net}^l) = \frac{1}{V_{th}}(1 + \frac{1}{\gamma} f'(t))$
$\delta^L = (\mathrm{output} - \mathrm{label}).a_{LIF}^L{}'(\mathrm{net}^L) = e.\frac{1}{T}$
Output Error (e) = output − label

FIGURE 4 | Illustration of the forward and backward propagation phase of the proposed spike-based BP algorithm in a multi-layer SNN comprised of LIF neurons. In the forward phase, the LIF neurons (in all layers) accumulate the weighted sum of the pre-spikes in the membrane potential, which decays exponentially over time. In addition, the LIF neurons in hidden layers generate post-spikes if the membrane potential exceeds a threshold and reset the membrane potential. However, the LIF neurons in the final layer, do not generate any spike, but rather accumulate the weighted sum of pre-spikes till the last time step to quantify the final outputs. Then, the final errors are evaluated by comparing the final outputs to the label data. In the backward phase, the final errors are propagated backward through the hidden layers using the chain rule to obtain the partial derivatives of final error with respect to weights. Finally, the synaptic weights are modified in a direction to reduce the final errors.

However, the effective neuronal thresholds are considered to be different for the two cases, as shown in **Figures 5A,B**. In the LIF neuron model, due to the leaky effect in the membrane potential, larger input current (as compared to IF neuron) needs to be accumulated in order to cross the neuronal threshold and generate a post-spike. Hence, the effective neuronal threshold becomes $V_{th} + \epsilon$ where ϵ is a positive value that reflects the leaky effect of membrane potential dynamics. Now, the derivative of LIF neuronal activation ($\frac{\partial a_{LIF}}{\partial net}$) can be approximated as a hard threshold function [similar to IF and Equation (13)] and written as $\frac{1}{V_{th}+\epsilon}$. Clearly, the output of a LIF neuron depends on the firing threshold and leaky characteristics (embodied in ϵ) of the membrane potential whereas the output of an IF neuron depends only on the firing threshold. Next, we explain the detailed steps to estimate the ϵ and in turn calculate the derivative of LIF neuronal activation ($\frac{\partial a_{LIF}}{\partial net}$).

To compute ϵ, the ratio (β) between the total membrane potential ($V_{mem}^{total}(t)$) of IF and LIF neurons is estimated at the end of forward propagation time (T) as shown in **Figure 5C**. Here, $V_{mem}^{total}(t)$ represents the hypothetical total membrane potential with accumulated input current without reset mechanism until time step (t). Suppose both the IF and LIF neurons received the same amount of total input current (i.e., $net(T)$), the total membrane potential of LIF neuron is expected to be lower than the total membrane potential of IF neuron ($V_{mem}^{total,LIF}(T) : V_{mem}^{total,IF}(T) = 1 : \beta$ where $\beta > 1$). Hence, by comparing the total membrane potential values of IF and LIF neurons in **Figure 5C**, the relation of ϵ and β can be obtained as follows,

$$V_{th} + \epsilon = \beta V_{th} \tag{14}$$

where $V_{th} + \epsilon$ represents the total membrane potential of IF neuron (point A in **Figure 5C**) and V_{th} indicates the total membrane potential of LIF neuron (point B in **Figure 5C**) when both neurons received the same amount of *net* inputs. Based on this assumption, we now estimate the ratio (β) by using the relation of the spike output evolution ($\frac{\partial a(t)}{\partial t}$) and the total membrane potential evolution ($\frac{\partial V_{mem}^{total}(t)}{\partial t}$) over time as described in Equations (16–20). As mentioned previously, the total input current (i.e., $net(t)$) and total membrane potential ($V_{mem}^{total}(t)$) are estimated similar to that of IF neuron (because

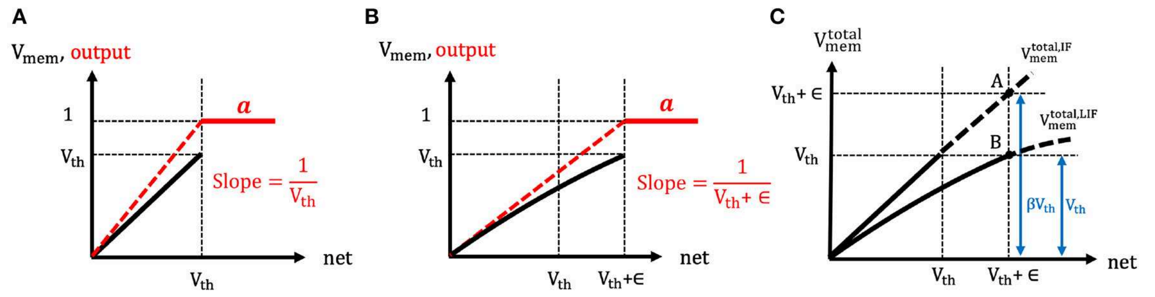


FIGURE 5 | (A,B) The illustration of the spike generation function of **(A)** IF and **(B)** LIF neuron models, respectively. The x-axis represents the total summation of input currents over time, and y-axis indicates the membrane potential (black) and output (red). The IF neuron generates a post-spike when the input currents accumulated in membrane potential overcome the firing threshold (because of no leaky effect in the membrane potential). However, LIF neuron needs more input currents to cross the firing threshold (because of leaky effect in the membrane potential). Hence, the effective threshold of LIF neurons is considered to be larger compared to the case of IF neurons. **(C)** The illustration of the estimation of the ratio (β) between the total membrane potential (V_{mem}^{total}) of LIF and IF neurons. If the LIF and IF neuron received the same amount of total input current, the ratio of the total membrane potential of LIF and IF neuron would be estimated as 1:β where β is greater than 1.

of no leaky effect) so that Equation (15) can be derived from Equation (12). By differentiating Equation (15) with respect to time, we get the relation of the spike output evolution ($\frac{\partial a_{IF}(t)}{\partial t}$) and the membrane potential evolution ($\frac{\partial V_{mem}^{total,IF}(t)}{\partial t}$) over time for IF neuron as described in Equation (16).

$$a_{IF}(t) \approx \frac{1}{V_{th}} net(t) \approx \frac{1}{V_{th}} V_{mem}^{total,IF}(t) \tag{15}$$

$$\frac{\partial a_{IF}(t)}{\partial t} \approx \frac{1}{V_{th}} \frac{\partial V_{mem}^{total,IF}(t)}{\partial t} \tag{16}$$

Hence, in IF neuron case, the evolution of membrane potential over time ($\frac{\partial V_{mem}^{total,IF}(t)}{\partial t}$) can be represented by the multiplication of firing threshold (V_{th}) and the spike output evolution ($\frac{\partial a_{IF}(t)}{\partial t}$) in Equation (17). Note, the evolution of membrane potential over time ($\frac{\partial V_{mem}^{total,IF}(t)}{\partial t}$) indicates the integration component due to the average input current over time. We consider $a_{IF}(t)$ as homogeneous spike trains where spike firing rates are constant, so that the $\frac{\partial a_{IF}(t)}{\partial t}$ can be replaced with the post-neuronal firing rate ($rate(t)$). The homogeneous post-neuronal firing rate, $rate(t)$, can be represented by $\frac{a(t)}{t}$ where $a(t)$ is the number of post-spikes and t means the given forward time window. In LIF neuron case, however, the evolution of membrane potential ($\frac{\partial V_{mem}^{total,LIF}(t)}{\partial t}$) can be expressed as the combination of average input current (integration component) and leaky (exponential decay) effect as shown in Equation (18). To measure the leaky effect in Equation (18), we estimate the low-pass filtered output spikes ($t_k \leq t$) that leak over time using the function $V_{th}f(t)$ (depicted in Equation, 19), and differentiate it with respect to time at $t \rightarrow t_k^+$ (from the right-sided limit). The $V_{th}f(t)$, as a post-synaptic potential, contains the total membrane potential history over time. The time constant (τ_m) in Equation (19) determines the decay rate of post-synaptic potential. Essentially, the main idea is to approximately estimate the leaky effect by comparing the total membrane potential and obtain the ratio (β) between both cases (i.e., IF and LIF neurons).

$$\frac{\partial V_{mem}^{total,IF}(t)}{\partial t} \approx V_{th} \frac{\partial a_{IF}(t)}{\partial t} \approx V_{th} rate(t) \tag{17}$$

$$\frac{\partial V_{mem}^{total,LIF}(t)}{\partial t} \approx V_{th} rate(t) + V_{th} \frac{\partial f(t)}{\partial t} \tag{18}$$

$$f(t) = \sum_k exp(-\frac{t - t_k}{\tau_m}) \tag{19}$$

$$\frac{\partial a_{IF}(t)}{\partial t} \approx \frac{1}{V_{th}} \frac{\partial V_{mem}^{total,IF}(t)}{\partial t} = \beta \frac{1}{V_{th}} \frac{\partial V_{mem}^{total,LIF}(t)}{\partial t} \tag{20}$$

By solving the Equations (17, 18, 20), the inverse ratio ($\frac{1}{\beta}$) is derived as follows in Equation (21),

$$\frac{1}{\beta} = 1 + \frac{1}{rate(t)} \frac{\partial f(t)}{\partial t} \tag{21}$$

where the first term (unity) indicates the effect of average input currents (that is observed from the approximate derivative of IF neuron activation, namely the straight-through estimation) and the second term ($\frac{1}{rate(t)} \frac{\partial f(t)}{\partial t}$) represents the leaky (exponential decay) effect of LIF neuron for the forward propagation time window. Then, by using the relations of ϵ and β in Equation (14), the derivative of LIF neuronal activation can be obtained as $\frac{\partial a_{LIF}}{\partial net} = \frac{1}{V_{th}+\epsilon} = \frac{1}{\beta V_{th}}$. In this work, to avoid the vanishing gradient phenomena during the error back-propagation, the leaky effect term ($\frac{1}{rate(t)} \frac{\partial f(t)}{\partial t}$) is divided by the size of the forward propagation time window (T). Hence, the scaled time derivative of this function, $\frac{1}{\gamma} f'(t)$, is used as the leak correctional term where γ denotes the number of output spike events for a particular neuron over the total forward propagation time. As a result, we obtain an approximate derivative for LIF neuronal activation (in hidden layers) as a combination of the straight-through estimation (i.e., approximate derivative of IF neuron activation) and the leak correctional term that compensates leaky effect in the membrane potential as described in Equation (22). Please note that, in our work, input and output spikes are not exponentially decaying, the leak only happens according to the mechanism of membrane potential. Moreover, $f(t)$ is not a part of the forward propagation phase, and rather it is only defined to approximately measure the leaky effect during the backward propagation phase by differentiating it with respect to time. The function $f(t)$ is a time-dependent function that simply integrates the output spikes ($t_k \leq t$) temporally, and the resultant sum is decayed over time. It is evident that $f(t)$ is continuous except where spikes occur and the activities jump up (Lee et al., 2016). Therefore, $f(t)$ is differentiable at $t \rightarrow t_k^+$ (from the right-sided limit). Note that, to capture the leaky effect (exponential decay), it is necessary to compute the derivative of $f(t)$ at the points in between the spiking activities, not at the time instant of spiking.

$$\begin{aligned}\frac{\partial a_{LIF}}{\partial net} &= \frac{1}{V_{th}+\epsilon} = \frac{1}{\beta V_{th}} \approx \frac{1}{V_{th}}(1+\frac{1}{\gamma}f'(t)) \\ &= \frac{1}{V_{th}}(1+\frac{1}{\gamma}\sum_k -\frac{1}{\tau_m}e^{-\frac{t-t_k}{\tau_m}})\end{aligned} \qquad (22)$$

In summary, the approximations applied to implement a spike-based BP algorithm in SNN are as follows:

- During the back-propagating phase, we consider the leaks in the membrane potential of final layer neurons as noises so that the accumulated membrane potential is approximated as equivalent to the total input current ($V_{mem}^L \approx net$). Therefore, the derivative of post-neuronal activation with respect to *net* ($\frac{\partial output}{\partial net}$) is calculated as $\frac{1}{T}$ for the final layer.
- For hidden layers, we first approximate the activation of an IF neuron as a linear function (i.e., straight-through estimation). Hence, we are able to estimate its derivative of IF neuron's activation (Bengio et al., 2013) with respect to total input current.
- To capture the leaky effect of a LIF neuron (in hidden layers), we estimate the scaled time derivative of the low-pass filtered output spikes that leak over time, using the function $f(t)$. This function is continuous except for the time points where spikes occur (Lee et al., 2016). Hence, it is differentiable in the sections between the spiking activities.
- We obtain an approximate derivative for LIF neuronal activation (in hidden layers) as a combination of two derivatives. The first one is the straight-through estimation (i.e., approximate derivative of IF neuron activation). The second one is the leak correctional term that compensates the leaky effect in the membrane potential of LIF neurons. The combination of straight-through estimation and the leak correctional term is expected to be less than 1.

Based on these approximations, we can train SNNs with direct spike inputs using a spike-based BP algorithm.

At the final layer, the error gradient, δ^L, represents the gradient of the output loss with respect to total input current (i.e., *net*) received by the post-neurons. It can be calculated by multiplying the final output error (e) with the derivative of the corresponding post-neuronal activation ($\frac{\partial output}{\partial net^L}$) as shown in Equation (23). At any hidden layer, the local error gradient, δ^l, is recursively estimated by multiplying the back-propagated gradient from the following layer ($(w^l)^{Tr} * \delta^{l+1}$) with derivative of the neuronal activation, $a'_{LIF}(net^l)$, as presented in Equation (24). Note that element-wise multiplication is indicated by "." while matrix multiplication is represented by "*" in the respective equations.

$$\delta^L = \frac{\partial E}{\partial output}\frac{\partial output}{\partial net^L} = e\frac{1}{T} = \frac{e}{T} \qquad (23)$$

$$\delta^l = ((w^l)^{Tr} * \delta^{l+1}).a'_{LIF}(net^l) \qquad (24)$$

The derivative of *net* with respect to weight is simply the total incoming spikes over time as derived in Equation (25). The derivative of the output loss with respect to the weights interconnecting the layers l and $l+1$ ($\triangle w^l$ in Equation, 26) is determined by multiplying the transposed error gradient at $l+1$ (δ^{l+1}) with the input spikes from layer l. Finally, the calculated partial derivatives of loss function are used to update the respective weights using a learning rate (η_{BP}) as illustrated in Equation (27). As a result, iterative updating of the weights over mini-batches of input patterns leads the network state to a local minimum, thereby enabling the network to capture multiple-levels of internal representations of the data.

$$\frac{\partial net}{\partial w^l} = \frac{\partial}{\partial w^l}(w^l * x^l(t)) = x^l(t) \qquad (25)$$

$$\triangle w^l = \frac{\partial E}{\partial w^l} = x^l(t) * (\delta^{l+1})^{Tr} \qquad (26)$$

$$w^l_{updated} = w^l - \eta_{BP}\triangle w^l \qquad (27)$$

2.2.2. Dropout in Spiking Neural Network

Dropout (Srivastava et al., 2014) is one of the popular regularization techniques while training deep ANNs. This technique randomly disconnects certain units with a given probability (p) to avoid units being overfitted and co-adapted too much to given training data. There are prior works (Kappel et al., 2015, 2018; Neftci et al., 2015) that investigated the biological insights on how synaptic stochasticity can provide dropout-like functional benefits in SNNs. In this work, we employ the concept of dropout technique in order to regularize deep SNNs effectively. Note, dropout technique is only applied during training and is not used when evaluating the performance of the network during inference. There is a subtle difference in the way dropout is applied in SNNs compared to ANNs. In ANNs, each epoch of training has several iterations of mini-batches. In each iteration, randomly selected units (with dropout ratio of p) are disconnected from the network while weighting by its posterior probability ($\frac{1}{1-p}$). However, in SNNs, each iteration has more than one forward propagation depending on the time length of

Algorithm 1: Forward propagation with dropout at each iteration in SNN

```
1:  Input : Poisson-distributed input spike train ($inputs$), Dropout ratio ($p$), Total number of time steps ($\#timesteps$), Membrane potential ($V_{mem}$), Time constant of membrane potential ($\tau_m$), Firing threshold ($V_{th}$)
2:  Initialize $SNN^l.V_{mem} \leftarrow 0\ \forall l = 2, ..., \#SNN.layer$
3:  // Define the random subset of units (with a probability $1-p$) at each iteration
4:  for $l \leftarrow 1$ to $\#SNN.layer - 1$ do
5:      $mask^l \leftarrow generate\_random\_subset(probability = 1 - p)$
6:  for $t \leftarrow 1$ to $\#timesteps$ do
7:      // Set input of first layer equal to spike train of a mini-batch data
8:      $SNN^1.spike[t] \leftarrow inputs[t]$;
9:      for $l \leftarrow 2$ to $\#SNN.layer$ do
10:         // Integrate weighted sum of input spikes to membrane potential
11:         $SNN^l.V_{mem}[t] \leftarrow SNN^l.V_{mem}[t-1] + SNN^{l-1}forward(SNN^{l-1}.spike[t]). * (mask^{l-1}/(1\text{-}p))$;
12:         // If $V_{mem}$ is greater than $V_{th}$, post-neuron generate a spike
13:         if $SNN^l.V_{mem}[t] > SNN^l.V_{th}$ then
14:             // Membrane potential resets if the corresponding neuron fires a spike
15:             $SNN^l.spike[t] \leftarrow 1$
16:             $SNN^l.V_{mem}[t] \leftarrow 0$
17:         else
18:             // Else, membrane potential decays over time
19:             $SNN^l.spike[t] \leftarrow 0$
20:             $SNN^l.V_{mem}[t] \leftarrow e^{-\frac{1}{\tau_m}} * SNN^l.V_{mem}[t]$
```

the spike train. We back-propagate the output error and modify the network parameters only at the last time step. For dropout to be effective in our training method, it has to be ensured that the set of connected units within an iteration of mini-batch data is not changed, such that the neural network is constituted by the same random subset of units during each forward propagation within a single iteration. On the other hand, if the units are randomly connected at each time-step, the effect of dropout will be averaged out over the entire forward propagation time within an iteration. Then, the dropout effect would fade-out once the output error is propagated backward and the parameters are updated at the last time step. Therefore, we need to keep the set of randomly connected units for the entire time window within an iteration. In the experiment, we use the SNN version of dropout technique with the probability (p) of omitting units equal to 0.2–0.25. Note that the activations are much sparser in SNN forward propagations compared to ANNs, hence the optimal p for SNNs needs to be less than a typical ANN dropout ratio (p = 0.5). The details of SNN forward propagation with dropout are specified in **Algorithm 1**.

TABLE 2 | Parameters used in the experiments.

Parameter	Value
Time Constant of Membrane Potential (τ_m)	100 time-steps
BP Training Time Duration	50–100 time-steps
Inference Time Duration	Same as training
Mini-batch Size	16–32
Spatial-pooling Non-overlapping Region/Stride	2×2, 2
Neuronal Firing Threshold	1 (hidden layer), ∞ (final layer)
Weight Initialization Constant (κ)	2 (non-residual network), 1 (residual network)
Learning rate (η_{BP})	0.002–0.003
Dropout Ratio (p)	0.2–0.25

3. RESULTS

3.1. Experimental Setup

The primary goal of our experiments is to demonstrate the effectiveness of the proposed spike-based BP training methodology in a variety of deep network architectures. We first describe our experimental setup and baselines. For the experiments, we developed a custom simulation framework using the Pytorch (Paszke et al., 2017) deep learning package for evaluating our proposed SNN training algorithm. Our deep convolutional SNNs are populated with biologically plausible LIF neurons (with neuronal firing thresholds of unity) in which a pair of pre- and post- neurons are interconnected by plastic synapses. At the beginning, the synaptic weights are initialized with Gaussian random distribution of zero-mean and standard deviation of $\sqrt{\frac{\kappa}{n^l}}$ (n^l: number of fan-in synapses) as introduced in He et al. (2015). Note, the initialization constant κ differs by the type of network architecture. For instance, we have used $\kappa = 2$ for non-residual network and $\kappa = 1$ for residual network. For training, the synaptic weights are trained with a mini-batch spike-based BP algorithm in an end-to-end manner, as explained in section 2.2.1. For static datasets, we train our network models for 150 epochs using mini-batch stochastic gradient descent BP that reduces its learning rate at 70, 100, and 125th training epochs. For the neuromorphic dataset, we use Adam (Kingma and Ba, 2014) learning method and reduce its learning rate at 40, 80, and 120th training epochs. Please, refer to **Table 2** for more implementation details. The datasets and network topologies used for benchmarking, the input spike generation scheme for event-based operation and determination of the number of time-steps required for training and inference are described in the following sub-sections.

3.1.1. Benchmarking Datasets

We demonstrate the efficacy of our proposed training methodology for deep convolutional SNNs on three standard vision datasets and one neuromorphic vision dataset, namely the MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky and Hinton, 2009), and N-MNIST (Orchard et al., 2015). The MNIST dataset is composed of gray-scale

TABLE 3 | Benchmark datasets.

Dataset	Image	#Training samples	#Testing samples	#Category
MNIST	28 × 28, gray	60,000	10,000	10
SVHN	32 × 32, color	73,000	26,000	10
CIFAR-10	32 × 32, color	50,000	10,000	10
N-MNIST	34 × 34 × 2 ON and OFF spikes	60,000	10,000	10

(one-dimensional) images of handwritten digits whose sizes are 28 by 28. The SVHN and CIFAR-10 datasets are composed of color (three-dimensional) images whose sizes are 32 by 32. The N-MNIST dataset is a neuromorphic (spiking) dataset that is converted from static MNIST dataset using Dynamic Vision Sensor (DVS) (Lichtsteiner et al., 2008). The N-MNIST dataset contains two-dimensional images that include ON and OFF event stream data whose sizes are 34 by 34. The ON (OFF) event represents the increase (decrease) in pixel bright changes. The details of the benchmark datasets are listed in **Table 3**. For evaluation, we report the top-1 classification accuracy by classifying the test samples (training samples and test samples are mutually exclusive).

3.1.2. Network Topologies

We use various SNN architectures depending on the complexity of the benchmark datasets. For MNIST and N-MNIST datasets, we used a network consisting of two sets of alternating convolutional and spatial-pooling layers followed by two fully-connected layers. This network architecture is derived from LeNet5 model (LeCun et al., 1998). Note that **Table 4** summarizes the layer type, kernel size, the number of output feature maps, and stride of SNN model for MNIST dataset. The kernel size shown in the table is for 3-D convolution where the 1st dimension is for number of input feature-maps and 2nd–3rd dimensions are for convolutional kernels. For SVHN and CIFAR-10 datasets, we used deeper network models consisting of 7 to 11 trainable layers including convolutional, spatial-pooling and fully-connected layers. In particular, these networks consisting of beyond 5 trainable layers are constructed using small (3 × 3) convolutional kernels. We term the deep convolutional SNN architecture that includes 3 × 3 convolutional kernel (Simonyan and Zisserman, 2014) without residual connections as "VGG SNN" and with skip (residual) connections (He et al., 2016) as "Residual SNN." In Residual SNNs, some convolutional layers convolve kernel with the stride of 2 in both x and y directions, to incorporate the functionality of spatial-pooling layers. Please, refer to **Tables 4**, **5** that summarize the details of deep convolutional SNN architectures. In the results section, we will discuss the benefit of deep SNNs in terms of classification performance as well as inference speedup and energy efficiency.

3.1.3. ANN-SNN Conversion Scheme

As mentioned previously, off-the-shelf trained ANNs can be successfully converted to SNNs by replacing ANN (ReLU) neurons with Integrate and Fire (IF) spiking neurons and adjusting the neuronal thresholds with respect to synaptic weights. In the literature, several methods have been proposed (Cao et al., 2015; Hunsberger and Eliasmith, 2015; Diehl et al., 2016; Rueckauer et al., 2017; Sengupta et al., 2019) for balancing appropriate ratios between neuronal thresholds and synaptic weights of spiking neuron in the case of ANN-SNN conversion. In this paper, we compare various aspects of our direct-spike trained models with two prior ANN-SNN conversion works (Sengupta et al. 2019; Diehl et al. 2016), which proposed near-lossless ANN-SNN conversion schemes for deep network architectures. The first scheme (Sengupta et al. 2019) balanced the neuronal firing thresholds with respect to corresponding synaptic weights layer-by-layer depending on the actual spiking activities of each layer using a subset of training samples. The second scheme (Diehl et al. 2016) balanced the neuronal firing thresholds with the consideration of ReLU activations in the corresponding ANN layer. Basically, we compare our direct-spike trained model with converted SNNs on the same network architecture in terms of accuracy, inference speed and energy-efficiency. Please note that there are a couple of differences on the network architecture between the conversion networks (Sengupta et al. 2019; Diehl et al. 2016) and our scheme. First, the conversion networks always use average-pooling to reduce the size of previous convolutional output feature-map, whereas our models interchangeably use average pooling or convolve kernels with a stride of 2 in the convolutional layer. Next, the conversion networks only consider identity skip connections for residual SNNs. However, we implement skip connections using either identity mapping or 1 × 1 convolutional kernel.

3.1.4. Spike Generation Scheme

For the static vision datasets (MNIST, SVHN, and CIFAR-10), each input pixel intensity is converted to a stream of Poisson-distributed spike events that have equivalent firing rates. Specifically, at each time step, the pixel intensity is compared with a uniformly distributed random number (in the range between 0 and 1). If pixel intensity is greater than the random number at the corresponding time step, a spike is generated. This rate-based spike encoding is used to feed the input spikes to the network for a given period of time during both training and inference. For color image datasets, we use the pre-processing technique of horizontal flip before generating input spikes. These input pixels are normalized to represent zero mean and unit standard deviation. Thereafter, we scale the pixel intensities to bound them in the range [–1,1] to represent the whole spectrum of input pixel representations. The normalized pixel intensities are converted to Poisson-distributed spike events such that the generated input signals are bipolar spikes. For the neuromorphic version of the dataset (N-MNIST), we use the original (unfiltered and uncentered) version of spike streams to directly train and test the network in the time domain.

3.1.5. Time-Steps

As mentioned in section 3.1.4, we generate a stochastic Poisson-distributed spike train for each input pixel intensity for event-based operation. The duration of the spike train is very important

TABLE 4 | The deep convolutional spiking neural network architectures for MNIST, N-MNIST, and SVHN dataset.

4 layer network				VGG7				ResNet7			
Layer type	**Kernel size**	**#o/p feature-maps**	**Stride**	**Layer type**	**Kernel size**	**#o/p feature-maps**	**Stride**	**Layer type**	**Kernel size**	**#o/p feature-maps**	**Stride**
Convolution	1×5×5	20	1	Convolution	3×3×3	64	1	Convolution	3×3×3	64	1
Average-pooling	2×2		2	Convolution	64×3×3	64	2	Average-pooling	2×2		2
				Average-pooling	2×2		2				
Convolution	20×5×5	50	1	Convolution	64×3×3	128	1	Convolution	64×3×3	128	1
Average-pooling	2×2		2	Convolution	128×3×3	128	2	Convolution	128×3×3	128	2
				Convolution	128×3×3	128	2	Skip convolution	64×1×1	128	2
				Average-pooling	2×2		2				
								Convolution	128×3×3	256	1
								Convolution	256×3×3	256	2
								Skip convolution	128×1×1	256	2
Fully-connected		200		Fully-connected		1024		Fully-connected		1024	
Output		10		Output		10		Output		10	

TABLE 5 | The deep convolutional spiking neural network architectures for a CIFAR-10 dataset.

VGG9				ResNet9				ResNet11			
Layer type	**Kernel size**	**#o/p feature-maps**	**Stride**	**Layer type**	**Kernel size**	**#o/p feature-maps**	**Stride**	**Layer type**	**Kernel size**	**#o/p feature-maps**	**Stride**
Convolution	3×3×3	64	1	Convolution	3×3×3	64	1	Convolution	3×3×3	64	1
Convolution	64×3×3	64	1	Average-pooling	2×2		2	Average-pooling	2×2		2
Average-pooling	2×2		2								
Convolution	64×3×3	128	1	Convolution	64×3×3	128	1	Convolution	64×3×3	128	1
Convolution	128×3×3	128	1	Convolution	128×3×3	128	1	Convolution	128×3×3	128	1
Average-pooling	2×2		2	Skip convolution	64×1×1	128	1	Skip convolution	64×1×1	128	1
Convolution	128×3×3	256	1	Convolution	128×3×3	256	1	Convolution	128×3×3	256	1
Convolution	256×3×3	256	1	Convolution	256×3×3	256	2	Convolution	256×3×3	256	2
Convolution	256×3×3	256	1	Skip connection	128×1×1	256	2	Skip convolution	128×1×1	256	2
Average-pooling	2×2		2								
				Convolution	256×3×3	512	1	Convolution	256×3×3	512	1
				Convolution	512×3×3	512	2	Convolution	512×3×3	512	1
				Skip convolution	256×1×1	512	2	Skip convolution	512×1×1	512	1
								Convolution	512×3×3	512	1
								Convolution	512×3×3	512	2
								Skip convolution	512×1×1	512	2
Fully-connected		1024		Fully-connected		1024		Fully-connected		1024	
Output		10		Output		10		Output		10	

for SNNs. We measure the length of the spike train (spike time window) in time-steps. For example, a 100 time-step spike train will have approximately 50 random spikes if the corresponding pixel intensity is half in a range of [0,1]. If the number of time-steps (spike time window) is too less, then the SNN will not receive enough information for training or inference. On the other hand, a large number of time-steps will destroy the stochastic property of SNNs and get rid of noise and imprecision at the cost of high latency and power consumption. Hence, the network will not have much energy efficiency over ANN implementations. For these reasons, we experimented with the different number of time-steps to empirically obtain the optimal number of time-steps required for both training and inference. The experimental process and results are explained in the following subsections.

3.1.5.1. Optimal #time-steps for Training

A spike event can only represent 0 or 1 in each time step, therefore usually its bit precision is considered 1. However, the spike train provides temporal data, which is an additional source of information. Therefore, the spike train length (number of time-steps) in SNN can be considered as its actual precision of neuronal activation. To obtain the optimal #time-steps required for our proposed training method, we trained VGG9 networks on

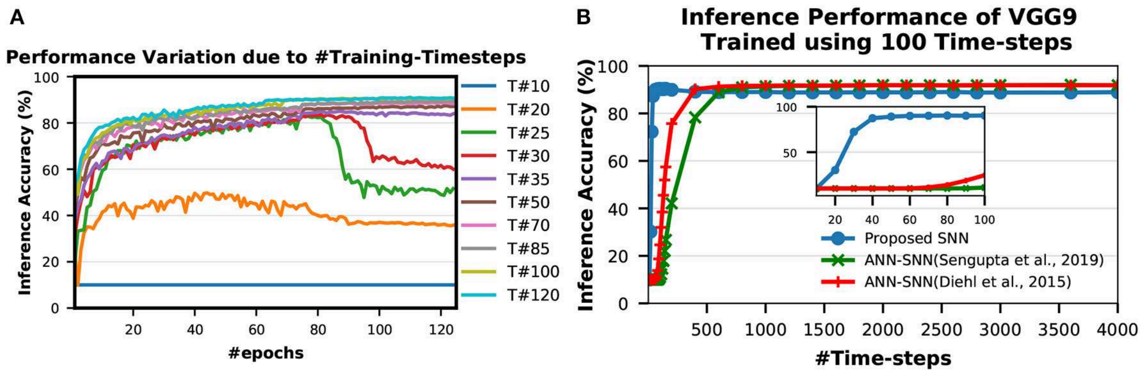


FIGURE 6 | Inference performance variation due to **(A)** #Training-Timesteps and **(B)** #Inference-Timesteps. *T#* in **(A)** indicates number of time-steps used for training. **(A)** shows that inference accuracy starts to saturate as #training-timesteps increase. In **(B)**, the zoomed version on the inset shows that the SNN trained with the proposed scheme performs very well even with only 30 time-steps while the peak performance occurs around 100 time-steps.

CIFAR-10 dataset using different time-steps ranging from 10 to 120 (shown in **Figure 6A**). We found that for only 10 time-steps, the network is unable to learn anything as there is not enough information (input precision too low) for the network to be able to learn. This phenomenon is explained by the lack of spikes in the final output. With the initial weights, the accumulated sum of the LIF neuron is not enough to generate output spikes in the later layers. Hence, none of the input spikes propagates to the final output neurons and the output distributions remain 0. Therefore, the computed gradients are always 0 and the network is not updated. For 35–50 time-steps, the network learns well and converges to a reasonable point. From 70 time-steps, the network accuracy starts to saturate. At about 100 time-steps, the network training improvement completely saturates. This is consistent with the bit precision of the inputs. It has been shown in Sarwar et al. (2018) that 8 bit inputs and activations are sufficient to achieve optimal network performance for standard image recognition tasks. Ideally, we need 128 time-steps to represent 8 bit inputs using bipolar spikes. However, 100 time-steps proved to be sufficient as more time-steps provide marginal improvement. We observe a similar trend in VGG7, ResNet7, ResNet9, and ResNet11 SNNs as well while training for SVHN and CIFAR-10 datasets. Therefore, we considered 100 time-steps as the optimal #time-steps for training in our proposed methodology. Moreover, for MNIST dataset, we used 50 time-steps since the required bit precision is only 4 bits (Sarwar et al., 2018).

3.1.5.2. Optimal #time-steps for inference

To obtain the optimal #time-steps required for inferring an image utilizing a network trained with our proposed method, we conducted similar experiments as described in section 3.1.5. We first trained a VGG9 network for CIFAR-10 dataset using 100 time-steps (optimal according to experiments in section 3.1.5). Then, we tested the network performances with different time-steps ranging from 10 to 4,000 (shown in **Figure 6B**). We observed that the network performs very well even with only 30 time-steps while the peak performance occurs around 100 time-steps. For more than 100 time-steps, the accuracy degrades slightly from the peak. This behavior is very different from ANN-SNN converted networks where the accuracy keeps on improving as #time-steps is increased (shown in **Figure 6B**). This can be attributed to the fact that our proposed spike-based training method incorporates the temporal information well into the network training procedure so that the trained network is tailored to perform best at a specific spike time window for the inference. On the other hand, the ANN-SNN conversion schemes are unable to incorporate the temporal information of the input in the trained network. Hence, the ANN-SNN conversion schemes require much higher #time-steps (compared to SNN trained using the proposed method) for the inference in order to resemble input-output mappings similar to ANNs.

3.2. Results

In this section, we analyze the classification performance and efficiency achieved by the proposed spike-based training methodology for deep convolutional SNNs compared to the performance of the transformed SNN using ANN-SNN conversion scheme.

3.2.1. The Classification Performance

Most of the classification performances available in the literature for SNNs are for MNIST and CIFAR-10 datasets. The popular methods for SNN training are "Spike Time Dependent Plasticity (STDP)" based unsupervised learning (Brader et al., 2007; Diehl and Cook, 2015; Srinivasan et al., 2018a,b) and "spike-based backpropagation" based supervised learning (Lee et al., 2016; Neftci et al., 2017; Jin et al., 2018; Mostafa, 2018; Wu et al., 2018b). There are a few works (Kheradpisheh et al., 2016; Tavanaei and Maida, 2016, 2017; Lee et al., 2018) which tried to combine the two approaches to get the best of both worlds. However, these training methods were able to neither train deep SNNs nor achieve good inference performance compared to ANN implementations. Hence, ANN-SNN conversion schemes have been explored by researchers (Cao et al., 2015; Hunsberger and Eliasmith, 2015; Diehl et al., 2016; Rueckauer et al., 2017;

TABLE 6 | Comparison of the SNNs classification accuracies on MNIST, N-MNIST, and CIFAR-10 datasets.

Model	Learning method	Accuracy (MNIST)	Accuracy (N-MNIST)	Accuracy (CIFAR-10)
Hunsberger and Eliasmith (2015)	Offline learning, conversion	98.37%	–	82.95%
Esser et al. (2016)	Offline learning, conversion	–	–	89.32%
Diehl et al. (2016)	Offline learning, conversion	99.10%	–	–
Rueckauer et al. (2017)	Offline learning, conversion	99.44%	–	88.82%
Sengupta et al. (2019)	Offline learning, conversion	–	–	91.55%
Kheradpisheh et al. (2016)	Layerwise STDP + offline SVM classifier	98.40%	–	–
Panda and Roy (2016)	Spike-based autoencoder	99.08%	–	70.16%
Lee et al. (2016)	Spike-based BP	99.31%	98.74%	–
Wu et al. (2018b)	Spike-based BP	99.42%	98.78%	50.70%
Lee et al. (2018)	STDP-based pretraining + spike-based BP	99.28%	–	–
Jin et al. (2018)	Spike-based BP	99.49%	98.88%	–
Wu et al. (2018a)	Spike-based BP	–	99.53%	90.53%
This work	Spike-based BP	99.59%	99.09%	90.95%

Sengupta et al., 2019). Till date, ANN-SNN conversion schemes achieved the best inference performance for CIFAR-10 dataset using deep networks (Rueckauer et al., 2017; Sengupta et al., 2019). Classification performances of all these works are listed in **Table 6** along with ours. To the best of our knowledge, we achieved the best inference accuracy for MNIST using LeNet structured network compared to our spike based training approaches. We also achieved accuracy performance comparable with ANN-SNN converted network (Diehl et al., 2015; Sengupta et al., 2019) for CIFAR-10 dataset while beating all other spike-based training methods.

For a more extensive comparison, we compare the inference performances of trained networks using our proposed methodology with the ANNs and ANN-SNN conversion scheme for same network configuration (depth and structure) side by side in **Table 7**. We also compare with the previous best SNN training results found in the literature that may or may not have the same network depth and structure as ours. The ANN-SNN conversion scheme is a modified and improved version of Sengupta et al. (2019). We are using this modified scheme since it achieves better conversion performance than (Sengupta et al. 2019) as explained in section 3.1.3. Note that all reported classification accuracies are the average of the maximum inference accuracies for 3 independent runs with different seeds.

After initializing the weights, we train the SNNs using a spike-based BP algorithm in an end-to-end manner with Poisson-distributed spike train inputs. Our evaluation of MNIST dataset yields a classification accuracy of 99.59%, which is the best compared to any other SNN training scheme and also identical to other ANN-SNN conversion schemes. We achieve ~96% inference accuracy on SVHN dataset for both trained non-residual and residual SNN. Inference performance for SNNs trained on SVHN dataset has not been reported previously in the literature. We implemented three different networks, as shown in **Table 5**, for classifying CIFAR-10 dataset using a proposed spike-based BP algorithm. For the VGG9 network, the ANN-SNN conversion schemes provides a near lossless converted network compared to baseline ANN implementation while our proposed training method yields a classification accuracy of 90.45%. For ResNet9 network, the ANN-SNN conversion schemes provide inference accuracy within 0.5–1% of baseline ANN implementation. However, our proposed spike-based training method achieves inference accuracy that is within ~1.5% of baseline ANN implementation. In the case of ResNet11, we observe that the inference accuracy improvement is marginal compared to ResNet9 for baseline ANN implementation and ANN-SNN conversion schemes. However, our proposed SNN training shows improvement of ~0.5% for ResNet11 compared to ResNet9. Overall, our proposed training method achieves comparable inference accuracies for both ResNet and VGG networks compared to baseline ANN implementation and ANN-SNN conversion schemes.

3.2.2. Accuracy Improvement With Network Depth

In order to analyze the effect of network depth for SNNs, we experimented with networks of different depths while training for SVHN and CIFAR-10 datasets. For SVHN dataset, we started with a shallow network derived from LeNet5 model (LeCun et al., 1998) with 2 convolutional and 2 fully-connected layers. This network was able to achieve inference accuracy of only 92.38%. Then, we increased the network depth by adding 1 convolutional layer before the 2 fully-connected layers and we termed this network as VGG5. VGG5 network was able to achieve significant improvement over its predecessor. Similarly, we tried VGG6 followed by VGG7, and the improvement started to become very small. We have also trained ResNet7 to understand how residual networks perform compared to non-residual networks of similar depth. The results of these experiments are shown in **Figure 7A**. We carried out similar experiments for CIFAR-10 dataset as well. The results show a similar trend as described in **Figure 7B**. These results ensure that network depth improves the learning capacity of direct-spike trained SNNs similar to ANNs. The non-residual networks saturate at a certain depth and start to degrade

TABLE 7 | Comparison of classification performance.

		Inference Accuracy				
Dataset	**Model**	**ANN**	**ANN-SNN** (Diehl et al., 2015)	**ANN-SNN** (Sengupta et al., 2019)	**SNN (Previous Best)**	**SNN This Work)**
MNIST	LeNet	99.57%	99.55%	99.59%	99.49% (Jin et al., 2018)	99.59%
N-MNIST	LeNet	–	–	–	99.53% (Wu et al., 2018a)	99.09%
SVHN	VGG7	96.36%	96.33%	96.30%	–	96.06%
	ResNet7	96.43%	96.33%	96.40%	–	96.21%
CIFAR-10	VGG9	91.98%	91.89%	92.01%		90.45%
	ResNet9	91.85%	90.78%	91.59%	90.53% (Wu et al., 2018a)	90.35%
	ResNet11	91.87%	90.98%	91.65%		90.95%

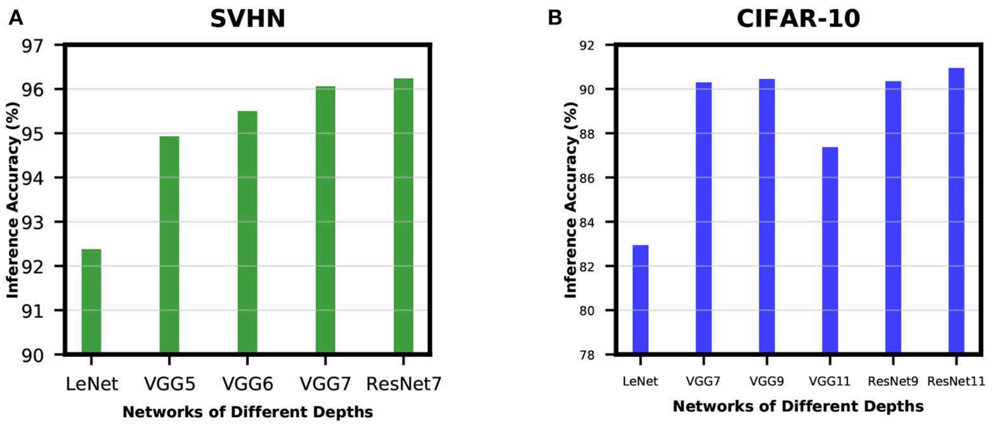


FIGURE 7 | Accuracy improvement with network depth for **(A)** SVHN dataset and **(B)** CIFAR-10 dataset. In **(A)**, inference accuracy improves with an increase in network depth. In **(B)**, the non-residual networks saturate at a certain depth and start to degrade if network depth increases further. However, the residual blocks in deep residual ANNs allow the network to maintain peak classification accuracy (ResNet9 and ResNet11).

if network depth is further increased (VGG11 in **Figure 7B**) due to the degradation problem mentioned in He et al. (2016). In such a scenario, the residual connections in deep residual ANNs allow the network to maintain peak classification accuracy utilizing the skip connections (He et al., 2016), as seen in **Figure 7B** (ResNet9 and ResNet11).

4. DISCUSSION

4.1. Comparison With Relevant Works

In this section, we compare our proposed supervised learning algorithm with other recent spike-based BP algorithms. The spike-based learning rules primarily focus on directly training and testing SNNs with spike-trains, and no conversion is necessary for applying in real-world spiking scenario. In recent years, there are an increasing number of supervised gradient descent method in spike-based learning. The Panda and Roy (2016) developed a spike-based auto-encoder mechanism to train deep convolutional SNNs. They dealt with membrane potential as a differentiable signal and showed recognition capabilities in standard vision tasks (MNIST and CIFAR-10 datasets). Meanwhile, Lee et al. (2016) followed the approach using differentiable membrane potential to explore a spike-based BP algorithm in an end-to-end manner. In addition, Lee et al. (2016) presented the error normalization scheme to prevent exploding gradient phenomena while training deep SNNs. Researchers in Jin et al. (2018) proposed hybrid macro/micro level backpropagation (HM2-BP). HM2-BP is developed to capture the temporal effect of the individual spike (in micro-level) and rate-encoded error (at macro-level). The reference Shrestha and Orchard (2018) employed exponential function for the approximate derivative of neuronal function and developed a credit assignment scheme to calculate the temporal dependencies of error throughout the layers. Huh and Sejnowski (2018) has trained recurrent spiking networks by replacing the threshold with a gate function and employing BPTT technique (Werbos, 1990). While BPTT technique has been a popular method to train recurrent artificial and spiking recurrent networks, Lillicrap and Santoro (2019) points out the storing and retrieving past variables and differentiation them through time in biological neurons seems to be impossible. Recently, e-prop Bellec et al. (2019) presented an approximation method to bypass neuronal state savings for enhancing the computational efficiency of BPTT. In temporal spike encoding domain, Mostafa (2018) proposed an interesting temporal spike-based BP algorithm by treating the spike-time as the differential activation of neuron. Temporal encoding based SNN has the potential to process the tasks with the small number of spikes. All of these works demonstrated spike-based learning in simple network architectures and has a large gap in classification accuracy compared to deep ANNs. More recently, Wu et al. (2018a) presented a neuron normalization technique (called NeuNorm) that calculates the average input firing rates to adjust neuron selectivity. NeuNorm enables spike-based training within a relatively short time-window while achieving competitive performances. In addition, they presented an input encoding scheme that receives both spike and non-spike signals for preserving the precision of input data.

There are several points that distinguish our work from others. First, we use a pseudo derivative method that accounts for leaky effect in membrane potential of LIF neurons. We approximately estimate the leaky effect by comparing total membrane potential value and obtain the ratio between IF and LIF neurons. During the back-propagating phase, the pseudo derivative of LIF neuronal function is estimated by combining the straight through estimation and leak correctional term as described in Equation (22). Next, we construct our networks by leveraging frequently used architectures such as VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016). To the best of our knowledge, this is the first work that demonstrates spike-based supervised BP learning for SNNs containing more than 10 trainable layers. Our deep SNNs obtain the superior classification accuracies in MNIST, SVHN, and CIFAR-10 datasets in comparison to the other networks trained with the spike-based algorithm. In addition, as opposed to complex error or neuron normalization method adopted by Lee et al. (2016) and Wu et al. (2018a), respectively, we demonstrate that deep SNNs can be naturally trained by only accounting for spiking activities of the network. As a result, our work paves the effective way for training deep SNNs with a spike-based BP algorithm.

4.2. Spike Activity Analysis

The most important advantage of event-based operation of neural networks is that the events are very sparse in nature. To verify this claim, we analyzed the spiking activities of the direct-spike trained SNNs and ANN-SNN converted networks in the following subsections.

4.2.1. Spike Activity per Layer

The layer-wise spike activities of both SNN trained using our proposed methodology, and ANN-SNN converted network (using scheme 1) for VGG9 and ResNet9 are shown in **Figures 8A,B**, respectively. In the case of ResNet9, only the first average pooling layer's output spike activity is shown in the figure as for the direct-spike trained SNN, the other spatial-poolings are done by stride 2 convolutions. In **Figure 8**, it can be seen that the input layer has the highest spike activity that is significantly higher than any other layer. The spike activity reduces significantly as the network depth increases.

We can observe from **Figures 8A,B** that the average spike activity in a direct-spike trained SNN is much higher than ANN-SNN converted network. The ANN-SNN converted network uses a higher threshold compared to 1 (in case of direct-spike trained SNN) since the conversion scheme applies layer-wise neuronal threshold modulation. This higher threshold reduces spike activity in ANN-SNN converted networks. However, in both cases, the spike activity decreases with increasing network depth.

4.2.2. #Spikes/Inference

From **Figure 8**, it is evident that average spike activity in ANN-SNN converted networks is much less than in the direct-spike trained SNN. However, for inference, the network has to be evaluated over many time-steps. Therefore, to quantify the actual spike activity for an inference operation, we measured the average number of spikes required for inferring one image. For this purpose, we counted the number of spikes generated (including input spikes) for classifying the test set of a particular dataset for a specific number of time-steps and averaged the count for generating the quantity *"#spikes per image inference."* We have used two different time-steps for ANN-SNN converted networks; one for iso-accuracy comparison and the other for maximum accuracy comparison with the direct-spike trained SNNs. Iso-accuracy inference requires less #time-steps than maximum accuracy inference, hence has a lower number of spikes per image inference. For few networks, the ANN-SNN conversion scheme always provides accuracy less than or equal to the direct-spike trained SNN. Hence, we only compare spikes per image inference in maximum accuracy condition for those ANN-SNN converted networks while comparing with direct-spike trained SNNs. For the analysis, we quantify the spike-efficiency (amount reduction in #spikes) from the #spikes/image inference. The results are listed in **Table 8**, where the 1st row corresponds to iso-accuracy and the 2nd row corresponds to maximum-accuracy condition for each network. As shown in **Table 8**, the direct-spike trained SNNs are more efficient in terms of #spikes/inference compared to the ANN-SNN converted networks for the maximum accuracy condition. For an iso-accuracy condition, only deep SNNs (such as VGG9 and ResNet11) are more efficient in terms of #spikes/inference compared to the ANN-SNN converted networks.

The **Figure 9** shows the relationship between inference accuracy, latency and #spikes/inference for ResNet11 networks trained on CIFAR-10 dataset. We can observe that #spikes/inference is higher for direct-spike trained SNN compared to ANN-SNN converted networks at any particular latency. However, SNN trained with spike-based BP requires only 100 time-steps for maximum inference accuracy, whereas ANN-SNN converted networks require 3,000–3,500 time-steps to reach maximum inference accuracy. Hence, under maximum accuracy condition, direct-spike trained ResNet11 requires much fewer #spikes/inference compared to ANN-SNN converted networks, while achieving comparable accuracy. Even under iso-accuracy condition, the direct-spike trained ResNet11 requires fewer #spikes/inference compared to the ANN-SNN converted networks (**Table 8**).

4.3. Inference Speedup

The time required for inference is linearly proportional to the #time-steps (**Figure 9**). Hence, we can also quantify the inference speedup for direct-spike trained SNNs compared to ANN-SNN

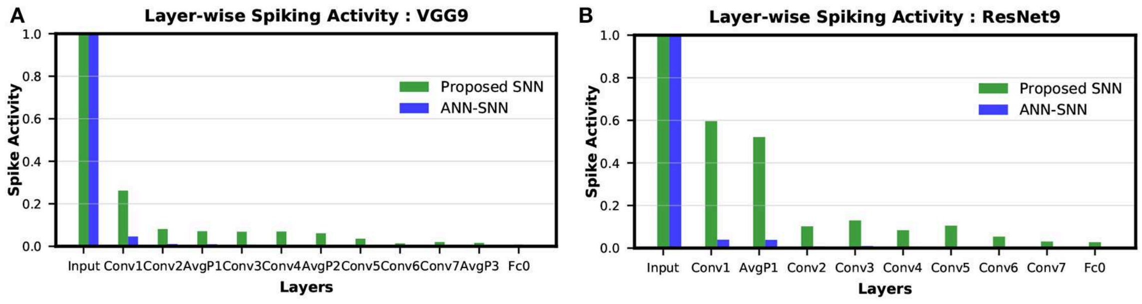


FIGURE 8 | Layer-wise spike activity in direct-spike trained SNN and ANN-SNN converted network for CIFAR-10 dataset: **(A)** VGG9 **(B)** ResNet9 network. The spike activity is normalized with respect to the input layer spike activity, which is the same for both networks. The spike activity reduces significantly for both SNN and ANN-SNN converted network toward the later layers. We have used scheme 1 for ANN-SNN conversion.

TABLE 8 | #Spikes/Image inference and spike efficiency comparison between SNN and ANN-SNN converted networks for benchmark datasets trained on different network models.

Dataset	**Model**	**Spike/Image**			**Spike efficiency compared to**	
		SNN	**ANN-SNN** (Diehl et al., 2015)	**ANN-SNN** (Sengupta et al., 2019)	**ANN-SNN** (Diehl et al., 2015)	**ANN-SNN** (Sengupta et al., 2019)
MNIST	LeNet	5.52E+04	3.4E+04	2.9E+04	0.62x	0.53x
				7.3E+04		1.32x
SVHN	VGG7	5.56E+06	3.7E+06	1.0E+07	0.67x	1.84x
			1.9E+07	1.7E+07	3.40x	2.99x
	ResNet7	4.66E+06	3.9E+06	3.1E+06	0.85x	0.67x
			2.4E+07	2.0E+07	5.19x	4.30x
CIFAR-10	VGG9	1.24E+06	1.6E+06	2.2E+06	1.32x	1.80x
			8.3E+06	9.6E+06	6.68x	7.78x
	ResNet9	4.32E+06	2.7E+06	1.5E+06	0.63x	0.35x
			1.0E+07	7.8E+06	2.39x	1.80x
	ResNet11	1.53E+06	9.7E+06	1.8E+06	6.33x	1.17x
				9.2E+06		5.99x

(For each network, the 1st row corresponds to iso-accuracy and the 2nd row corresponds to maximum-accuracy condition).

converted networks from the #time-steps required for inference, as shown in **Table 9**. For example, for VGG9 network, the proposed training method can achieve 8x (5x) speedup for iso-accuracy and up to 36x (25x) speedup for maximum accuracy in inference compared to respective ANN-SNN converted networks [i.e., scheme 1 (Sengupta et al., 2019) and scheme 2 (Diehl et al., 2016)]. Similarly, for ResNet networks, the proposed training method can achieve 6x speedup for iso-accuracy and up to 35x speedup for maximum accuracy condition in inference. It is interesting to note that direct-spike trained SNN is always more efficient in terms of time-steps compared to the equivalent ANN-SNN conversion network, but not in terms of the number of spikes, in some cases. It will require a detailed investigation to determine if ANN-SNN methods used higher firing rates, whether they would be able to classify quickly as well, while incurring a lower number of spike/inference.

4.4. Complexity Reduction

Deep ANNs struggle to meet the demand of extraordinary computational requirements. SNNs can mitigate this effort by enabling efficient event-based computations. To compare the computational complexity of these two cases, we first need to understand the operation principle of both. An ANN operation for inferring the category of a particular input requires a single feed-forward pass per image. For the same task, the network must be evaluated over a number of time-steps in the spiking domain. If regular hardware is used for both ANN and SNN, then it is evident that SNN will have computation complexity in the order of hundreds or thousands more compared to an ANN. However, there are specialized hardwares that account for the event-based neural operation and "computes only when required" for inference. SNNs can potentially exploit such alternative mechanisms of network operation and carry out an inference operation in the spiking domain much more efficiently than an ANN. Also, for deep SNNs, we have observed the increase in sparsity as the network depth increases. Hence, the benefits from event-based neuromorphic hardware are expected to increase as the network depth increases.

An estimate of the actual energy consumption of SNNs and comparison with ANNs is outside the scope of this

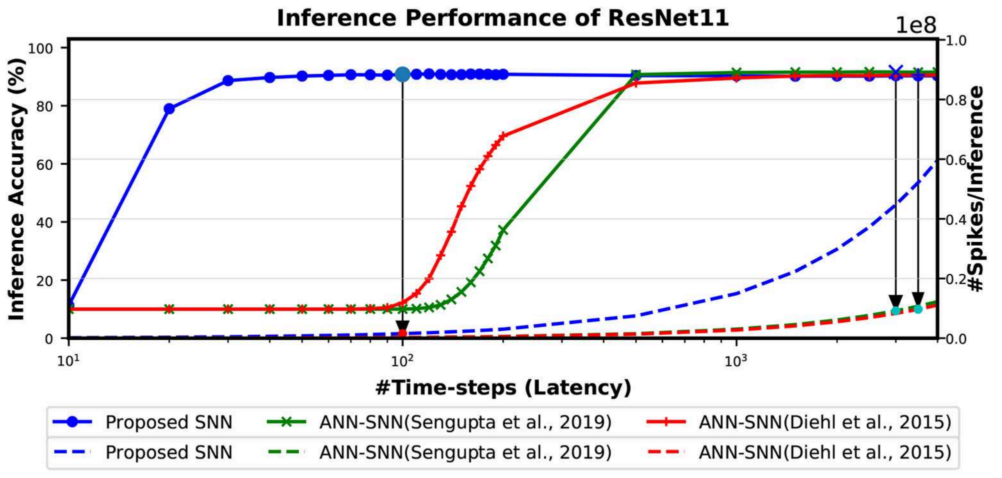


FIGURE 9 | The comparison of "accuracy vs. latency vs. #spikes/inference" for ResNet11 architecture. In this figure, the solid lines are representing inference accuracy while the dashed lines are representing #spikes/inference. The slope of #spikes/inference curve of the proposed SNN is larger than ANN-SNN converted networks. However, since proposed SNN requires much less time-steps for inference, the number of spikes required for one image inference is significantly lower compared to ANN-SNN. The required #time-steps and corresponding #spikes/inference are shown using highlighted points connected by arrows. Log scale is used for x-axis for easier viewing of the accuracy changes for lower number of time-steps.

TABLE 9 | Inference #time-steps and corresponding speedup comparison between SNN and ANN-SNN converted networks for benchmark datasets trained on different network models.

Dataset	Model	Timesteps			SNN Inference Speedup Compared to	
		SNN	ANN-SNN (Diehl et al., 2015)	ANN-SNN (Sengupta et al., 2019)	ANN-SNN (Diehl et al., 2015)	ANN-SNN (Sengupta et al., 2019)
MNIST	LeNet	50	180	200	3.6x	4x
				500		10x
SVHN	VGG7	100	500	1,600	5x	16x
			2,500	2,600	25x	26x
	ResNet7	100	500	400	5x	4x
			3,000	2,500	30x	25x
CIFAR-10	VGG9	100	500	800	5x	8x
			2,500	3,600	25x	36x
	ResNet9	100	800	600	8x	6x
			3,000	3,000	30x	30x
	ResNet11	100	3500	600	35x	6x
				3,000		30x

(For each network, the 1st row corresponds to iso-accuracy and the 2nd row corresponds to maximum-accuracy condition).

work. However, we can gain some insight by quantifying the computational energy consumption for a synaptic operation and comparing the number of synaptic operations being performed in the ANN vs. the SNN trained with our proposed algorithm and ANN-SNN converted network. We can estimate the number of synaptic operations per layer of a neural network from the structure for the convolutional and linear layers. In an ANN, a multiply-accumulate (MAC) computation is performed per synaptic operation. While a specialized SNN hardware would perform simply an accumulate computation (AC) per synaptic operation only if an incoming spike is received. Hence, the total number of AC operations in a SNN can be estimated by the layer-wise product and summation of the average neural spike count for a particular layer and the corresponding number of synaptic connections. We also have to multiply the #time-steps with the #AC operations to get total #AC operation for one inference. For example, assume that there are L layers each with N_l neurons, S_l synaptic connections and a_l average spiking activity where l is the layer number. Then, the total number of synaptic operations in a layer is $N_l \times S_l \times a_l$. The $N_l \times S_l$ is equal to the ANN (#MAC) operations of a particular layer. Therefore, the total number of synaptic operations in a layer of an SNN becomes $\#MAC_l \times a_l$. The total number of #AC operations required for an image inference is the sum of synaptic

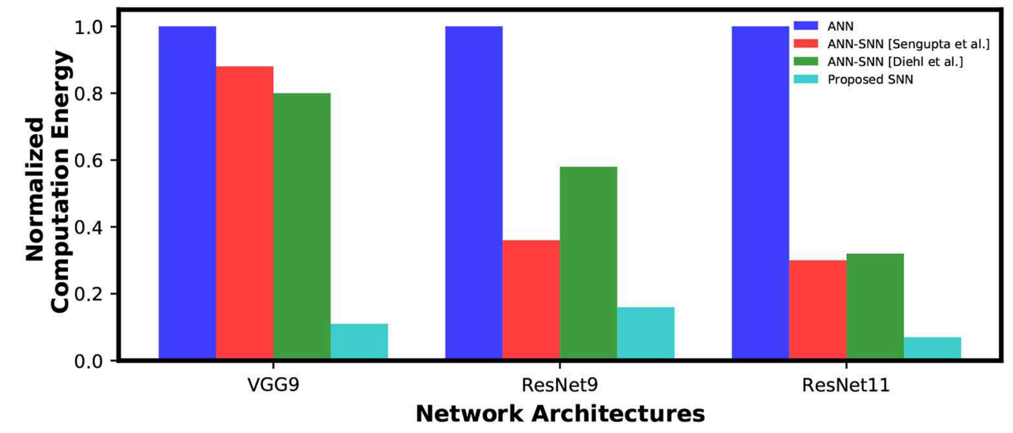


FIGURE 10 | Inference computation complexity comparison between ANN, ANN-SNN conversion and SNN trained with spike-based backpropagation. ANN computational complexity is considered as a baseline for normalization.

TABLE 10 | Iso-spike comparison for optimal condition.

Dataset	Model	Time-steps SNN	Time-steps ANN-SNN (Diehl et al., 2015)	Time-steps ANN-SNN (Sengupta et al., 2019)	Accuracy (%) SNN	Accuracy (%) ANN-SNN (Diehl et al., 2015)	Accuracy (%) ANN-SNN (Sengupta et al., 2019)
MNIST	LeNet	20	62	75	99.36	99.19	88.62
SVHN	VGG7	30	235	235	95.00	95.34	88.13
	ResNet7	30	200	200	95.06	95.63	95.48
CIFAR-10	VGG9	50	228	260	89.33	69.53	61.08
	ResNet9	50	390	490	89.52	89.51	90.06
	ResNet11	50	307	280	90.24	82.75	73.82

SNN time-steps corresponds to the latency to reach accuracy within ~1% of maximum accuracy. ANN-SNN time-steps corresponds to the latency required for same number of spike/inference as SNN to occur. SNN and ANN-SNN accuracies are accuracies corresponding to respective latency.

operations in all layers during the inference time-window. Hence, $\#AC/inference = (\sum_l (\#MAC_l \times a_l)) \times \#timesteps$. This formula is used for estimating both ANN-SNN AC operations and SNN AC operations per image inference. On the other hand, the number of ANN (MAC) operation per inference becomes simply, $\#MAC/inference = \sum_1^L (\#MAC_l)$. Based on this concept, we estimated the total number of MAC operations for ANN, and the total number of AC operations for direct-spike trained SNN and ANN-SNN converted network, for VGG9, ResNet9 and ResNet11. The ratio of ANN-SNN converted networks' (scheme1-scheme2) AC operations to direct-spike trained SNN AC operations to ANN MAC operations is (28.18–25.60):3.61:1 for VGG9 while the ratio is (11.67–18.42):5.06:1 for the ResNet9 and (9.6–10.16):2.09:1 for ResNet11 (for maximum accuracy condition).

However, a MAC operation usually consumes an order of magnitude more energy than an AC operation. For instance, according to Han et al. (2015), a 32-bit floating point MAC operation consumes 4.6 pJ and a 32-bit floating point AC operation consumes 0.9 pJ in 45 nm technology node. Hence, one synaptic operation in an ANN is equivalent to ~5 synaptic operations in a SNN. Moreover, 32-bit floating point computation can be replaced by fixed point computation using integer MAC and AC units without losing accuracy since the conversion is reported to be almost loss-less Lin et al. (2016). A 32-bit integer MAC consumes roughly 3.2 pJ while a 32-bit AC operation consumes only 0.1 pJ in 45nm process technology. Considering this fact, our calculations demonstrate that the SNNs trained using the proposed method will be 7.81x~7.10x and 8.87x more computationally energy-efficient for inference compared to the ANN-SNN converted networks and an ANN, respectively, for the VGG9 network architecture. We also gain 4.6x~4.87x(2.31x~3.64x) and 15.32x(6.32x) energy-efficiency, for the ResNet11(ResNet9) network, compared to the ANN-SNN converted networks and an ANN, respectively. The **Figure 10** shows the reduction in computation complexity for ANN-SNN conversions and SNN trained with the proposed methodology compared to ANNs.

It is worth noting here that as the sparsity of the spike signals increases with an increase in network depth in SNNs. Hence, the energy-efficiency is expected to increase almost exponentially in both ANN-SNN conversion network (Sengupta et al., 2019) and SNN trained with proposed methodology compared to an ANN implementation. The depth of network is the key factor for achieving a significant increase in the energy efficiency for SNNs in neuromorphic hardware. However, the computational efficiency does not perfectly align with the overall efficiency since the dominant energy consumption can be the memory traffic on von-Neumann computing hardware. The dataflows in asynchronous SNNs are less predictable and more complicated. Hence, a detailed study is required to estimate the overall efficiency of SNNs accurately.

4.5. Iso-Spike Comparison for Optimal Condition

In section 4.3, we observe that SNNs trained with proposed method achieve significant speed-up in both max-accuracy and iso-accuracy condition. However, in section 4.2.2, we found that the proposed method is in some cases (in an iso-accuracy condition) not more efficient than ANN-SNN conversions in terms of #spike/inference. The reason behind it is that an iso-accuracy condition may not be optimal for the SNNs trained with proposed method. In an iso-accuracy case, we have used max-accuracy latency (50 time-steps for MNIST and 100 time-steps for other networks) for direct-spike trained SNN, whereas most of the conversion networks used much less latency than the max-accuracy condition. In view of this, there is a need to determine the circumstances where our proposed method performs as well as or better than the SNN-ANN conversion methods on spike count, time steps, and accuracy. Consequently, in this section we analyze another interesting comparison.

In this analysis, we compare our proposed method and ANN-SNN conversion methods (Diehl et al., 2015; Sengupta et al., 2019) under the optimal condition at equal number of spikes. We define the "optimal #time-steps" for SNNs trained with our proposed method as the #time-steps required to reach within ~1% of peak accuracy (when the accuracy starts to saturate). Based on this definition, we observed that the optimal #time-steps for MNIST, SVHN, CIFAR10 networks are 20, 30, and 50, respectively. For this comparison, we recorded the achieved accuracy and #spike/inference of the SNNs trained with our proposed method for the corresponding optimal #time-steps. Then, we ran ANN-SNN networks for a length of time such that they use the similar number of spikes. In this iso-spike condition, we recorded the accuracy of the ANN-SNN networks (for both conversion methods) and the number of time-steps they require. The results are summarized in **Table 10**.

For comparatively shallower networks such as LeNet, VGG7 (VGG type) and ResNet7, ResNet9 (Residual type), the ANN-SNN conversion networks achieve as good as or slightly better accuracy at iso-spike condition compared to the SNNs trained with our proposed method. However, these ANN-SNN conversion networks require 3x-10x higher latency for inference. On the other hand, for deeper networks such as VGG9 and ResNet11, the ANN-SNN conversion networks achieve significantly lower accuracy compared to SNNs trained with our proposed method even with much higher latency. This trend indicates that for deeper networks, SNNs trained with our proposed method will be more energy-efficient than the conversion networks under an iso-spike condition.

5. CONCLUSION

In this work, we propose a spike-based backpropagation training methodology for popular deep SNN architectures. This methodology enables deep SNNs to achieve comparable classification accuracies on standard image recognition tasks. Our experiments show the effectiveness of the proposed learning strategy on deeper SNNs (7–11 layer VGG and ResNet network architectures) by achieving the best classification accuracies in MNIST, SVHN, and CIFAR-10 datasets among other networks trained with spike-based learning till date. The performance gap in terms of quality between ANN and SNN is substantially reduced by the application of our proposed methodology. Moreover, significant computational energy savings are expected when deep SNNs (trained with the proposed method) are employed on suitable neuromorphic hardware for the inference.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: MNIST, N-MNIST, SVHN, CIFAR-10 datasets. The source code is publicly released at "https://github.com/chan8972/Enabling_Spikebased_Backpropagation".

AUTHOR CONTRIBUTIONS

CL and SS implemented the algorithm and conducted the experiments. CL, SS, PP, GS, and KR discussed about the results and analysis, and wrote the manuscript.

FUNDING

This work was supported in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the National Science Foundation, Intel Corporation, the DoD Vannevar Bush Fellowship and the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

ACKNOWLEDGMENTS

This manuscript has been released as a Pre-Print at Lee et al. (2019a). We would like to thank Dr. Gerard (Rod) Rinkus for helpful comments.

REFERENCES

Ankit, A., Sengupta, A., Panda, P., and Roy, K. (2017). "Resparc: a reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017* (New York, NY), 1–6.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., et al. (2019). A solution to the learning dilemma for recurrent networks of spiking neurons. *bioRxiv [Preprint]*. doi: 10.1101/738385

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv [Preprint]*. arXiv:1308.3432.

Bohte, S. M., Kok, J. N., and La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 17–37. doi: 10.1016/S0925-2312(01)00658-0

Brader, J. M., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vision* 113, 54–66. doi: 10.1007/s11263-014-0788-3

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Dayan, P., and Abbott, L. F. (2001). *Theoretical Neuroscience, Vol. 806*. Cambridge, MA: MIT Press.

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9:99. doi: 10.3389/fncom.2015.00099

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)* (Killarney: IEEE), 1–8.

Diehl, P. U., Zarrella, G., Cassidy, A., Pedroni, B. U., and Neftci, E. (2016). "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)* (San Diego, CA: IEEE), 1–8.

Esser, S., Merolla, P., Arthur, J., Cassidy, A., Appuswamy, R., Andreopoulos, A., et al. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *arXiv [Preprint]*. Available online at: http://arxiv.org/abs/1603.08270

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., et al. (2013). Overview of the spinnaker system architecture. *IEEE Trans. Comput.* 62, 2454–2467. doi: 10.1109/TC.2012.142

Han, S., Pool, J., Tran, J., and Dally, W. (2015). "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems* (Montréal, QC), 1135–1143.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision* (Santiago), 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV), 770–778.

Hodgkin, A. L., and Huxley, A. F. (1952). Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *J. Physiol.* 116, 449–472. doi: 10.1113/jphysiol.1952.sp004717

Huh, D., and Sejnowski, T. J. (2018). "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems* (Montréal, QC), 1440–1450.

Hunsberger, E., and Eliasmith, C. (2015). Spiking deep networks with lif neurons. *arXiv [Preprint]*. arXiv:1510.08829.

Ioffe, S., and Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv [Preprint]*. arXiv:1502.03167.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Jin, Y., Li, P., and Zhang, W. (2018). Hybrid macro/micro level backpropagation for training deep spiking neural networks. *arXiv [Preprint]*. arXiv:1805.07866.

Kappel, D., Habenschuss, S., Legenstein, R., and Maass, W. (2015). Network plasticity as bayesian inference. *PLoS Comput. Biol.* 11:e1004485. doi: 10.1371/journal.pcbi.1004485

Kappel, D., Legenstein, R., Habenschuss, S., Hsieh, M., and Maass, W. (2018). A dynamic connectome supports the emergence of stable computational function of neural circuits through reward-based learning. *Eneuro* 5:ENEURO.0301-17.2018. doi: 10.1523/ENEURO.0301-17.2018

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. (2016). Stdp-based spiking deep neural networks for object recognition. *arXiv [Preprint]*. arXiv:1611.01421.

Kingma, D. P., and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv [Preprint]*. arXiv:1412.6980.

Krizhevsky, A., and Hinton, G. (2009). *Learning Multiple Layers of Features From Tiny Images*. Technical report, Citeseer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Adv. Neural. Inform. Process. Syst.* 25, 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lee, C., Panda, P., Srinivasan, G., and Roy, K. (2018). Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning. *Front. Neurosci.* 12:435. doi: 10.3389/fnins.2018.00435

Lee, C., Sarwar, S. S., and Roy, K. (2019a). Enabling spike-based backpropagation in state-of-the-art deep neural network architectures. *arXiv [Preprint]*. arXiv:1903.06379.

Lee, C., Srinivasan, G., Panda, P., and Roy, K. (2019b). Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity. *IEEE Trans. Cogn. Dev. Syst*. 11, 384–394. doi: 10.1109/TCDS.2018.2833071

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Front. Neurosci.* 10:508. doi: 10.3389/fnins.2016.00508

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128$\times$128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circuits* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Lillicrap, T. P., and Santoro, A. (2019). Backpropagation through time and the brain. *Curr. Opin. Neurobiol.* 55, 82–89. doi: 10.1016/j.conb.2019.01.011

Lin, D., Talathi, S., and Annapureddy, S. (2016). "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning* (New York, NY), 2849–2858.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Netw.* 10, 1659–1671. doi: 10.1016/S0893-6080(97)00011-7

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Mostafa, H. (2018). Supervised learning based on temporal coding in spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst*. 29, 3227–3235. doi: 10.1109/TNNLS.2017.2726060

Neftci, E. O., Augustine, C., Paul, S., and Detorakis, G. (2017). Event-driven random back-propagation: enabling neuromorphic deep learning machines. *Front. Neurosci.* 11:324. doi: 10.3389/fnins.2017.00324

Neftci, E. O., Pedroni, B. U., Joshi, S., Al-Shedivat, M., and Cauwenberghs, G. (2015). Unsupervised learning in synaptic sampling machines. *arXiv [Preprint]*. arXiv:1511.04484.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Vol. 2011*, 5.

Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* 9:437. doi: 10.3389/fnins.2015.00437

Panda, P., and Roy, K. (2016). "Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition," in *2016 International Joint Conference on Neural Networks (IJCNN)* (Vancouver, BC: IEEE), 299–306.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). "Automatic differentiation in pytorch," in *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques* (Long Beach, CA).

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11:682. doi: 10.3389/fnins.2017.00682

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). *Learning Internal Representations by Error Propagation.* Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Sarwar, S. S., Srinivasan, G., Han, B., Wijesinghe, P., Jaiswal, A., Panda, P., et al. (2018). Energy efficient neural computing: a study of cross-layer approximations. *IEEE J. Emerg. Sel. Top. Circuits Syst.* 8, 796–809. doi: 10.1109/JETCAS.2018.2835809

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: Vgg and residual architectures. *Front. Neurosci.* 13:95. doi: 10.3389/fnins.2019.00095

Shrestha, S. B., and Orchard, G. (2018). "Slayer: spike layer error reassignment in time," in *Advances in Neural Information Processing Systems* (Montréal, QC), 1412–1421.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529:484. doi: 10.1038/nature16961

Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv [Preprint].* arXiv:1409.1556.

Srinivasan, G., Panda, P., and Roy, K. (2018a). Spilinc: Spiking liquid-ensemble computing for unsupervised speech and image recognition. *Front. Neurosci.* 12:524. doi: 10.3389/fnins.2018.00524

Srinivasan, G., Panda, P., and Roy, K. (2018b). Stdp-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing. *ACM J. Emerg. Technol. Comput. Syst.* 14:44. doi: 10.1145/3266229

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1929–1958. doi: 10.5555/2627435.2670313

Tavanaei, A., and Maida, A. S. (2016). Bio-inspired spiking convolutional neural network using layer-wise sparse coding and stdp learning. *arXiv [Preprint].* arXiv:1611.03000.

Tavanaei, A., and Maida, A. S. (2017). "Multi-layer unsupervised learning in a spiking convolutional neural network," in *2017 International Joint Conference on Neural Networks (IJCNN)* (Anchorage, AK: IEEE), 2023–2030.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 1550–1560.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018a). Direct training for spiking neural networks: faster, larger, better. *arXiv [Preprint].* arXiv:1809.05793.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018b). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12:331. doi: 10.3389/fnins.2018.00331

Zhao, B., Ding, R., Chen, S., Linares-Barranco, B., and Tang, H. (2015). Feedforward categorization on aer motion events using cortex-like features in a spiking neural network. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1963–1978. doi: 10.1109/TNNLS.2014.2362542

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.

ORIGINAL RESEARCH
published: 12 May 2020
doi: 10.3389/fnins.2020.00425

A Spiking Neuron and Population Model Based on the Growth Transform Dynamical System

Ahana Gangopadhyay[1], *Darshit Mehta*[2] *and Shantanu Chakrabartty*[1*]

[1] *Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO, United States,*
[2] *Department of Biomedical Engineering, Washington University in St. Louis, St. Louis, MO, United States*

In neuromorphic engineering, neural populations are generally modeled in a bottom-up manner, where individual neuron models are connected through synapses to form large-scale spiking networks. Alternatively, a top-down approach treats the process of spike generation and neural representation of excitation in the context of minimizing some measure of network energy. However, these approaches usually define the energy functional in terms of some statistical measure of spiking activity (ex. firing rates), which does not allow independent control and optimization of neurodynamical parameters. In this paper, we introduce a new spiking neuron and population model where the dynamical and spiking responses of neurons can be derived directly from a network objective or energy functional of continuous-valued neural variables like the membrane potential. The key advantage of the model is that it allows for independent control over three neuro-dynamical properties: (a) control over the steady-state population dynamics that encodes the minimum of an exact network energy functional; (b) control over the shape of the action potentials generated by individual neurons in the network without affecting the network minimum; and (c) control over spiking statistics and transient population dynamics without affecting the network minimum or the shape of action potentials. At the core of the proposed model are different variants of Growth Transform dynamical systems that produce stable and interpretable population dynamics, irrespective of the network size and the type of neuronal connectivity (inhibitory or excitatory). In this paper, we present several examples where the proposed model has been configured to produce different types of single-neuron dynamics as well as population dynamics. In one such example, the network is shown to adapt such that it encodes the steady-state solution using a reduced number of spikes upon convergence to the optimal solution. In this paper, we use this network to construct a spiking associative memory that uses fewer spikes compared to conventional architectures, while maintaining high recall accuracy at high memory loads.

Keywords: spiking neuron model, growth transforms, energy-minimization, dynamical system, network model, neural dynamics, associative memory, adaptation

OPEN ACCESS

Edited by:
Runchun Mark Wang,
Western Sydney University, Australia

Reviewed by:
Hesham Mostafa,
Intel, United States
André van Schaik,
Western Sydney University, Australia

****Correspondence:***
Shantanu Chakrabartty
shantanu@wustl.edu

Specialty section:
This article was submitted to
Neuromorphic Engineering,
a section of the journal
Frontiers in Neuroscience

Received: *17 February 2020*
Accepted: *07 April 2020*
Published: *12 May 2020*

Citation:
Gangopadhyay A, Mehta D and Chakrabartty S (2020) A Spiking Neuron and Population Model Based on the Growth Transform Dynamical System. Front. Neurosci. 14:425. doi: 10.3389/fnins.2020.00425

1. INTRODUCTION

Spiking neural networks that emulate neural ensembles have been studied extensively within the context of dynamical systems (Izhikevich, 2007), and modeled as a set of differential equations that govern the temporal evolution of its state variables. For a single neuron, the state variables are usually its membrane potential and the conductances of ion channels that mediate changes in the membrane potential via flux of ions across the cell membrane. A vast body of literature, ranging from the classical Hodgkin-Huxley model (Hodgkin and Huxley, 1952), FitzHugh-Nagumo model (FitzHugh, 1961), Izhikevich model (Izhikevich, 2003) to simpler integrate-and-fire models (Abbott, 1999), treats the problem of single-cell excitability at various levels of detail and biophysical plausibility. Individual neuron models are then connected through synapses, bottom-up, to form large-scale spiking neural networks.

An alternative to this bottom-up approach is a top-down approach that treats the process of spike generation and neural representation of excitation in the context of minimizing some measure of network energy. The rationale for this approach is that physical processes occurring in nature have a tendency to self-optimize toward a minimum-energy state. This principle has been used to design neuromorphic systems where the state of a neuron in the network is assumed to be either binary in nature (spiking or not spiking) (Jonke et al., 2016), or replaced by its average firing rate (Nakano et al., 2015). However, in all of these approaches, the energy functionals have been defined with respect to some statistical measure of neural activity, for example spike rates, instead of continuous-valued neuronal variables like the membrane potential. As a result in these models, it is difficult to independently control different neuro-dynamical parameters, for example the shape of the action-potential, bursting activity or adaptation in neural activity, without affecting the network solution.

In Gangopadhyay and Chakrabartty (2018), we proposed a model of a Growth Transform (GT) neuron which reconciled the bottom-up and top-down approaches such that the dynamical and spiking responses were derived directly from a network objective or an energy functional. Each neuron in the network implements an asynchronous mapping based on polynomial Growth Transforms, which is a fixed-point algorithm for optimizing polynomial functions under linear and/or bound constraints (Baum and Sell, 1968; Gangopadhyay et al., 2017). It was shown in Gangopadhyay and Chakrabartty (2018) that a network of GT neurons can solve binary classification tasks while producing stable and unique neural dynamics (for example, noise-shaping, spiking and bursting) that could be interpreted using a classification margin. However, in the previous formulation, all of these neuro-dynamical properties were directly encoded into the network energy function. As a result, the formulation did not allow independent control and optimization of different neuro-dynamics. In this paper, we address these limitations by proposing a novel GT spiking neuron and population model, along with a neuromorphic framework, according to the following steps:

- We first remap the synaptic interactions in a standard spiking neural network in a manner that the solution (steady-state attractor) could be encoded as a first-order condition of an optimization problem. We show that this network objective function or energy functional can be interpreted as the total extrinsic power required by the remapped network to operate, and hence a metric to be minimized.
- We then introduce a dynamical system model based on Growth Transforms that evolves the network toward this steady-state attractor under the specified constraints. The use of Growth Transforms ensures that the neuronal states (membrane potentials) involved in the optimization are always bounded and that each step in the evolution is guaranteed to reduce the network energy.
- We then show how gradient discontinuity in the network energy functional can be used to modulate the shape of the action potential while maintaining the local convexity and the location of the steady-state attractor.
- Finally, we use the properties of Growth Transforms to generalize the model to a continuous-time dynamical system. The formulation will then allow for modulating the spiking and the population dynamics across the network without affecting network convergence toward the steady-state attractor.

We show that the proposed framework can be used to implement a network of coupled neurons that can exhibit memory, global adaptation, and other interesting population dynamics under different initial conditions and based on different network states. We also illustrate how decoupling transient spiking dynamics from the network solution and spike-shapes could be beneficial by using the model to design a spiking associative memory network that can recall a large number of patterns with high accuracy while using fewer spikes than traditional associative memory networks. This paper is also accompanied by a publicly available software implementing the proposed model (Mehta et al., 2019) using MATLAB$^{©}$. Users can experiment with different inputs and network parameters to explore and create other unique dynamics than what has been reported in this paper. In the future, we envision that the model could be extended to incorporate spike-based learning within an energy-minimization framework similar to the framework used in traditional machine learning models (LeCun et al., 2006). This could be instrumental in bridging the gap between neuromorphic algorithms and traditional energy-based machine learning models.

2. METHODS

In this section, we present the network energy functional by remapping the synaptic interactions of a standard spiking neural network and then propose a Growth Transform based dynamical system for minimizing this objective. For the rest of the paper, we will follow the mathematical notations as summarized below.

	Notations
x	Real scalar variable
$\boldsymbol{x}$	Real-valued vector with x_i as its i-th element
$\boldsymbol{X}$	Real-valued matrix with X_{ij} as the element at the i-th row and the j-th column
$x_{i,n}$	i-th element of real-valued vector $\boldsymbol{x}$ where $n = 1, 2, ..$ denotes a discrete time step
$x_i(t)$	i-th element of real-valued vector $\boldsymbol{x}$ at time t
$x_i[n]$	a sequence of scalar variables $x_{i,p}$ where $p = n, n-1, ..$
$\mathbb{E}_N\left(x_i[n]\right)$	Empirical expectation of a sequence $x_i[n]$ estimated over a window of size N, i.e., $\frac{1}{N}\sum_{p=n-N+1}^{n} x_{i,p}$
$\bar{x}_i[n]$	Empirical expectation estimated over an asymptotically infinite window, i.e., $\lim_{N\to\infty} \mathbb{E}_N\left(x_i[n]\right)$
$\bar{\boldsymbol{x}}[n]$	Real-valued vector with $\bar{x}_i[n]$ as its i-th element
$\mathbb{E}_T[x_i(t)]$	Empirical expectation of $x_i(t)$ over a time-interval $[t-T, t]$, i.e., $\frac{1}{T}\int_{t-T}^{t} x_i(t')dt'$
$\mathbb{R}^M$	Vector space spanned by M-dimensional real vectors
$\lvert x \rvert$	Absolute value of a scalar
$\lvert\lvert \boldsymbol{x} \rvert\rvert_p$	l_p-norm of an M-dimensional vector, defined as $(\sum_{i=1}^{M} \lvert x_i \rvert^p)^{1/p}$
$\boldsymbol{x}^T$	Transpose of the vector $\boldsymbol{x}$
$\boldsymbol{x}.\boldsymbol{y}$	Inner product between the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
$\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}$	Gradient vector $[\frac{\partial \mathcal{H}}{\partial x_1}, \frac{\partial \mathcal{H}}{\partial x_2}, ..., \frac{\partial \mathcal{H}}{\partial x_M}]^T$

2.1. Remapping Synaptic Interactions in a Standard Spiking Network

In generalized threshold models like the Spike Response Model (Gerstner and Kistler, 2002), the membrane voltage is given using response kernels that accurately model the post-synaptic responses due to pre-synaptic input spikes, external driving currents and the shape of the spike - the latter term being also used to model refractoriness. However, in simpler adaptations of spiking neuron models, the spike shape is often disregarded, and the membrane potentials are written as simple linear post-synaptic integrations of input spikes and external currents (Cassidy et al., 2013; Davies et al., 2018). We consider a similar model where $v_i \in \mathbb{R}$ represents an inter-cellular membrane potential corresponding to neuron i in a network of M neurons. The i-th neuron receives spikes from the j-th neuron that are modulated by a synapse whose strength or weight is denoted by $W_{ij} \in \mathbb{R}$. Assuming that the synaptic weights are constant, the following discrete-time temporal equation governs the dynamics when the membrane potential increases (Soula et al., 2006; Cessac, 2011)

$$v_{i,n+1} = \gamma v_{i,n} + \sum_{j=1}^{M} W_{ij}\Psi(v_{j,n}) + y_{i,n}, \quad \forall i = 1, ..., M, \quad (1)$$

where $v_{i,n} \equiv v_i(n\Delta t)$ and $v_{i,n+1} \equiv v_i\big((n+1)\Delta t\big)$, Δt being the time increment between two time-steps. $y_{i,n}$ represents the depolarization due to an external stimulus that can be viewed as $y_{i,n} = R_{mi}I_{i,n}$, where $I_{i,n} \in \mathbb{R}$ is the current stimulus at the n-th time-step and $R_{mi} \in \mathbb{R}$ is the membrane resistance of the i-th neuron. Here, $0 \leq \gamma \leq 1$ denotes the leakage factor and $\Psi(.)$ denotes a simple spiking function that is positive only when the voltage $v_{j,n}$ exceeds a threshold and 0 otherwise. Note that in (1), the filter $\Psi(.)$ implicitly depends on the pre-synaptic spike-times through the pre-synaptic membrane voltage $v_{j,n}$. Such a spiking neural network model is shown in **Figure 1A**.

We further enforce that the membrane potentials are bounded by v_c as

$$|v_{i,n}| \leq v_c, \quad \forall i = 1, ..., M, \quad \forall n. \quad (2)$$

Note that in biological neural networks, the membrane potentials are also bounded (Wright, 2004).

If $\Psi(.)$ was a smooth function of the membrane potential, $v_{i,n}$ would track the net input at every instant. For a non-smooth $\Psi(.)$, however, we make the additional assumption that the temporal expectation of $v_{i,n}$ encodes the net input over a sufficiently large time-window. Considering $\bar{y}_i[n]$ to be the empirical expectation of the external input estimated at the n-th time-window, and under the bound constraints outlined in (2), we can get the following relation (justification in **Appendix A**)

$$(1-\gamma)\bar{v}_i[n] = \sum_{j=1}^{M} W_{ij}\bar{\Psi}_j[n] + \bar{y}_i[n], \quad (3)$$

where $\bar{\Psi}_j[n] = \lim_{N\to\infty} \sum_{p=n-N+1}^{n} \Psi(v_{j,p})$. To reduce notational clutter, we will re-write (3) in a matrix form as

$$(1-\gamma)\bar{\boldsymbol{v}}[n] = \mathbf{W}\bar{\boldsymbol{\Psi}}[n] + \bar{\boldsymbol{y}}[n], \quad (4)$$

where $\bar{\boldsymbol{v}}[n] \in \mathbb{R}^M$ is the vector of mean membrane potentials for a network of M neurons, $\mathbf{W} \in \mathbb{R}^M \times \mathbb{R}^M$ is the synaptic weight matrix for the network, $\bar{\boldsymbol{y}}[n] \in \mathbb{R}^M$ is the vector of mean external inputs for the n-th time-window and $\bar{\boldsymbol{\Psi}}[n] = \left[\bar{\Psi}_1[n], \bar{\Psi}_2[n], ..., \bar{\Psi}_M[n]\right]^T$ is the vector of mean spike currents. As $\Psi(.)$ is a non-linear function of the membrane potential, it is difficult to derive an exact network energy functional corresponding to (4). However, if we assume that the synaptic weight matrix $\boldsymbol{W}$ is invertible, we can re-write Equation (4) as

$$\bar{\boldsymbol{\Psi}}[n] = (1-\gamma)\mathbf{W}^{-1}\bar{\boldsymbol{v}}[n] - \mathbf{W}^{-1}\bar{\boldsymbol{y}}[n], \quad \text{or} \quad (5)$$

$$\bar{\boldsymbol{\Psi}}[n] = -\mathbf{Q}\bar{\boldsymbol{v}}[n] + \bar{\boldsymbol{b}}[n], \quad (6)$$

where $\mathbf{Q} = -(1-\gamma)\mathbf{W}^{-1}$, and $\bar{\boldsymbol{b}}[n] = -\mathbf{W}^{-1}\bar{\boldsymbol{y}}[n]$ is the effective external current stimulus. Note that in case $\boldsymbol{W}$ is not invertible, $\boldsymbol{W}^{-1}$ could represent a pseudo-inverse. For the i-th neuron, (6) is equivalent to

$$\bar{\Psi}_i[n] = -\sum_{j=1}^{M} Q_{ij}\bar{v}_j[n] + \bar{b}_i[n], \quad (7)$$

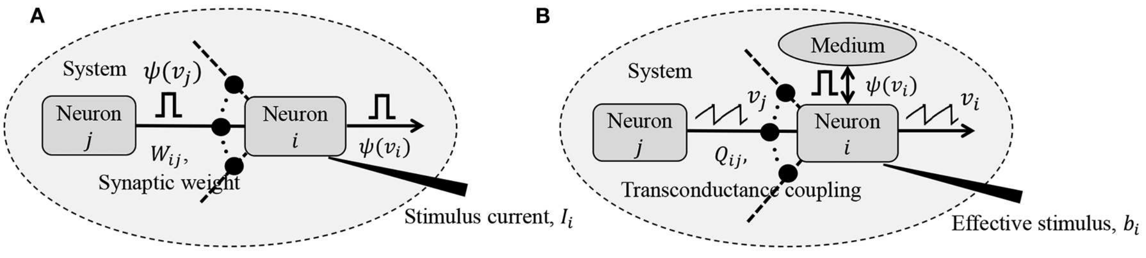


FIGURE 1 | (A) Simple but general model of a spiking neural network; **(B)** Compartmental network model obtained after remapping synaptic interactions.

subject to the bound constraint $|v_{i,n}| \leq v_c\ \forall i, n$. In the subsequent sections, we show that (7) can be viewed as the first-order condition of the following network objective function or energy functional

$$\min_{|v_i|\leq v_c\ \forall i} \mathcal{H}(\{v_i\}) = \min_{|v_i|\leq v_c\ \forall i} \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} Q_{ij}v_jv_i - \sum_{i=1}^{M} b_iv_i + \sum_{i=1}^{M}\int_{-\infty}^{v_i} \Psi(v)dv. \quad (8)$$

The network energy functional $\mathcal{H}(.)$ in (8) also admits a physical interpretation, as shown in **Figure 1B**. Each neuron i receives a voltage input from the neuron j through a synapse that can be modeled by a transconductance Q_{ij}. The neuron i also receives an electrical current stimulus b_i and exchanges a voltage-dependent ionic-current with its medium, denoted by $\Psi(v_i)$. Then, the function $\mathcal{H}(.)$ in Equation (8) represents the extrinsic (or metabolic) power supplied to the network, comprising the following three components: (a) Power dissipation due to coupling between neurons; (b) Power injected to or extracted from the system as a result of external stimulation; and (c) Power dissipated due to neural responses.

2.2. Neuron Model Using the Growth Transform Dynamical System

In order to solve the energy minimization problem given in (8) under the constraints given in (2), we first propose a dynamical system based on polynomial Growth Transforms. We also show how the dynamical system evolves over time to satisfy (7) as a first-order condition.

Growth Transforms are multiplicative updates derived from the well-known Baum-Eagon inequality (Baum and Sell, 1968; Chatterjee and Chakrabartty, 2018) that optimize a Lipschitz continuous cost function under linear and/or bound constraints on the optimization variables. Each neuron in the network implements a continuous mapping based on Growth Transforms, ensuring that the network evolves over time to reach an optimal solution of the energy functional within the constraint manifold. The summary of a discrete-time Growth Transform dynamical system is presented in **Table 1** and the detailed derivation is provided in **Appendix B**.

TABLE 1 | Discrete-time Growth Transform dynamical system (Proof in **Appendix B**).

Proposition I. Let $\mathcal{H}(\{v_i\}) : \mathbb{R}^M \to \mathbb{R}$ be a function of $v_i,\ i = 1, ..., M$ with bounded partial derivatives, and let $\lambda > |\frac{\partial \mathcal{H}}{\partial v_{i,n}}|\ \forall i, n$, be a parameter. Then for $|v_{i,0}| \leq v_c\ \forall i$, the discrete-time dynamical system

$$v_{i,n+1} \leftarrow v_C \frac{-\frac{\partial \mathcal{H}}{\partial v_{i,n}} v_C + \lambda v_{i,n}}{-\frac{\partial \mathcal{H}}{\partial v_{i,n}} v_{i,n} + \lambda v_C},\ i = 1, ..., M \quad (9)$$

satisfies the following criteria for all time-indices n:

(a) $|v_{i,n}| \leq v_C\ \forall i$; (10)

(b) $\mathcal{H}(\{v_{i,n+1}\}) \leq \mathcal{H}(\{v_{i,n}\})$ in domains where $\frac{\partial \mathcal{H}}{\partial v_{i,n}}$ is continuous; and (11)

(c) $\lim_{N\to\infty} \left(\mathbb{E}_N\left(z_i[n]\right)\right) \to 0\ \forall i, n$; where $z_{i,n} = (v_c^2 - v_{i,n}v_{i,n+1})\frac{\partial \mathcal{H}}{\partial v_{i,n}}$. (12)

2.2.1. Growth Transform Spiking Neuron Model

Considering the n-th iteration of the update equation in (9) as the n-th time-step for the neuron i, we can rewrite (9) in terms of the objective function for the neuron model presented in (8), as given below

$$v_{i,n+1} \leftarrow v_c \frac{-\frac{\partial \mathcal{H}}{\partial v_{i,n}} v_c + \lambda v_{i,n}}{-\frac{\partial \mathcal{H}}{\partial v_{i,n}} v_{i,n} + \lambda v_c},\ \ i = 1, ..., M, \quad (13)$$

where

$$\frac{\partial \mathcal{H}}{\partial v_{i,n}} = \sum_{j=1}^{M} Q_{ij}v_{j,n} - b_{i,n} + \Psi(v_{i,n}). \quad (14)$$

Then asymptotically from (1), and as shown in **Appendix B**, we have

$$\lim_{N\to\infty} \left(\mathbb{E}_N\left(z_i[n]\right)\right) \to 0\ \ \forall i, n, \quad (15)$$

where $z_{i,n} = (v_c^2 - v_{i,n}v_{i,n+1})\frac{\partial \mathcal{H}}{\partial v_{i,n}}$. We first show the dynamics resulting from (13) for a trivial barrier function $\Psi(.) = 0$. Since $\mathcal{H}(.)$ is a smooth function in this case, the neural variables $v_{i,n}$ converge to a local minimum, such that

$$\lim_{n\to\infty} v_{i,n} = v_i^*. \quad (16)$$

Therefore, (15) can be written as

$$(v_c^2 - v_i^{*2})\frac{\partial \mathcal{H}}{\partial v_{i,n}}\bigg|_{v_i^*} \to 0. \quad (17)$$

Thus as long as $|v_i^*| < v_c$, the gradient term goes to zero, ensuring that the dynamical system converges to the optimal solution within the domain defined by the bound constraints.

The dynamical system presented in (9) ensures that the steady-state neural responses $|v_i^*| \leq v_c \ \forall i$. In the absence of the barrier term, the membrane potentials can converge to any value between $-v_c$ and $+v_c$ based on the effective inputs to individual neurons. **Figure 2A** illustrates this for 2 different neurons where **Q** is an identity matrix. For the sake of simplicity, we have considered the membrane potentials to be normalized in all the experiments in this paper (i.e., $v_c = 1 \ V$), and $0 \ V$ as the threshold voltage. Here v_1^* is hyperpolarized due to a negative stimulus, and v_2^* is depolarized beyond the threshold. **Figure 2B** shows the corresponding energy contours, where the steady-state neural responses encode the optimal solution of the energy function. We next show how this framework can be extended to a spiking neuron model when the trans-membrane current in the compartmental model described in (8) is approximated by a discontinuous $\Psi(.)$. In general, the penalty function $R(v_i) = \int_{-\infty}^{v_i} \Psi(v)dv$ is chosen to be convex, where $R(v_i > 0 \ V) > 0 \ W$ and $R(v_i \leq 0 \ V) = 0 \ W$. **Figure 2C** shows one such form that has a gradient discontinuity at a threshold ($v_i = 0 \ V$) at which the neuron generates an action potential. The corresponding $\Psi(.)$, also shown in **Figure 2C**, is given by

$$\Psi(v_{i,n}) = \begin{cases} I_\Psi \ A \ ; \ v_{i,n} > 0 \ V \\ 0 \ A \quad ; \ v_{i,n} \leq 0 \ V. \end{cases} \tag{18}$$

When there is no external stimulus b_i, the neuron response converges to $v_i^* = 0 \ V$ as in the non-spiking case, as illustrated in **Figure 2D** for a single neuron without any synaptic connections. When a positive stimulus b_i is applied, the optimal solution for v_i, indicated by v_i^*, shifts upward to a level that is a function of the stimulus magnitude, also shown in **Figure 2D**. However, a penalty term $R(v_i)$ of the form as described above works as a barrier function, penalizing the energy functional whenever v_i exceeds the threshold, thereby forcing v_i to reset below the threshold. The stimulus and the barrier function therefore introduce opposing tendencies, making v_i oscillate back and forth around the discontinuity (which, in our case, coincides with the threshold) as long as the stimulus is present. Thus when $\Psi(.)$ is introduced, the potential $v_{i,n}$ switches when $\Psi(v_{i,n}) > 0 \ A$ or only when $v_{i,n} > 0 \ V$. However, the dynamics of $v_{i,n}$ remains unaffected for $v_{i,n} < 0 \ V$. During the brief period when $v_{i,n} > 0 \ V$, we assume that the neuron enters into a runaway state leading to a voltage spike. The composite spike signal $s_{i,n}$, shown in **Figure 2E**, is then treated as a combination of the sub-threshold and supra-threshold responses and is given by

$$s_{i,n} = v_{i,n} + C\Psi(v_{i,n}), \tag{19}$$

where the trans-impedance parameter $C > 0 \ \Omega$ determines the magnitude of the spike. Note that in the current version, the proposed model does not explicitly model the runaway process that leads to the spike, unlike other neuron models (Hodgkin and Huxley, 1952; FitzHugh, 1961; Izhikevich, 2003). However, it does incorporate the hyperpolarization part of the spike as a result of v_i oscillating around the gradient discontinuity. Thus a refractory period is automatically incorporated in between two spikes.

In order to show the effect of $\Psi(.)$ on the nature of the solution, we plot in **Figures 2F,G** the neural responses and contour plots for the 2-neuron network in **Figures 2A,B**, for the same set of inputs, this time considering the case when the barrier function is present. The penalty function produces a barrier at the thresholds, which are indicated by red dashed lines, transforming the time-evolution of s_2 into a digital, spiking mode, where the firing rate is determined by the extent to which the neuron is depolarized. It can be seen from the neural trajectories in **Figure 2G** and from (8) that $\Psi(.) > 0$ behaves as a Lagrange parameter corresponding to the spiking threshold constraint $v_{i,n} < 0$.

In **Appendix C**, we outline how, for non-pathological cases, it can be shown from (12) that for spiking neurons or for neurons whose membrane potentials $v_{i,n} > -v_c \ \forall n$,

$$\lim_{N\to\infty} \left(\mathbb{E}_N \left(\frac{\partial \mathcal{H}}{\partial v_i}[n] \right) \right) = 0, \tag{20}$$

This implies that asymptotically the network exhibits limit-cycles about a single attractor or a fixed-point such that the time-expectations of its state variables encode this optimal solution. A similar stochastic first-order framework was used in Gore and Chakrabartty (2010) to derive a dynamical system corresponding to $\Sigma\Delta$ modulation for tracking low-dimensional manifolds embedded in high-dimensional analog signal spaces. Combining (14) and (20), we have

$$\sum_{j=1}^{M} Q_{ij}\bar{v}_j[n] - \bar{b}_i[n] + \bar{\Psi}_i[n] = 0, \tag{21}$$

where $\bar{\Psi}_i[n] = \lim_{N\to\infty} \frac{1}{N} \sum_{p=n-N+1}^{n} \Psi(v_{i,p})$. Rearranging the terms in (21), we obtain (7).

The penalty function $R(v_i) = \int_{-\infty}^{v_i} \Psi(v)dv$ in the network energy functional in effect models the power dissipation due to spiking activity. For the form of $R(.)$ chosen in this paper, the power dissipation due to spiking is taken to be zero below the threshold, and increases linearly above threshold. A plot of the composite spike signal for a ramp input for the spiking neuron model is presented in **Figure 2H**. As $v_{i,n}$ exceeds the threshold for a positive stimulus, the neuron enters a spiking regime and the firing rate increases with the input, whereas the sub-threshold response is similar to the non-spiking case. **Figure 2I** shows the tuning curve for the neuron as the input stimulus is increased. It is nearly sigmoidal in shape and shows how the firing rate reaches a saturation level for relatively high inputs. The proposed spiking neuron model based on the discrete-time Growth Transform dynamical system is summarized in **Table 2**.

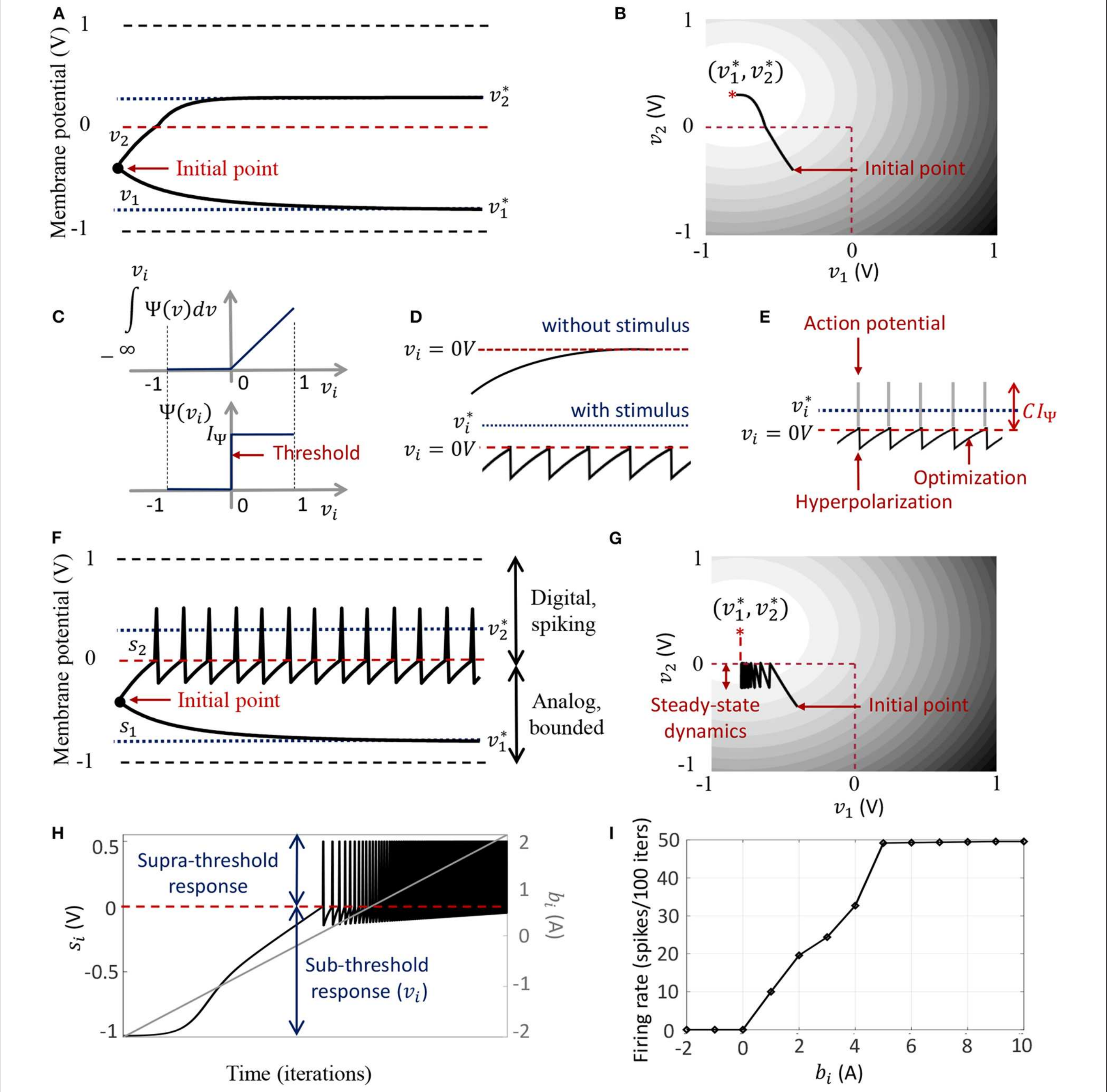


FIGURE 2 | (A) Bounded dynamics in a 2-neuron network in absence of the barrier function; **(B)** Corresponding contour plot showing convergence of the membrane potentials in the presence of external stimulus; **(C)** The function $\int \Psi(.)dv$ and its derivative $\Psi(.)$ used in this paper for the spiking neuron model; **(D)** Time-evolution of the membrane potential v_i of a single neuron in the spiking model in the absence and presence of external stimulus; **(E)** The composite signal upon addition of spikes when v_i crosses the threshold; **(F)** Bounded and spiking dynamics in the same 2-neuron network in presence of barrier function; **(G)** Corresponding contour plot showing steady-state dynamics of the membrane potentials in the presence of external stimulus; **(H)** Plot of composite spike signal s_i of the spiking neuron model when the external current stimulus is increased; **(I)** Input-output characteristics for the spiking neuron model.

2.2.2. Encoding Stimuli as a Combination of Sub-threshold and Supra-Threshold Dynamics

As explained previously, the penalty term $R(v_i)$ of the form presented above works analogous to a barrier function, penalizing the energy functional whenever $v_{i,n}$ exceeds the threshold. This transforms the time-evolution of $v_{i,n}$ into a spiking mode above the threshold, while keeping the sub-threshold dynamics similar to the non-spiking case. The Growth Transform dynamical system ensures that the membrane potentials are bounded, thereby implementing a squashing

TABLE 2 | Discrete-time GT spiking neuron model.

For a network of M neurons with state variables $\boldsymbol{v} = \{v_i\} \in \mathbb{R}^M$, where the trans-conductance coupling matrix is denoted by $\mathbf{Q} = \{Q_{ij}\} \in \mathbb{R}^M \times \mathbb{R}^M$ and the external stimulus vector is denoted by $\boldsymbol{b} = \{b_i\} \in \mathbb{R}^M$, the time-evolution of the network under bound constraints on the state variables $|v_{i,n}| \leq v_c$ for all time-indices n, is governed by the following discrete-time updates:

$$v_{i,n+1} \leftarrow v_C \frac{-\frac{\partial \mathcal{H}}{\partial v_{i,n}} v_C + \lambda v_{i,n}}{-\frac{\partial \mathcal{H}}{\partial v_{i,n}} v_{i,n} + \lambda v_C}, \; i = 1, ..., M, \tag{22}$$

where

$$\frac{\partial \mathcal{H}}{\partial v_{i,n}} = \sum_{j=1}^{M} Q_{ij} v_{j,n} - b_{i,n} + \Psi(v_{i,n});$$

$$\Psi(v_{i,n}) = \begin{cases} I_\Psi \, A \;; \; v_{i,n} > 0\, V \\ 0\, A \;; \; v_{i,n} \leq 0\, V \end{cases};$$

λ is a fixed current parameter such that $\lambda > |\frac{\partial \mathcal{H}}{\partial v_{i,n}}| \; \forall i, n$.

The composite spike response of the i-th neuron at time-step n is given by

$$s_{i,n} = v_{i,n} + C\Psi(v_{i,n}),$$

where the trans-impedance parameter $C > 0\ \Omega$ determines the magnitude of each spike.

(compressive) function on the neural responses. We now show how the proposed model encodes external stimulus as a combination of spiking and bounded dynamics. In the steady-state, from (21) we can write

$$\bar{\Psi}_i[n] = \bar{b}_i[n] - \sum_{j=1}^{M} Q_{ij} \bar{v}_j[n]. \tag{23}$$

Thus the average spiking activity of the i-th neuron encodes the error between the average input and the weighted sum of membrane potentials. For a single, uncoupled neuron where

$$Q_{ij} = \begin{cases} Q_0\ \Omega^{-1} \;; \; \forall i = j \\ 0\ \Omega^{-1} \;; \; \forall i \neq j \end{cases}, \tag{24}$$

we have

$$\bar{\Psi}_i[n] + Q_0 \bar{v}_i[n] = \bar{b}_i[n]. \tag{25}$$

Multiplying (25) on both sides by $C\ \Omega$, where we have chosen $C = \frac{1}{Q_0}$, we have

$$C\bar{\Psi}_i[n] + \bar{v}_i[n] = C\bar{b}_i[n] \tag{26}$$

$$\text{or,} \; s_i[n] = C\bar{b}_i[n], \tag{27}$$

where we have used the relation (19). Equation (27) indicates that through a suitable choice of the trans-impedance parameter C, the sum of sub-threshold and supra-threshold responses encodes the external input to the neuron. This is also the rationale behind adding a spike to the sub-threshold response $v_{i,n}$, as illustrated in **Figure 2E**, to yield the composite neural response. If $Q_0 = 0\ \Omega^{-1}$, we similarly have

$$\bar{\Psi}_i[n] = \bar{b}_i[n], \tag{28}$$

where the average spiking activity tracks the stimulus. Thus, by defining the coupling matrix in various ways, we can obtain different encoding schemes for the network.

2.3. From Neuron to Network: Geometric Interpretation of Network Dynamics

The remapping from standard coupled conditions of a spiking neural network to the proposed formulation admits a geometric interpretation of neural dynamics. Similar to the network coding framework presented in Gangopadhyay and Chakrabartty (2018), we show in this section how the activity of individual neurons in a network can be visualized with respect to a network hyper-plane. This geometric interpretation can then be used to understand network dynamics in response to different stimuli.

Like a Hessian, if we assume that the matrix $\mathbf{Q}$ is positive-definite about a local attractor, there exists a set of vectors $\boldsymbol{x}_i \in \mathbb{R}^D, \; i = 1, ..., M$ such that each of the elements Q_{ij} can be written as an inner product between two vectors as $Q_{ij} = \boldsymbol{x}_i . \boldsymbol{x}_j, \; 1 \leq i, j \leq M$. This is similar to kernel methods that compute similarity functions between pairs of vectors in the context of support vector machines (Chakrabartty and Cauwenberghs, 2007). This associates the i-th neuron in the network with a vector $\boldsymbol{x}_i$, mapping it onto an abstract metric space $\mathbb{R}^D$ and essentially providing an alternate geometric representation of the neural network. From (21), the spiking activity of the i-th neuron for the n-th time-window can then be represented as

$$\begin{aligned} \bar{\Psi}_i[n] &= -\sum_{j=1}^{M} Q_{ij} \bar{v}_j[n] + \bar{b}_i[n] \\ &= \sum_{j=1}^{M} -(\boldsymbol{x}_i . \boldsymbol{x}_j) \bar{v}_j[n] + \bar{b}_i[n] \\ &= \boldsymbol{w}_n . \boldsymbol{x}_i + \bar{b}_i[n], \end{aligned} \tag{29}$$

where

$$\boldsymbol{w}_n = -\sum_{j=1}^{M} \boldsymbol{x}_j \bar{v}_j[n]. \tag{30}$$

$\bar{\Psi}$ therefore represents the distance of the vector $\boldsymbol{x}_i$ from a network hyperplane in the D-dimensional vector space, which is parameterized by the weight vector $\boldsymbol{w}_n$ and offset $\bar{b}_i[n]$. When a stimulus $\bar{b}_i[n]$ is applied, the hyperplane shifts, leading to a stimulus-specific value of this distance for each neuron that is also dependent on the network configuration $\mathbf{Q}$. Hence, $\bar{\Psi}(.)$ is denoted as a "network variable," that signifies how the response of each neuron is connected to the rest of the network. Note that we can also write the elements of the coupling matrix in a kernelized form as $Q_{ij} = K(\boldsymbol{x}_i).K(\boldsymbol{x}_j)$, where $K(.)$ is a non-linear transformation function, defining a non-linear boundary for each neuron. This idea of a dynamic and stimulus-specific hyperplane can offer intuitive interpretations about several population dynamics reported in literature and have been elaborated on in section 3.

2.4. Complete Continuous-Time Model of the Growth Transform Neuron

Single neurons show a vast repertoire of response characteristics and dynamical properties that lend richness to their computational properties at the network level. Izhikevich

(2004) provides an extensive review of different spiking neuron models and their ability to produce the different dynamics observed in biology. In this section, we extend the proposed model into a continuous-time dynamical system, which enables us to reproduce a vast majority of such dynamics and also allows us to explore interesting properties in coupled networks. In Appendix D, we derive the continuous-time version of the dynamical system using a special property of Growth Transforms. The complete neuron model is summarized in **Table 3**.

The operation of the proposed neuron model is therefore governed by two sets of dynamics: (a) minimization of the network energy functional $\mathcal{H}$; (b) modulation of the trajectory using a time-constant $\tau_i(t)$, also referred to as modulation function in this paper. Fortunately, the evolution of $\tau_i(t)$ can be made as complex as possible without affecting the asymptotic fixed-point solution of the optimization process. It can be a made a function of local variables like v_i and $\dot{v}_i$ or a function of global/network variables like $\mathcal{H}$ and $\dot{\mathcal{H}}$. Different choices of the modulation function can lead to different trajectories followed by the neural variables under the same energy contour, as illustrated in **Figure 3**. In section 3, we show how different forms of $\tau_i(t)$ produce different sets of neuronal dynamics consistent with the dynamics that have been reported in neurobiology.

3. RESULTS

The proposed approach enables us to decouple the three following aspects of the spiking neural network:

(a) Fixed points of the network energy functional, which depend on the network configuration and external inputs;
(b) Nature and shape of neural responses, without affecting the network minimum; and
(c) Spiking statistics and transient neural dynamics at the cellular level, without affecting the network minimum or spike shapes.

This makes it possible to independently control and optimize each of these neuro-dynamical properties without affecting the others. The first two aspects arise directly from an appropriate selection of the energy functional and were demonstrated in section 2.2.1. In this section, we show how the modulation function, in essence, loosely models cell excitability, and can be varied to tune transient firing statistics based on local and/or global variables. This allows us to encode the same optimal solution using widely different firing patterns across the network, and have unique potential benefits for neuromorphic applications. Codes for the representative examples given in this section are available at Mehta et al. (2019).

3.1. Single-Neuron Dynamics

We first show how we can reproduce a number of single-neuron response characteristics by changing the modulation function $\tau_i(t)$ in the neuron model. For this, we consider an uncoupled network, where

$$Q_{ij} = \begin{cases} Q_0\ \Omega^{-1} & , \forall i = j \\ 0\ \Omega^{-1} & , \forall i \neq j \end{cases} \tag{33}$$

We will subsequently extend these dynamics to build coupled networks with interesting properties like memory and global adaptation for energy-efficient neural representation. The results reported here are representative of the types of dynamical properties the proposed model can exhibit, but are by no means exhaustive. Readers are encouraged to experiment with different inputs and network parameters in the software (MATLAB©) implementation of the Growth Transform neuron model (Mehta et al., 2019). The tool enables users to visualize the effects of different modulation functions and other parameters on the neural dynamics, as well as the time-evolution of population trajectories and the network energy function with different inputs and under different initial conditions.

3.1.1. Standard Tonic-Spiking Response

When stimulated with a constant current stimulus b_i, a vast majority of neurons fire single, repetitive action potentials for the duration of the stimulus, with or without adaptation (McCormick et al., 1985; Agmon and Connors, 1989; Gibson et al., 1999). The proposed model shows tonic spiking without adaptation when the modulation function $\tau_i(t) = \tau$, where $\tau > 0$ s. A simulation of tonic spiking response using the neuron model is given in **Figure 4A**.

3.1.2. Bursting Response

Bursting neurons fire discrete groups of spikes interspersed with periods of silence in response to a constant stimulus (McCormick et al., 1985; Agmon and Connors, 1989; Gray and McCormick,

TABLE 3 | Complete continuous-time GT spiking neural network (Proof in Appendix D).

For a network of M neurons with state variables $\boldsymbol{v} = \{v_i\} \in \mathbb{R}^M$, where the trans-conductance coupling matrix is denoted by $\mathbf{Q} = \{Q_{ij}\} \in \mathbb{R}^M \times \mathbb{R}^M$ and the external stimulus vector is denoted by $\boldsymbol{b} = \{b_i\} \in \mathbb{R}^M$, the time-evolution of the network under bound constraints on the state variables $v_i(t) \leq v_c \ \ \forall t$, is governed by the following continuous-time dynamical system:

$$\tau_i(t)\frac{dv_i(t)}{dt} + v_i(t) = v_C \frac{-\frac{\partial \mathcal{H}}{\partial v_j(t)} v_C + \lambda v_i(t)}{-\frac{\partial \mathcal{H}}{\partial v_j(t)} v_i(t) + \lambda v_C}, \tag{31}$$

where

$$\frac{\partial \mathcal{H}}{\partial v_i(t)} = \sum_{j=1}^{M} Q_{ij} v_j(t) - b_i(t) + \Psi(v_i(t));$$

$$\Psi(v_i(t)) = \left\{ \begin{array}{ll} I_\Psi\ A & ; \ v_i(t) > 0\ V \\ 0\ A & ; \ v_i(t) \leq 0\ V \end{array} \right\};$$

λ is a fixed current parameter such that $\lambda > |\frac{\partial \mathcal{H}}{\partial v_i(t)}| \ \ \forall i, t$;

$0 \leq \tau_i(t, v_i, \dot{v}_i, \mathcal{H}, \dot{\mathcal{H}}) < \infty$ is a modulation function that can be tuned individually for each neuron to encode different trajectories and different steady-state spiking dynamics corresponding to the optimal solution.

The composite spike response of the i-th neuron at time t is given by

$$s_i(t) = \ v_i(t) + C\Psi(v_i(t)), \tag{32}$$

where the trans-impedance parameter $C > 0\ \Omega$ determines the magnitude of each spike.

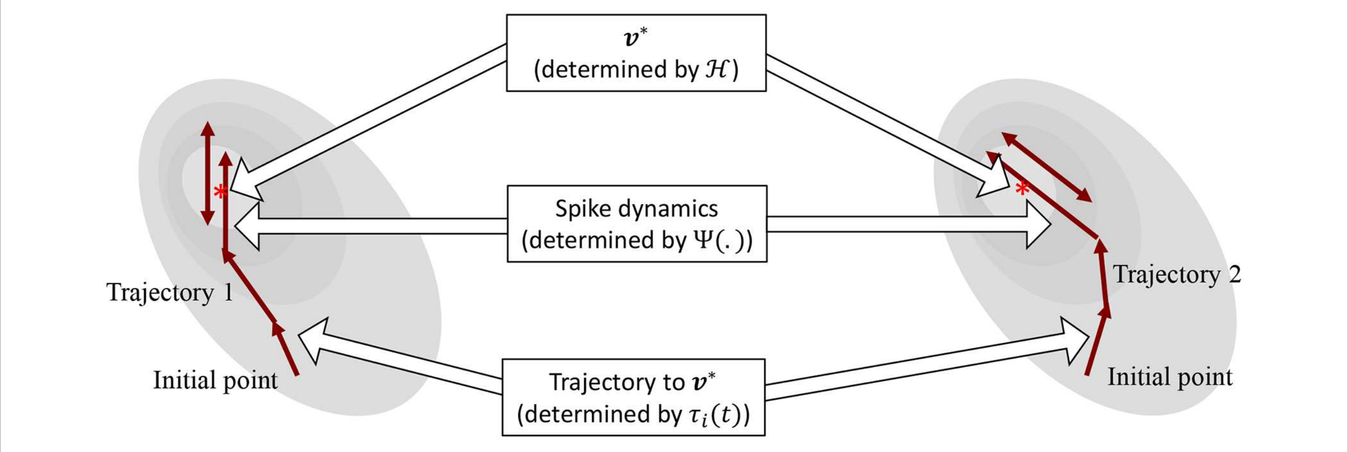

$\boldsymbol{v}^*$ (determined by $\mathcal{H}$)
Spike dynamics (determined by $\Psi(.)$)
Trajectory to $\boldsymbol{v}^*$ (determined by $\tau_i(t)$)
Trajectory 1
Initial point
Trajectory 2
Initial point

FIGURE 3 | Decoupling of network solution, spike shape and response trajectory using the proposed model. Different modulation functions lead to different steady-state spiking dynamics under the same energy contour.

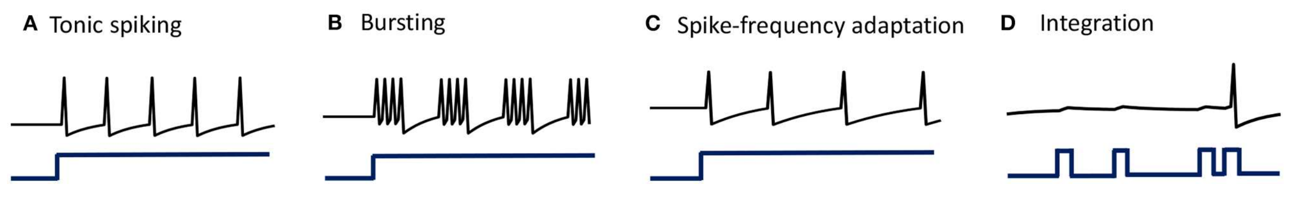


FIGURE 4 | (A–D) Simulations demonstrating different single-neuron responses obtained using the GT neuron model.

1996; Brumberg et al., 2000). Bursting arises from an interplay of fast ionic currents responsible for spiking, and slower intrinsic membrane currents that modulate the spiking activity, causing the neuron to alternate between activity and quiescence. Bursting response can be simulated in the proposed model by modulating $\tau_i(t)$ at a slower rate compared to the generation of action potentials, in the following way

$$\tau_i(t) = \begin{cases} \tau_1 \ s \ , \ c_i(t) < B \\ \tau_2 \ s \ , \ c_i(t) \geq B \end{cases} \tag{34}$$

where $\tau_1 > \tau_2 > 0$ s, B is a parameter and the count variable $c_i(t)$ is updated according to

$$c_i(t) = \begin{cases} \lim\limits_{\Delta t \to 0} c_i(t - \Delta t) + \mathcal{I}[v_i(t) > 0]) \ , \ \lim\limits_{\Delta t \to 0} c_i(t - \Delta t) < B \\ 0 \qquad\qquad\qquad\qquad\qquad\quad , \ \lim\limits_{\Delta t \to 0} c_i(t - \Delta t) \geq B \end{cases} , \tag{35}$$

$\mathcal{I}[.]$ being an indicator function. Simulation of a bursting neuron in response to a step input is given in **Figure 4B**.

3.1.3. Spike-Frequency Adaptation

When presented with a prolonged stimulus of constant amplitude, many cortical cells initially respond with a high-frequency spiking that decays to a lower steady-state frequency (Connors and Gutnick, 1990). This adaptation in the firing rate is caused by a negative feedback to the cell excitability due to the gradual inactivation of depolarizing currents or activation of slow hyperpolarizing currents upon depolarization, and occur at a time-scale slower than the rate of action potential generation. We modeled spike-frequency adaptation by varying the modulation function according to

$$\tau_i(t) = \tau - 2\phi\big(h(t) * \Psi(v_i(t))\big) \tag{36}$$

where $h(t) * \Psi(v_i)(t)$ is a convolution operation between a continuous-time first-order smoothing filter $h(t)$ and the spiking function $\Psi(v_i(t))$, and

$$\phi(x) = \tau\left(\frac{1}{1 + \exp(x)}\right) \tag{37}$$

is a compressive function that ensures $0 \leq \tau_i(t) \leq \tau$ s. The parameter τ determines the steady-state firing rate for a particular stimulus. A tonic-spiking response with spike-frequency adaptation is shown in **Figure 4C**.

3.1.4. Integrator Response

When the baseline input is set slightly negative so that the fixed point is below the threshold, the neuron works like a leaky integrator as shown in **Figure 4D**, preferentially spiking to high-frequency or closely-spaced input pulses that are more likely to make v_i cross the threshold.

3.2. Coupled Spiking Network With Pre-synaptic Adaptation

We can extend the proposed framework to a network model where the neurons, apart from external stimuli, receive inputs from other neurons in the network. We begin by considering **Q** to be a positive-definite matrix, which gives a unique solution of (8). Although elements of the coupling matrix **Q** already capture the interactions among neurons in a coupled network, we can further define the modulation function as follows to make the proposed model behave as a standard spiking network

$$\tau_i(t) = \phi\left(h(t) * \sum_{j=1}^{M} Q_{ij}\Psi(v_j(t))\right) \tag{38}$$

with the compressive-function $\phi(.)$ given by (37). Equation (38) ensures that $Q_{ij} > 0$ corresponds to an excitatory coupling from the pre-synaptic neuron j, and $Q_{ij} < 0$ corresponds to an inhibitory coupling, as demonstrated in **Figure 5A**. Note that irrespective of whether such a pre-synaptic adaptation is implemented or not, the neurons under the same energy landscape would converge to the same sub-domain, albeit with different response trajectories and steady-state limit-cycles. This is illustrated in **Figure 5B** which plots the energy contours for a two-neuron network corresponding to a **Q** matrix with excitatory and inhibitory connections and a fixed stimulus vector ***b***. **Figure 5B** also shows the responses of the two neurons starting from the same initial conditions, with and without pre-synaptic adaptation (where the latter corresponds to the case where the only coupling between the two neurons is through the coupling matrix **Q**, but there is no pre-synaptic spike-time dependent adaptation). Because the energy landscape is the same in both cases, the neurons converge to the same sub-domain, but with widely varying trajectories and steady-state response patterns.

3.3. Coupled Network With Pre-synaptic and Global Adaptation

Apart from the pre-synaptic adaptation that changes individual firing rates based on the input spikes received by each neuron, neurons in the coupled network can be made to adapt according to the global dynamics by changing the modulation function as follows

$$\tau_i(t) = \phi\left(h(t) * \Big(\sum_{j=1}^{M} Q_{ij}\Psi(v_j(t)) - \mathcal{F}(\mathcal{H}, \dot{\mathcal{H}})\Big)\right) \tag{39}$$

with the compressive-function $\phi(.)$ given by (37). The new function $\mathcal{F}(.)$ is used to capture the dynamics of the network cost-function. As the network starts to stabilize and converge to a fixed-point, the function $\tau_i(.)$ adapts to reduce the spiking rate of the neuron without affecting the steady-state solution. **Figures 5C,D** show the time-evolution of the spiking energy $\int \Psi(.)dv$ and the spike-trains for a two-neuron network without global adaptation and with global adaptation, respectively, using

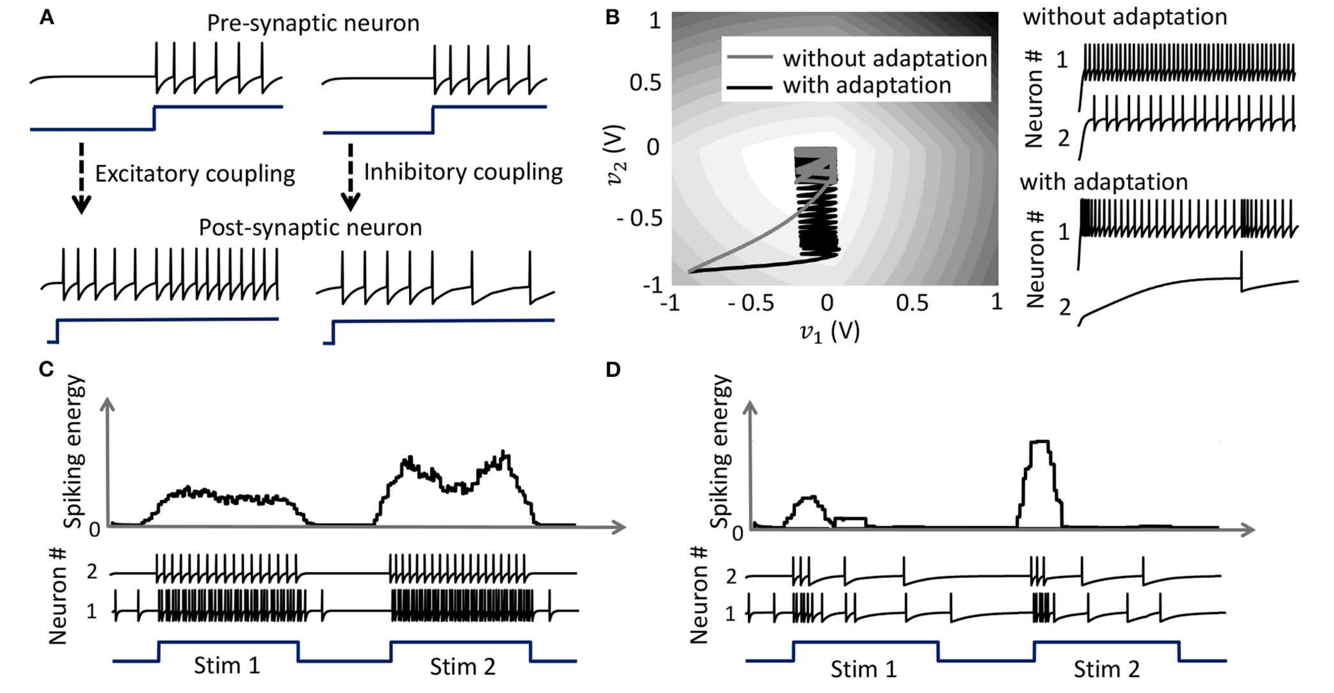


FIGURE 5 | (A) Results from a 2-neuron network with excitatory and inhibitory couplings; **(B)** Energy optimization process under different conditions lead to different limit cycles within the same energy landscape. **(C,D)** Mean spiking energy $\int \Psi(.)dv$ and firing patterns in response to two stimuli in the absence and presence of global adaptation, respectively.

the following form for the adaptation term

$$\mathcal{F}(\mathcal{H}, \dot{\mathcal{H}}) = \begin{cases} \mathcal{F}_0 \ , \ \mathbb{E}_T(\dot{\mathcal{H}}) \approx 0 \\ 0 \quad , \ \text{otherwise.} \end{cases} \qquad (40)$$

where $\mathcal{F}_0 > 0$ is a tunable parameter. This feature is important in designing energy-efficient spiking networks where energy is only dissipated during transients.

3.4. Network Response and Network Trajectories

In order to outline the premises of the next few experiments on population dynamics using the geometric interpretation outlined in section 2.3, we consider a small network of neurons on a two-dimensional co-ordinate space, and assign arbitrary inputs to the neurons. A Gaussian kernel is chosen for the coupling matrix **Q** as follows

$$Q_{ij} = \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2). \qquad (41)$$

This essentially clusters neurons with stronger couplings between them closer to each other on the co-ordinate space, while placing neurons with weaker couplings far away from each other. A network consisting of 20 neurons is shown in **Figure 6A**, which also shows how the spiking activity changes as a function of the location of the neuron w.r.t. the hyperplane corresponding to $\bar{\Psi} = 0$, indicated by the white dashed line. Each neuron is color coded based on the mean firing rate (normalized w.r.t. the maximum mean firing rate) with which it responds when the stimulus is on. **Figure 6B** shows the spike raster for the entire network. We see that the responsiveness of the neurons to a particular stimulus increases with the distance at which it is located from the hypothetical hyperplane in

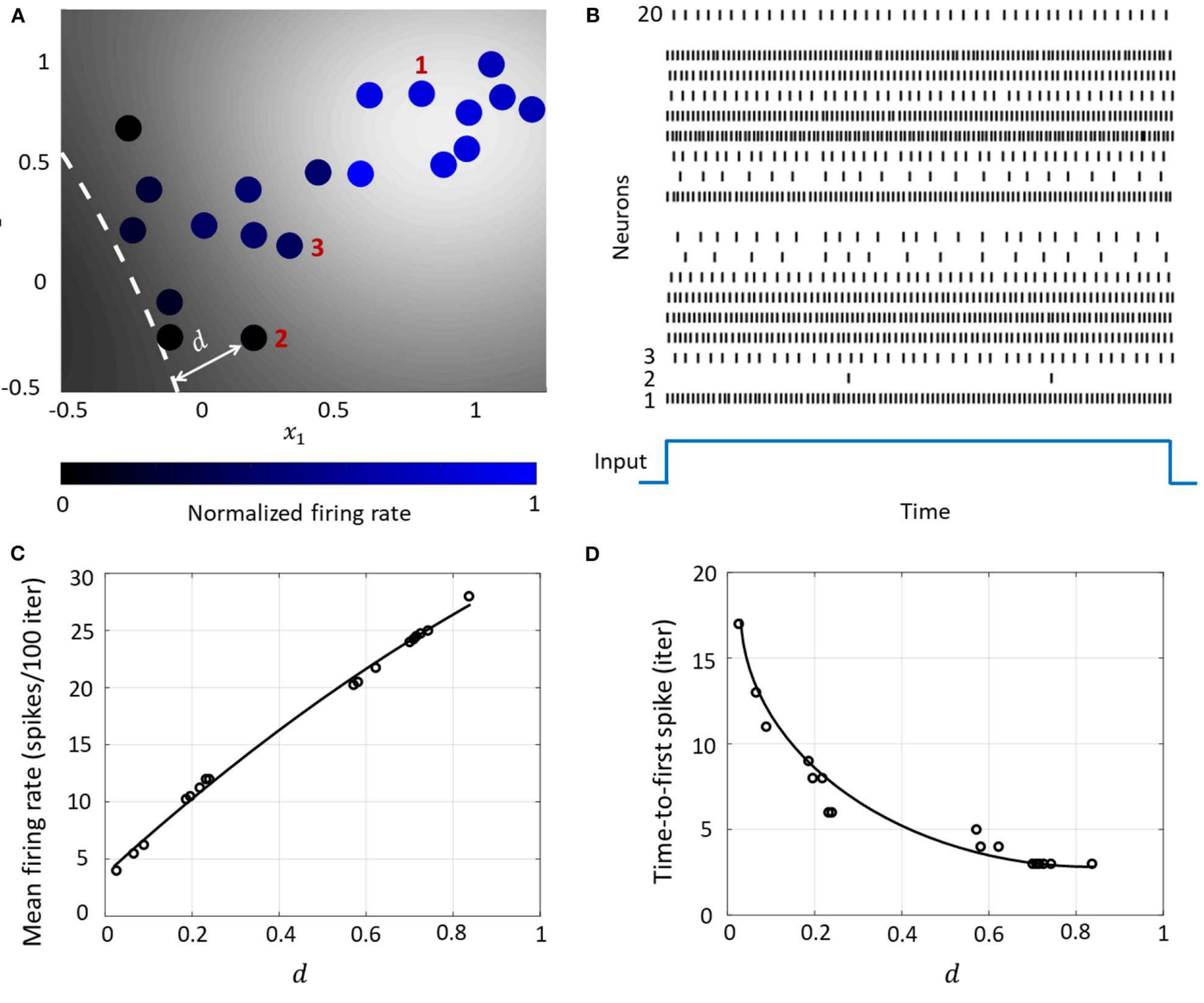


FIGURE 6 | (A) Contour plot of spiking activity corresponding to a particular stimulus vector. Neurons are colored according to their mean firing rate (normalized w.r.t. the maximum firing rate) during the stimulus period. The white dashed line is the hyperplane corresponding to $\bar{\Psi} = 0$. **(B)** Spike raster for all neurons for the input in **(A)**. **(C)** The mean firing rate and **(D)** time-to-first spike as a function of the distance d for each neuron in the network.

the high-dimensional space to which the neurons are mapped through kernel transformation. We show below how this geometric representation can provide insights on population-level dynamics in the network considered.

3.4.1. Rate and Temporal Coding

The Growth Transform neural network inherently shows a number of encoding properties that are commonly observed in biological neural networks (Rieke et al., 1999; Gerstner and Kistler, 2002). For example, the firing rate averaged over a time window is a popular rate coding technique that claims that the spiking frequency or rate increases with stimulus intensity (Adrian and Zotterman, 1926). A temporal code like the time-to-first-spike posits that a stronger stimulus brings a neuron to the spiking threshold faster, generating a spike, and hence relative spike arrival times contain critical information about the stimulus (Thorpe, 1990).

These coding schemes can be interpreted under the umbrella of network coding using the same geometric representation as considered above. Here, the responsiveness of a neuron is closely related to its proximity to the hyperplane. The neurons which exhibit more spiking are located at a greater distance from the hyperplane. We see from **Figures 6C,D** that as this value increases, the average firing rate of a neuron (number of spikes in a fixed number of time-steps or iterations) increases, and the time-to-first spike becomes progressively smaller. Neurons with a distance value below a certain threshold do not spike at all during the stimulus period, and therefore have a mean firing rate of zero and time-to-spike at infinity. Therefore, based on how the network is configured in terms of synaptic inputs and connection strengths, the spiking pattern of individual neurons conveys critical information about the network hyperplane and their placement with respect to it.

3.4.2. Network Coding and Neural Population Trajectories

The encoding of a stimulus in the spatiotemporal evolution of activity in a large population of neurons is often represented in neurobiological literature by a unique trajectory in a high-dimensional space, where each dimension accounts for the time-binned spiking activity of a single neuron. Projection of the high-dimensional activity to two or three critical dimensions using dimensionality reduction techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) have been widely used across organisms and brain regions to shed light on how neural population response evolves when a stimulus is delivered (Friedrich and Laurent, 2001; Stopfer et al., 2003). For example in identity coding, trajectories corresponding to different stimuli evolve toward different regions in the reduced neural subspace, that often become more discriminable with time and are stable over repeated presentations of a particular stimulus (Friedrich and Laurent, 2001; Stopfer et al., 2003; Galán et al., 2004). We show how this can be explained in the context of the geometric interpretation.

For the same network as above, we start with the simplest possible experiment, starting from the same baseline, and perturbing the stimulus vector in two different directions. This pushes the network hyperplane in two different directions, exciting different subsets of neurons, as illustrated in **Figures 7A,B**. A similar dimensionality reduction to three principal components in **Figure 7C** shows the neural activity unfolding in distinct stimulus-specific areas of the neural subspace. The two contour plots also show that some neurons may spike for both the inputs, while some spike selectively for one of them. Yet others may not show any spiking for either stimulus, but may spike for some other stimulus vector and the corresponding stimulus-specific hyperplane.

3.5. Coupled Spiking Network With Non-positive Definite Q

As illustrated in **Figure 8**, a coupled spiking network can function as a memory element, when **Q** is a non-positive definite matrix and

$$\tau_i(t) = \phi\left(h(t) * \sum_{j=1}^{M} Q_{ij}\Psi(v_j(t))\right), \tag{42}$$

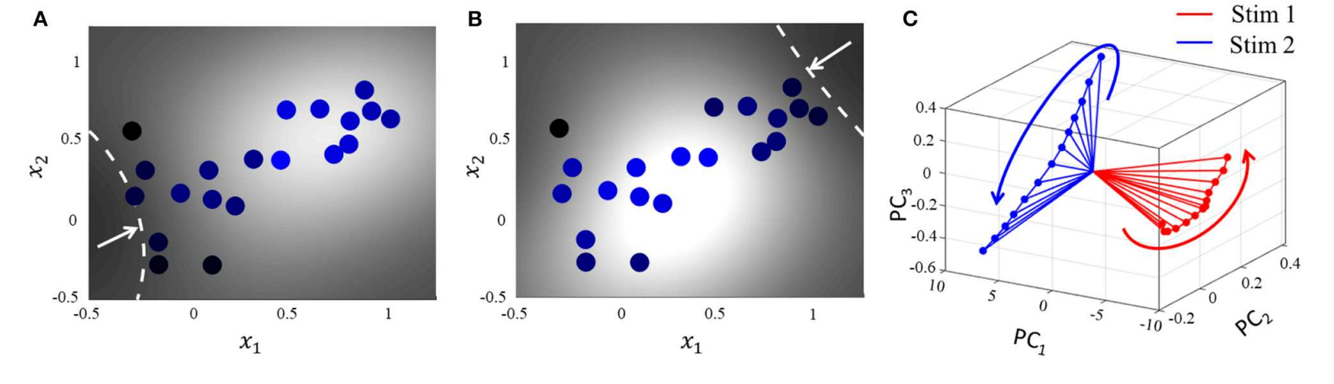


FIGURE 7 | (A,B) Perturbation of the stimulus vector in different directions for the same network produces two different contours. **(C)** Corresponding population activities trace different trajectories in the neural subspace, similar to what has been reported in (Stopfer et al., 2003).

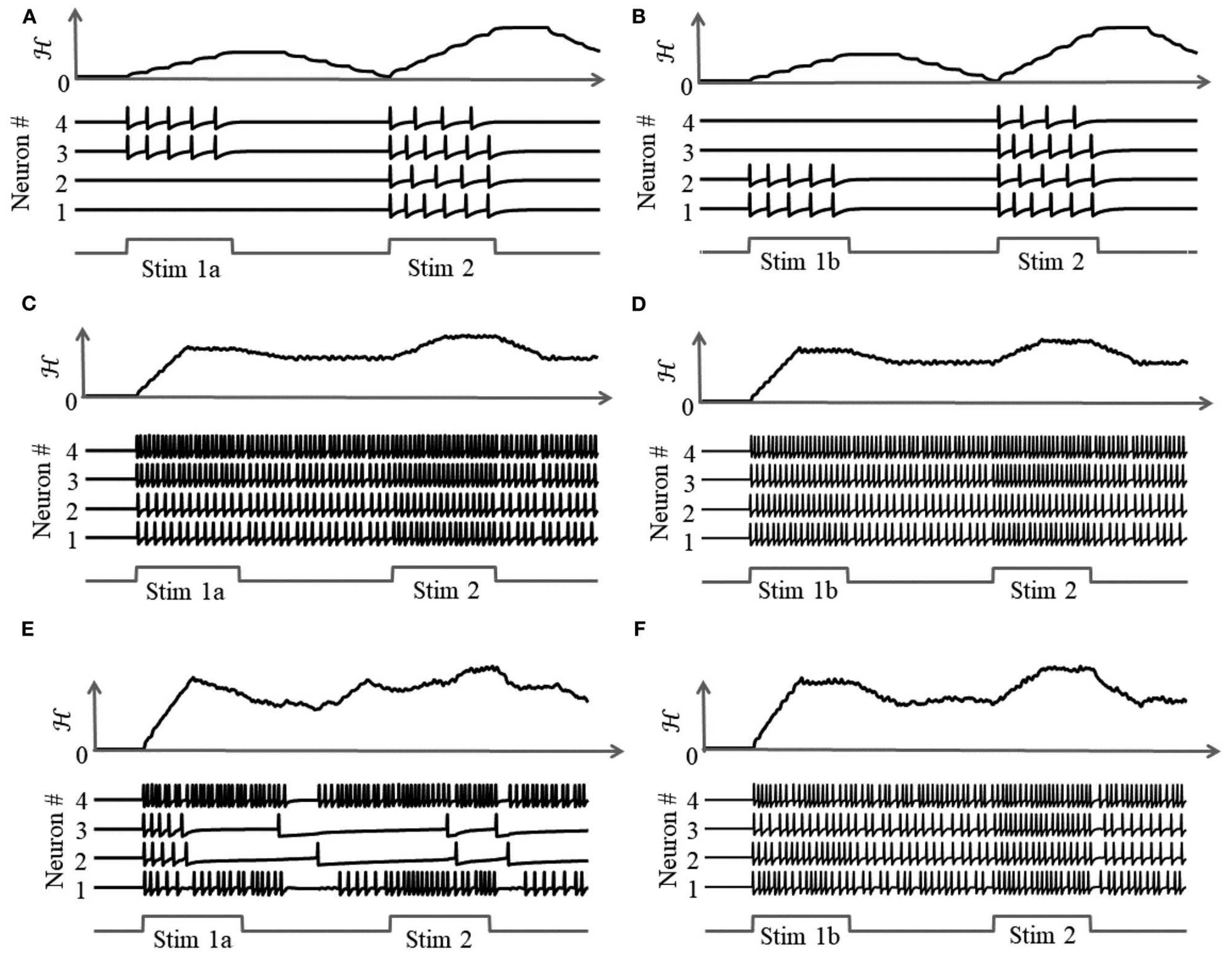


FIGURE 8 | Stimulus response for a 4-neuron network with different stimulus histories for: **(A,B)** an uncoupled network; **(C,D)** a coupled network with a positive definite coupling matrix **Q**; **(E,F)** a coupled network with a non-positive definite coupling matrix **Q**.

due to the presence of more than one attractor state. We demonstrate this by considering two different stimulus histories in a network of four neurons, where a stimulus "Stim 1a" precedes another stimulus "Stim 2" in **Figures 8A,C,E**, and a different stimulus 'Stim 1b' precedes "Stim 2" in **Figures 8B,D,F**. Here, each "stimulus" essentially corresponds to a different input vector ***b***. For an uncoupled network, where neurons do not receive any inputs from other neurons, the network energy increases when the first stimulus is applied and returns to zero afterwards, and the network begins from the same state again for the second stimulus as for the first, leading to the same firing pattern for the second stimulus, as shown in **Figures 8A,B**, independent of the history. For a coupled network with a positive definite coupling matrix **Q**, reinforcing loops of spiking activity in the network may not allow the network energy to go to zero after the first stimulus is removed, and the residual energy may cause the network to exhibit a baseline activity that depends on stimulus history, as long as there is no dissipation. When the second stimulus is applied, the initial conditions for the network are different for the two stimulus histories, leading to two different transients until the network settles down into the same steady-state firing patterns, as shown in **Figures 8C,D**. For a non-positive definite coupling matrix **Q** however, depending on the initial condition, the network may settle down to different solutions for the same second stimulus, due to the possible presence of more than one local minimum. This leads to completely different transients as well as steady-state responses for the second stimulus, as shown in **Figures 8E,F**. This history-dependent stimulus response could serve as a short-term memory, where residual network energy from a previous external input subserves synaptic interactions among a population of neurons to set specific initial conditions for a future stimulus based on the stimulus history, forcing the network to settle down in a particular attractor state.

3.6. Associative Memory Network Using Growth Transform Neuron Models

Associative memories are neural networks which can store memory patterns in the activity of neurons in a network through a Hebbian modification of their synaptic weights; and recall a stored pattern when stimulated with a partial fragment or a noisy version of the pattern (Cutsuridis et al., 2010). Various works have studied associative memories using networks of spiking neuron models having different degrees of abstraction and architectural complexities (Lansner, 2009; Palm, 2013). Here, we demonstrate using an associative memory network of Growth Transform neurons how we can use network trajectories to recall stored patterns, and moreover, use global adaptation to do so using very few spikes and high recall accuracy.

Our network comprises $M = 100$ neurons, out of which a randomly selected subset $m = 10$ are active for any stored memory pattern. The elements of the transconductance coupling matrix are set according to the following standard Hebbian learning rule

$$Q_{ij} = \frac{1}{k}\sum_{s=1}^{S} t_i^s t_j^s, \tag{43}$$

where k is a scaling factor and $\boldsymbol{t}^s \in [0,1]^M,\ s = 1, ..., S$, are the binary patterns stored in the network. During the recall phase, only half of the cells active in the original memory are stimulated with a steady depolarizing input, and the spiking pattern across the network is recorded. Instead of determining the active neurons during recall through thresholding and directly comparing with the stored binary pattern, we quantitatively measure the recall performance of the network by computing the mean distance between each pair of original-recall spiking dynamics as they unfold over time. This ensures that we not only take into account the firing of the neurons that belong to the pattern albeit are not directly stimulated, but also enables us to exploit any contributions from the rest of the neurons in making the spiking dynamics more dissimilar in comparison to recalls for other patterns.

When the network is made to globally adapt according to the system dynamics, the steady-state trajectories can be encoded using very few spikes. **Figures 9A,B** show the raster plots for the stored patterns without and with global adaptation, respectively, when $S = 10$; and **Figures 9C,D** are the corresponding plots during recall. For each recall pattern, spike patterns for the directly stimulated neurons are plotted first, followed by the other 5 neurons that are not directly stimulated but belong to the pattern; and finally the rest of the neurons in random order. The ordering of neurons is kept the same for plotting spike rasters for

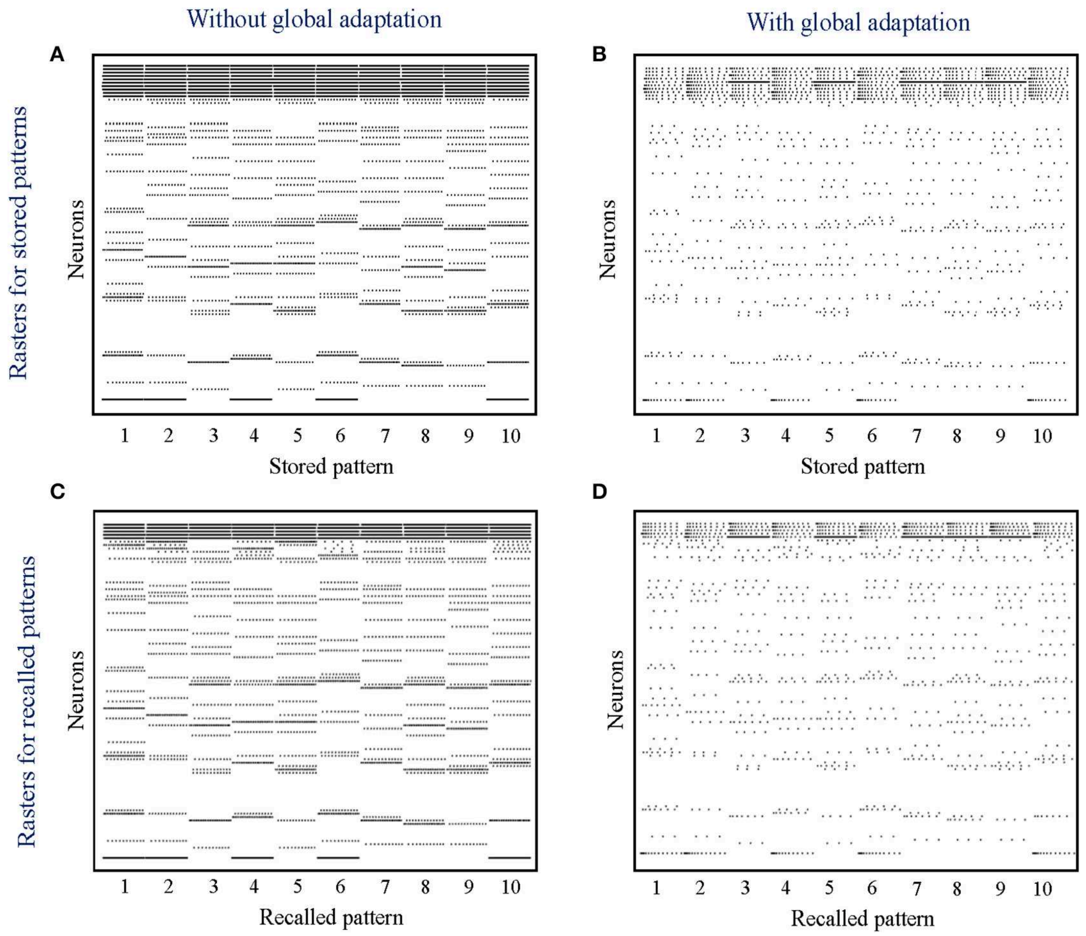


FIGURE 9 | (A,B) Spike rasters for the 10 stored patterns in the absence and presence of global adaptation, respectively; **(C,D)** Spike rasters for the 10 recall cases in the absence and presence of global adaptation, respectively.

the stored patterns. During decoding, a straightforward metric using the average distance between time-binned mean firing rates for the original and recall trajectories produces similarity matrices presented in **Figures 10A,B**, where we see that global adaptation does not perform as well. However, the information in this case also lies in the spike-times and changes in firing rate over time for each neuron. Including these features in the decoding vectors for stored and recalled patterns, we get clean recalls in both cases as shown in **Figures 10C,D**. The decoding vector for the *n*-th time-bin in this case is given by

$$\boldsymbol{d}_n = \begin{bmatrix} \boldsymbol{r}_n \\ \Delta \boldsymbol{t}_{\Psi n} \\ \Delta \boldsymbol{r}_n \end{bmatrix}, \tag{44}$$

where $\boldsymbol{r}_n$, $\Delta \boldsymbol{t}_{\Psi n}$ and $\Delta \boldsymbol{r}_n$ are the vectors of mean firing rates, mean inter-spike intervals and changes in the mean firing rates for the *n*-th bin for the entire network, respectively. The mean inter-spike interval is set equal to the bin length if there is a single spike over the entire bin length, and equal to twice the bin length if there are none. Note that the inter-spike interval computed for one time-bin may be different from $(1/r)$, particularly for low firing rates, and hence encodes useful information. The similarity metric between the *u*-th stored pattern and the *v*-th recall pattern is given by

$$s_{u,v} = 1 - \text{dist}_{u,v}, \tag{45}$$

where $\text{dist}_{u,v}$ is the mean Euclidean distance between the two decoding vectors over the total number of time-bins, normalized between $[0, 1]$.

To estimate the capacity of the network, we calculate the mean recall accuracy over 10 trials for varying number of stored patterns, both with and without global adaptation. **Figure 11A** plots the mean recall accuracy for different number of patterns stored for the two cases, and **Figure 11B** plots the mean number of spikes for each storage. For each plot, the shaded region indicates the range of values across trials. As expected, the accuracy is 100% for lesser storage,but degrades with higher loading. However with global adaptation, the degradation is seen to be more graceful for a large range of storage with the decoding used in **Figures 10C,D**, allowing the network to recall patterns more accurately using much fewer spikes. Hence by exploiting suitable decoding techniques, we can implement highly energy-efficient spiking associative memory networks with high storage capacity.

Note that the recall accuracy using global adaptation deteriorates faster for > 175 patterns. The proposed decoding algorithm, which determines the recall accuracy, takes into

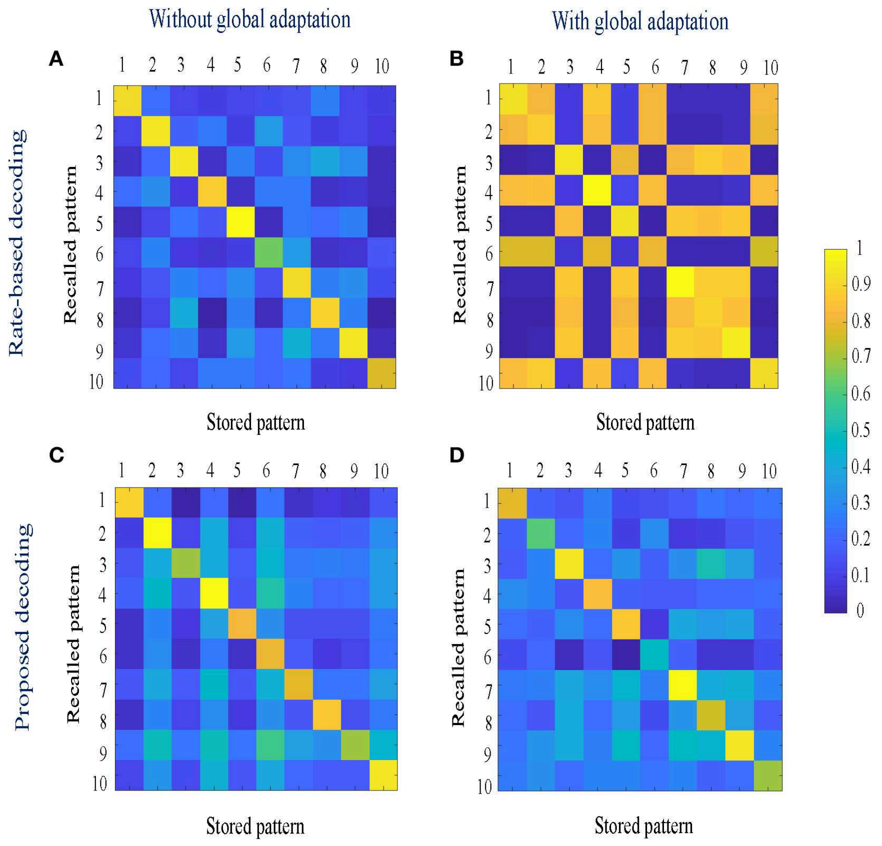


FIGURE 10 | (A,B) Similarity matrices between storage and recall with a rate-based decoding metric; **(C,D)** Similarity matrices with a decoding metric that also includes spike-times and changes in mean firing rates.

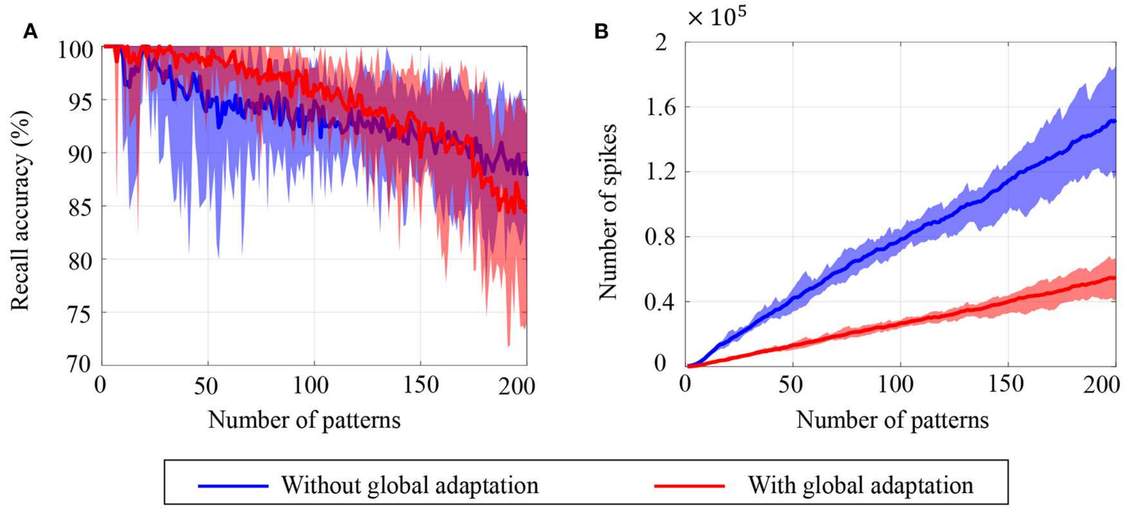


FIGURE 11 | Ensemble plots showing **(A)** mean recall accuracy and **(B)** mean number of spikes as memory load increases for the network, in the absence as well as presence of global adaptation. The range of values across the ensemble is shown by the shaded area.

account the mean spiking rates, inter-spike intervals and changes in spike rates. It is possible that as the number of spikes is reduced through the use of global adaptation, the information encoded in first-order differences (inter-spike intervals or spike rates) may not be sufficient to encode information at high fidelity, resulting in the degradation in recall accuracy when the number of patterns increased. However, augmenting the decoding features with higher-order differences in inter-spike intervals or spike rates may lead to an improved performance for higher storage.

3.6.1. Classification of Noisy MNIST Images

Aside from pattern completion, associative networks are also commonly used for identifying patterns from their noisy counterparts. We use a similar associative memory network as above to classify images from the MNIST dataset which were corrupted with additive white Gaussian noise at different signal-to-noise ratios (SNRs), and which were, unlike in the previous case, unseen by the network before the recall phase. The network size in this case was $M = 784$, the number of pixels in each image, and the connectivity matrix was set using a separate, randomly selected subset of 5,000 binary, thresholded images from the training dataset according to (43). Unseen images from the test dataset were corrupted at different SNRs and fed to the network after binary thresholding. **Figures 12A–C** show instances of the same test image at different SNRs after binary thresholding. As before, the non-zero pixels got a steady depolarizing input. A noisy test image was assigned to the class corresponding to the closest training image according to the similarity metric in (45).

The test accuracies and mean spike counts for a test image are plotted in **Figures 12D,E**, respectively, for different noise levels. We see that even for relatively high noise levels, the network has a robust classification performance. As before, a global adaptation based on the state of convergence of the network produces a slightly better performance with fewer spikes per test image.

4. CONCLUSIONS

This paper introduces the theory behind a new spiking neuron and population model based on the Growth Transform dynamical system. The system minimizes an appropriate energy functional under realistic physical constraints to produce emergent spiking activity in a population of neurons. The proposed work is the first of its kind to treat the spike generation and transmission processes in a spiking network as an energy-minimization problem involving continuous-valued neural state variables like the membrane potential. The neuron model and its response are tightly coupled to the network objective, and are flexible enough to incorporate different neural dynamics that have been observed at the cellular level in electrophysiological recordings.

The paper is accompanied by a software tool (Mehta et al., 2019) that enables readers to visualize the effects of different model parameters on the neural dynamics. Many more neural dynamics can be simulated using the model and readers are encouraged to experiment with different network parameters. The paper and the tool illustrate how dynamical and spiking responses of neurons can be derived directly from a network objective or energy functional of continuous-valued neural variables. The general approach offers an elegant way to design neuromorphic machine learning algorithms by bridging the gap that currently exists between bottom-up models that can simulate biologically realistic neural dynamics but do not have a network-level representation, and top-down machine learning models that start with a network loss function, but reduce the problem to the model of a non-spiking neuron with static non-linearities.

In this regard, machine learning models are primarily developed with the objective of minimizing the error in inference by designing a loss function that captures dependencies among variables, for example, features and class labels. Learning in

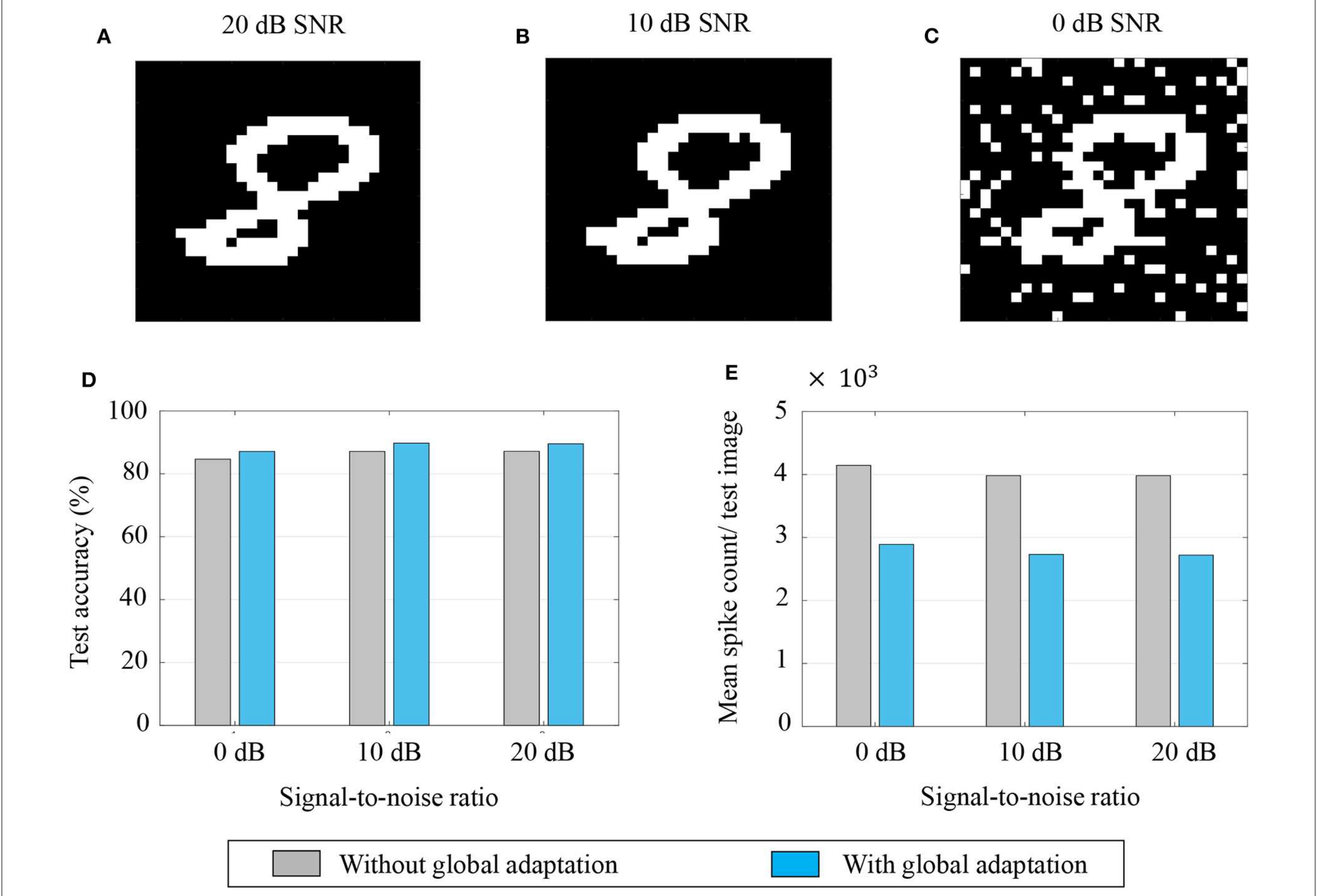


FIGURE 12 | (A–C) An example of a test image corrupted with additive white Gaussian noise at 20, 10, and 0 dB SNR, respectively; **(D,E)** Test accuracy and mean spike count/test image for different noise levels.

this case, as pointed out in LeCun et al. (2006), consists of adapting weights in order to associate low energies (losses) to observed configurations of variables, and high energies (losses) to unobserved ones. The non-differentiable nature of spiking dynamics makes it difficult to formulate loss functions involving neural variables. Neuromorphic algorithms currently work around this problem in different ways, including mapping deep neural nets to spiking networks through rate-based techniques (O'Connor et al., 2013; Rueckauer et al., 2017), formulating loss functions that penalize the difference between actual and desired spike-times (Xin and Embrechts, 2001; Bohte et al., 2002), or approximating the derivatives of spike signals through various means (Lee et al., 2016; Shrestha and Orchard, 2018; Zenke and Ganguli, 2018). Formulating the spiking dynamics of the entire network using an energy function involving neural state variables across the network would enable us to directly use the energy function itself for learning weight parameters; and forms the basis for our future work. Since the proposed energy function encompasses all the neurons in the network, and not just the "visible neurons" as in most neuromorphic machine learning algorithms, it can potentially enable easier and more effective training of hidden neurons in deep networks. Moreover, it would allow us to incorporate and experiment with biologically relevant neural dynamics that could have significant performance and energy benefits.

4.1. Relation With Other Neural Networks and Spiking Neuron Models

The network energy functional bears similarity with the Ising Hamiltonians used in Hopfield networks (Hopfield, 1982), Boltzmann machines (Hinton et al., 1986) or spin-glass models (Gardner and Derrida, 1988), but contains an additional integral term $\int \Psi(.)dv$ as in continuous-time Hopfield networks with graded neurons (Šíma and Orponen, 2003). However, unlike in continuous-time Hopfield networks where $\Psi^{-1}(.)$ is assumed to be a saturation/squashing function of a rate-based representation, the role of $\Psi(.)$ in the proposed model is to implement a barrier or a penalty, such that the neural responses can produce spiking dynamics. This enables us to obtain neural responses at the level of individual spikes instead of average rate-based responses; and allows for a more fine-grained control over the spiking responses of the network. The saturation (squashing) function, on the other hand, is implemented by the bound constraints on the Growth Transform updates, and hence the network is not limited to choosing a specific form of saturation non-linearity (e.g., sigmoid).

The energy-based formulation described in section 2.1 could also admit other novel interpretations. For instance, for the form of $\Psi(.)$ considered in (18), the barrier function can be rewritten as $\int_{-\infty}^{v_i} \Psi(v)dv = \Psi_i v_i$, where $\Psi_i = 0\ A$ if $v_i \leq 0\ V$ and $\Psi_i = I_\Psi\ A$ if $v_i > 0\ V$. For a continuous-time implementation (discrete-time step that is sufficiently small), $v_i(t)$ will be reset as soon as it reaches the threshold ($0\ V$), and will not exceed the threshold. In this case, we can write $\Psi_i(t) \geq 0$ and $\Psi_i(t)v_i(t) = 0\ \forall t$. This is equivalent to Karush-Kuhn-Tucker (KKT) conditions. Thus the spike events $\Psi_i(t),\ i = 1, ..., M$, act as the KKT multipliers corresponding to the M inequality constraints $v_i(t) \leq 0,\ i = 1, ..., M$ (Tucker and Kuhn, 1951), encoding the sensitivity of the i-th neuron to the constraint.

Also, if we consider the spike response as a displacement current, we can write

$$o_i = C_{out}\frac{dv_i}{dt}, \tag{46}$$

where C_{out} is the membrane capacitance. Note that o_i is the analog spike response current output and is different from $\Psi(v_i)$, which is the binary spike event. Then the membrane potential for the continuous-time Growth Transform neuron model in **Table 3** can be rewritten as

$$v_i(t) = \frac{1}{C_{out}}\int_{-\infty}^{t} o_i(t')dt'. \tag{47}$$

Thus according to this interpretation, the communication between neurons takes place using current waveforms, similar to integrate-and-fire models, and the current waveforms can be integrated at the post-synaptic neuron to recover the membrane potential. Note that the remapping between **W** and **Q** (described in section 2.1) would still hold, since we are transmitting analog spike current waveforms, and not post-synaptic current waveforms such as exponentially decaying functions, α-functions or simplified current pulses (digital bits) used in integrate-and-fire models (Traub and Miles, 1991; Mar et al., 1999).

4.2. Implication of Remapping on Neuromorphic Architectures

In the proposed neuron model, we abstracted out the essential aspects of spike generation and transmission that can replicate neural dynamics, and remapped synaptic interactions to an energy-based framework. As a result of the remapping procedure, the coupling matrix **Q** in our proposed model is proportional to the inverse of the synaptic weight matrix **W**. This paves the way for developing novel neuromorphic learning algorithms in the **Q**-domain that involves sparse local analog connectivity, but which actually translates to fully-connected non-sparse global connectivity in the **W**-domain. Thus, adapting one synaptic connection in the **Q**-domain, in this case, will be equivalent to adapting multiple synapses in the **W**-domain. Learning in the **Q**-domain will be a topic for future research.

4.3. Benefits of Decoupling Neurodynamical Parameters

A key advantage of the proposed framework is that it enables the decoupling of the three neurodynamical parameters - network solution, spike shapes and transient dynamics. Thus while the solution to the energy functional is determined by the coupling matrix **Q** and the stimulus vector **b**, independent control of the modulation function allows users to program the trajectory to the solution, which could be determined by an optimization process that is different from optimizing the energy functional. Some examples of these alternate objectives could be:

- A hybrid spiking network comprising neurons of different types (tonic spiking, bursting, non-spiking, etc.), as illustrated in section 3.1. The network would still converge to the same solution, but the spiking dynamics across the network could be exploited to influence factors such as speed, energy efficiency and noise-sensitivity of information processing.
- Optimization of some auxiliary network parameter, e.g., the total spiking activity. A related example (although not optimized w.r.t. any objective function) was illustrated in section 3.6 for a simple associative network. In this example, the network recalled the same set of patterns and classified MNIST images using two different time-evolutions of the modulation function corresponding to the presence and absence of global adaptation. In this case, it had the benefit of using fewer spikes to achieve better recall when a modified decoding metric was used.
- Modeling the effect of neurotransmitters and metabolic factors that have been known to affect the properties, activity and functional connectivity of populations of neurons. These factors endow the same network with the flexibility to generate different output patterns and produce different behaviors for the same stimulus (Hasselmo, 2002; McCormick and Nusbaum, 2014).
- Modeling the effect of diffusion processes or glial processes, that have been known to modulate response properties and synaptic transmission in neurons, influencing information processing and learning in the brain (Clarke and Barres, 2013; Fields et al., 2014).

DATA AVAILABILITY STATEMENT

All datasets generated for this study are included in the article/**Supplementary Material**.

AUTHOR CONTRIBUTIONS

AG and SC contributed to the conception and design of the study and wrote the first draft of the manuscript. AG and DM conducted the simulations. AG, DM, and SC designed the MATLAB interface. All authors contributed to the manuscript revision, read and approved the submitted version.

FUNDING

This work was supported in part by a research grant from the National Science Foundation (ECCS: 1935073). DM was supported by a research grant from the National Institutes of Health (5R21EY02836202).

ACKNOWLEDGMENTS

The authors would like to thank Dr. Kenji Aono at the Electrical and Systems Engineering department, Washington University, for developing a GPU version of the GT neural network model which is also included with the accompanying software toolbox (Mehta et al., 2019).

This manuscript has been released as a pre-print at Gangopadhyay et al. (2019).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2020.00425/full#supplementary-material

REFERENCES

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* 50, 303–304. doi: 10.1016/S0361-9230(99)00161-6

Adrian, E. D., and Zotterman, Y. (1926). The impulses produced by sensory nerve-endings: Part II. The response of a single end-organ. *J. Physiol.* 61, 151–171. doi: 10.1113/jphysiol.1926.sp002281

Agmon, A., and Connors, B. (1989). Repetitive burst-firing neurons in the deep layers of mouse somatosensory cortex. *Neurosci. Lett.* 99, 137–141. doi: 10.1016/0304-3940(89)90278-4

Baum, L. E., and Sell, G. (1968). Growth transformations for functions on manifolds. *Pac. J. Math.* 27, 211–227. doi: 10.2140/pjm.1968.27.211

Bohte, S. M., Kok, J. N., and La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 17–37. doi: 10.1016/S0925-2312(01)00658-0

Brumberg, J. C., Nowak, L. G., and McCormick, D. A. (2000). Ionic mechanisms underlying repetitive high-frequency burst firing in supragranular cortical neurons. *J. Neurosci.* 20, 4829–4843. doi: 10.1523/JNEUROSCI.20-13-04829.2000

Cassidy, A. S., Merolla, P., Arthur, J. V., Esser, S. K., Jackson, B., Alvarez-Icaza, R., et al. (2013). "Cognitive computing building block: a versatile and efficient digital neuron model for neurosynaptic cores," in *2013 International Joint Conference on Neural Networks (IJCNN)* (Dallas, TX: IEEE), 1–10. doi: 10.1109/IJCNN.2013.6707077

Cessac, B. (2011). A discrete time neural network model with spiking neurons: II: dynamics with noise. *J. Math. Biol.* 62, 863–900. doi: 10.1007/s00285-010-0358-4

Chakrabartty, S., and Cauwenberghs, G. (2007). Gini support vector machine: quadratic entropy based robust multi-class probability regression. *J. Mach. Learn. Res.* 8, 813–839.

Chatterjee, O., and Chakrabartty, S. (2018). Decentralized global optimization based on a growth transform dynamical system model. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 6052–6061. doi: 10.1109/TNNLS.2018.2817367

Clarke, L. E., and Barres, B. A. (2013). Emerging roles of astrocytes in neural circuit development. *Nat. Rev. Neurosci.* 14, 311–321. doi: 10.1038/nrn3484

Connors, B. W., and Gutnick, M. J. (1990). Intrinsic firing patterns of diverse neocortical neurons. *Trends Neurosci.* 13, 99–104. doi: 10.1016/0166-2236(90)90185-D

Cutsuridis, V., Cobb, S., and Graham, B. P. (2010). Encoding and retrieval in a model of the hippocampal ca1 microcircuit. *Hippocampus* 20, 423–446. doi: 10.1007/978-1-4419-0996-1

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Fields, R. D., Araque, A., Johansen-Berg, H., Lim, S.-S., Lynch, G., Nave, K.-A., et al. (2014). Glial biology in learning and cognition. *Neuroscientist* 20, 426–431. doi: 10.1177/1073858413504465

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophys. J.* 1, 445–466. doi: 10.1016/S0006-3495(61)86902-6

Friedrich, R. W., and Laurent, G. (2001). Dynamic optimization of odor representations by slow temporal patterning of mitral cell activity. *Science* 291, 889–894. doi: 10.1126/science.291.5505.889

Galán, R. F., Sachse, S., Galizia, C. G., and Herz, A. V. (2004). Odor-driven attractor dynamics in the antennal lobe allow for simple and rapid olfactory pattern classification. *Neural Comput.* 16, 999–1012. doi: 10.1162/089976604773135078

Gangopadhyay, A., and Chakrabartty, S. (2018). Spiking, bursting, and population dynamics in a network of growth transform neurons. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 2379–2391. doi: 10.1109/TNNLS.2017.2695171

Gangopadhyay, A., Chatterjee, O., and Chakrabartty, S. (2017). Extended polynomial growth transforms for design and training of generalized support vector machines. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 1961–1974. doi: 10.1109/TNNLS.2017.2690434

Gangopadhyay, A., Mehta, D., and Chakrabartty, S. (2019). A spiking neuron and population model based on the growth transform dynamical system. *bioRxiv* 523944. doi: 10.1101/523944

Gardner, E., and Derrida, B. (1988). Optimal storage properties of neural network models. *J. Phys. A Math. Gen.* 21:271. doi: 10.1088/0305-4470/21/1/031

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press. doi: 10.1017/CBO9780511815706

Gibson, J. R., Beierlein, M., and Connors, B. W. (1999). Two networks of electrically coupled inhibitory neurons in neocortex. *Nature* 402:75. doi: 10.1038/47035

Gore, A., and Chakrabartty, S. (2010). A min-max optimization framework for designing sigma-delta learners: theory and hardware. *IEEE Trans. Circuits Syst. I* 57, 604–617. doi: 10.1109/TCSI.2009.2025002

Gray, C. M., and McCormick, D. A. (1996). Chattering cells: superficial pyramidal neurons contributing to the generation of synchronous oscillations in the visual cortex. *Science* 274, 109–113. doi: 10.1126/science.274.5284.109

Hasselmo, M. (2002). "Neuromodulation in mammalian nervous systems," in *Handbook of Brain Theory and Neural Networks*, ed M. A. Arbib (MIT Press).

Hinton, G. E., Sejnowski, T. J., et al. (1986). "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol 1 (MIT Press), 282–317.

Hodgkin, A. L., and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 117, 500–544. doi: 10.1113/jphysiol.1952.sp004764

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.* 79, 2554–2558. doi: 10.1073/pnas.79.8.2554

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* 15, 1063–1070. doi: 10.1109/TNN.2004.832719

Izhikevich, E. M. (2007). *Dynamical Systems in Neuroscience*. MIT Press. doi: 10.7551/mitpress/2526.001.0001

Jonke, Z., Habenschuss, S., and Maass, W. (2016). Solving constraint satisfaction problems with networks of spiking neurons. *Front. Neurosci.* 10:118. doi: 10.3389/fnins.2016.00118

Lansner, A. (2009). Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends Neurosci*. 32, 178–186. doi: 10.1016/j.tins.2008.12.002

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F.-J. (2006). "A tutorial on energy-based learning," in *Predicting Structured Data*, eds G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar (MIT Press).

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Front. Neurosci*. 10:508. doi: 10.3389/fnins.2016.00508

Mar, D., Chow, C., Gerstner, W., Adams, R., and Collins, J. (1999). Noise shaping in populations of coupled model neurons. *Proc. Natl. Acad. Sci. U.S.A*. 96, 10450–10455. doi: 10.1073/pnas.96.18.10450

McCormick, D. A., Connors, B. W., Lighthall, J. W., and Prince, D. A. (1985). Comparative electrophysiology of pyramidal and sparsely spiny stellate neurons of the neocortex. *J. Neurophysiol*. 54, 782–806. doi: 10.1152/jn.1985.54.4.782

McCormick, D. A., and Nusbaum, M. P. (2014). Editorial overview: neuromodulation: tuning the properties of neurons, networks and behavior. *Curr. Opin. Neurobiol*. 29:4. doi: 10.1016/j.conb.2014.10.010

Mehta, D., Gangopadhyay, A., Aono, K., and Chakrabartty, S. (2019). *Growth Transform Neuron Model Matlab GUI*. Available online at: https://github.com/aimlab-wustl/growth-transform-NN

Nakano, T., Otsuka, M., Yoshimoto, J., and Doya, K. (2015). A spiking neural network model of model-free reinforcement learning with high-dimensional sensory input and perceptual ambiguity. *PLoS ONE* 10:e0115620. doi: 10.1371/journal.pone.0115620

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci*. 7:178. doi: 10.3389/fnins.2013.00178

Palm, G. (2013). Neural associative memories and sparse coding. *Neural Netw*. 37:165–171. doi: 10.1016/j.neunet.2012.08.013

Rieke, F., Warland, D., Van Steveninck, R. D. R., and Bialek, W. S. (1999). *Spikes: Exploring the Neural Code, Vol. 7*. Cambridge: MIT Press.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci*. 11:682. doi: 10.3389/fnins.2017.00682

Shrestha, S. B., and Orchard, G. (2018). "Slayer: spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, Vol. 31, eds Bengio S. et al. (Montreal, QC: Curran Associates, Inc.) 1412–1421.

Sima, J., and Orponen, P. (2003). Continuous-time symmetric hopfield nets are computationally universal. *Neural Comput*. 15, 693–733. doi: 10.1162/089976603321192130

Soula, H., Beslon, G., and Mazet, O. (2006). Spontaneous dynamics of asymmetric random recurrent spiking neural networks. *Neural Comput*. 18, 60–79. doi: 10.1162/089976606774841567

Stopfer, M., Jayaraman, V., and Laurent, G. (2003). Intensity versus identity coding in an olfactory system. *Neuron* 39, 991–1004. doi: 10.1016/j.neuron.2003.08.011

Thorpe, S. J. (1990). "Spike arrival times: a highly efficient coding scheme for neural networks," in *Parallel Processing in Neural Systems*, eds R. Eckmiller, G. Hartmann, and G. Hauske (North-Holland: Elsevier), 91–94.

Traub, R. D., and Miles, R. (1991). *Neuronal Networks of the Hippocampus, Vol. 777*. Cambridge University Press. doi: 10.1017/CBO9780511895401

Tucker, A. W., and Kuhn, H. (1951). "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (Berkeley, CA: Univ. of California Press), 481–492.

Wright, S. H. (2004). Generation of resting membrane potential. *Adv. Physiol. Educ*. 28, 139–142. doi: 10.1152/advan.00029.2004

Xin, J., and Embrechts, M. J. (2001). "Supervised learning with spiking neural networks," in *International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222) IJCNN'01*, Vol. 3 (Washington, DC: IEEE), 1772–1777.

Zenke, F., and Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput*. 30, 1514–1541. doi: 10.1162/neco_a_01086

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.