frontiers | Research Topics

Geometries of learning

Edited by
Pavan Turaga, Yunye Gong, P. Thomas Fletcher
and Ajay Divakaran

Published in
Frontiers in Computer Science
Frontiers in Artificial Intelligence
Frontiers in Big Data

ISSN 1664-8714
ISBN 978-2-8325-6261-1
DOI 10.3389/978-2-8325-6261-1

About Frontiers

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

Frontiers journal series

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

Dedication to quality

Each Frontiers article is a landmark of the highest quality, thanks to genuinely collaborative interactions between authors and review editors, who include some of the world's best academicians. Research must be certified by peers before entering a stream of knowledge that may eventually reach the public - and shape society; therefore, Frontiers only applies the most rigorous and unbiased reviews. Frontiers revolutionizes research publishing by freely delivering the most outstanding research, evaluated with no bias from both the academic and social point of view. By applying the most advanced information technologies, Frontiers is catapulting scholarly publishing into a new generation.

What are Frontiers Research Topics?

Frontiers Research Topics are very popular trademarks of the *Frontiers journals series*: they are collections of at least ten articles, all centered on a particular subject. With their unique mix of varied contributions from Original Research to Review Articles, Frontiers Research Topics unify the most influential researchers, the latest key findings and historical advances in a hot research area.

Find out more on how to host your own Frontiers Research Topic or contribute to one as an author by contacting the Frontiers editorial office: frontiersin.org/about/contact

Geometries of learning

Topic editors

Pavan Turaga — Arizona State University, United States
Yunye Gong — SRI International, United States
P. Thomas Fletcher — University of Virginia, United States
Ajay Divakaran — SRI International, United States

Citation

Turaga, P., Gong, Y., Fletcher, P. T., Divakaran, A., eds. (2025). *Geometries of learning*. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-8325-6261-1

Table of contents

frontiers | Frontiers in Computer Science

TYPE Editorial
PUBLISHED 09 April 2025
DOI 10.3389/fcomp.2025.1597931

OPEN ACCESS

EDITED AND REVIEWED BY
Marcello Pelillo,
Ca' Foscari University of Venice, Italy

*CORRESPONDENCE
Yunye Gong
yunye.gong@sri.com

RECEIVED 21 March 2025
ACCEPTED 26 March 2025
PUBLISHED 09 April 2025

CITATION
Gong Y, Turaga P, Fletcher PT and Divakaran A (2025) Editorial: Geometries of learning. *Front. Comput. Sci.* 7:1597931. doi: 10.3389/fcomp.2025.1597931

Editorial: Geometries of learning

Yunye Gong[1]*, Pavan Turaga[2], P. Thomas Fletcher[3] and Ajay Divakaran[1]

[1]Center for Vision Technologies, SRI International, Princeton, NJ, United States, [2]School of Arts Media and Engineering, Arizona State University, Tempe, AZ, United States, [3]Department of Computer Science, University of Virginia, Charlottesville, VA, United States

KEYWORDS

geometries of learning, manifold learning, deep neural networks, adversarial learning, topological data analysis

Editorial on the Research Topic
Geometries of learning

Despite the widespread application and empirical success of deep neural networks, the theoretical development of these networks remains an under-explored area. The lack of theory prevents rigorous explanation and prediction of networks' performance and hinders confident deployment of the networks especially in safety-critical applications. Geometric analysis provides a unique perspective to bridge this gap between the practice and the theory of deep neural networks. As complex real-world data often lie on low-dimensional manifolds, geometries of these underlying structures provide informative insights for understanding the behavior of networks fitting to the data.

In this Research Topic of Frontiers in Computer Science on Geometries of Learning, we aim to present the latest advances on inspecting deep learning through the lens of geometry. We have followed a rigorous peer review process and collected 12 articles that showcase the depth and breadth of current approaches across applications of geometric analysis across diverse problem domains from adversarial defense to object pose estimation. A summary of the collection is introduced below.

In the article titled "*Leveraging linear mapping for model-agnostic adversarial defense*" Jamil et al. shed light on generalization of adversarial defense by demonstrating the existence of a linear mapping between adversarial image representations across different deep neural networks. Based on this insight, the authors further demonstrate a model-agnostic adversarial defense strategy by mapping adversarial representations from diverse models into a canonical space where adversarial representation can be identified accurately using a simple linear classifier without re-training.

In the article titled "*Probabilistic and semantic descriptions of image manifolds and their applications*", Tu et al. model image manifolds as probability density functions using advanced deep generative models including normalizing flows and diffusion models. The authors further introduce a model based on the variational auto encoder to demonstrate semantic disentanglement on the image manifold and demonstrate reliable density estimation by enforcing semantic consistency as an application in adversarial robustness against patch attacks.

In the article titled "*Integrating geometries of ReLU feedforward neural networks*", Liu Y. et al. introduce a toolbox that computes geometric properties of ReLU feedforward neural networks for characterizing networks' partition of the input space. The geometric analysis framework based on polyhedral decomposition lays a foundation for comprehensive analysis of network geometries and their implications on model behavior.

In the article titled "*Locally linear attributes of ReLU neural networks*", Sattelberg et al. investigate ReLU neural networks which partitions the input space into convex polytopes. By studying the evolution of the polytope structure and affine transformations associated, the authors present insights on how the number of the polytopes can be reduced and how similar structures observed across different networks, suggesting improved network design for reducing complexity and enhancing generalization.

In the article titled "*An algorithm for computing Schubert varieties of best fit with applications*", Karimov et al. present a geometric tool based on Schubert variety to present a collection of subspaces of a fixed vector space. The authors further introduce integration of the representation as a mathematically interpretable abstract node for artificial neural networks. The proposed algorithms demonstrated on classification problems suggest the existence of a best dimension for the representative subspace, as exceeding the dimension leads to no improvement in accuracy.

In the article titled "*Exploring fMRI RDMs: enhancing model robustness through neurobiological data*", Pickard et al. introduce a large public benchmark consisting biological representations for classic vision datasets. With curated data, metrics and analysis based on representational dissimilarity matrices and Fiedler partitioning, the authors demonstrate the use of the benchmark to facilitate research on inspecting the alignment between neural network representations and neurobiological representations from fMRI and how it relates to network performance and robustness.

In the article titled "*Manifold-driven decomposition for adversarial robustness*", Zhang et al. investigate the adversarial risk and robustness-accuracy trade-off of machine learning models from the manifold perspective. Considering decomposing adversarial risk into normal adversarial risk and in-manifold adversarial risk, the authors present theoretical findings on bounding the adversarial risks as well as empirical validation of the findings on synthetic and real-world datasets.

In the article titled "*On-manifold projected gradient descent*", Mahler et al., present a novel solution for generating on-manifold adversarial data, by leveraging mathematical rigorous tools to approximate the data manifold and its tangent directions for sample perturbation. The perturbed samples are further projected back to the data manifold, resulting in adversarial samples which effectively confuse trained classifiers. The misclassification can be further explained based on the semantic basis of the manifold.

In the article titled "*Leveraging diffusion models for unsupervised out-of-distribution detection on image manifold*", Liu Z. et al. propose a novel solution for unsupervised out-of-distribution detection by performing image inpainting with in domain diffusion models. The diffusion model serves as a mapping to its training manifold and the distance between mapped image and the original image serves an indicative metric for out-of-distribution detection, as supported by empirical experiments across images with diverse characteristics.

In the article titled "*Orthogonality and graph divergence losses promote disentanglement in generative models*", Shukla et al. improve deep generative models by integrating an architecture promoting separation of latent space, an orthonormality constraint modeling statistical independence of latent attributes and a differentiable graph divergence loss promoting manifold preservation. The proposed solution achieves disentangled representations and controlled generation as demonstrated in experiments on 3D shape datasets.

In the article titled "*Implications of data topology for deep generative models*", Jin et al. study the ability of various deep generative models to model complex data topologies. With experiments using synthetic data, the authors demonstrate the limitations of models with normal assumptions on latent distribution and demonstrate improved abilities of recent models including chart auto encoders and denoising diffusion probabilistic models. The work further identifies challenges including the limitations of distribution-based metrics for assessing deep generative models with respect to capturing underlying data topologies.

In the article titled "*Recovering manifold representations via unsupervised meta-learning*", Gong et al. address the challenge of learning complex data manifold without uniformly or densely sampled data by leveraging novel episodic sampling strategies to improve auto encoders' generalization. The authors adopt topological and geometric metrics based on persistent homology to demonstrate the quantitative improvement on manifold reconstruction on 6-D object pose estimation benchmarks.

In summary, the selected papers highlight the cutting-edge developments in theories, methods, tools and datasets for geometric understanding of deep learning across different applications. Based on the insightful findings from the collection, we anticipate promising future research in multiple directions such as scaling up geometric analysis for larger network architectures including transformers which serve the core of recent developments in language modeling, promoting geometric and topological interpretation of learning with direct links to semantics, and expanding the study for a wide range of applications involving diverse data manifolds at real-world complexities.

The guest editor team would like to extend our sincere gratitude to all the authors, reviewers and the editorial team for their valuable contribution to this Research Topic. Our special thanks go to Dr. Bruce Draper for his vision and leadership in the research and discussions that laid the foundation of this Research Topic as well as his guidance and support throughout the editing process.

Author contributions

YG: Writing – review & editing, Writing – original draft. PT: Writing – original draft, Writing – review & editing. PF: Writing – original draft, Writing – review & editing. AD: Writing – review & editing, Writing – original draft.

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.

Publisher's note

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 30 October 2023
DOI 10.3389/fcomp.2023.1274832

OPEN ACCESS

EDITED BY
Pavan Turaga,
Arizona State University, United States

REVIEWED BY
Norman Tatro,
Systems and Technology Research, United States
Yunye Gong,
SRI International, United States

*CORRESPONDENCE
Huma Jamil
✉ huma.jamil@colostate.edu

RECEIVED 08 August 2023
ACCEPTED 11 October 2023
PUBLISHED 30 October 2023

CITATION
Jamil H, Liu Y, Blanchard N, Kirby M and Peterson C (2023) Leveraging linear mapping for model-agnostic adversarial defense. *Front. Comput. Sci.* 5:1274832. doi: 10.3389/fcomp.2023.1274832

Leveraging linear mapping for model-agnostic adversarial defense

Huma Jamil[1]*, Yajing Liu[2], Nathaniel Blanchard[1], Michael Kirby[1,2] and Chris Peterson[2]

[1]Department of Computer Science, Colorado State University, Fort Collins, CO, United States, [2]Department of Mathematics, Colorado State University, Fort Collins, CO, United States

In the ever-evolving landscape of deep learning, novel designs of neural network architectures have been thought to drive progress by enhancing embedded representations. However, recent findings reveal that the embedded representations of various state-of-the-art models are mappable to one another via a simple linear map, thus challenging the notion that architectural variations are meaningfully distinctive. While these linear maps have been established for traditional non-adversarial datasets, e.g., ImageNet, to our knowledge no work has explored the linear relation between adversarial image representations of these datasets generated by different CNNs. Accurately mapping adversarial images signals the feasibility of generalizing an adversarial defense optimized for a specific network. In this work, we demonstrate the existence of a linear mapping of adversarial inputs between different models that can be exploited to develop such model-agnostic, generalized adversarial defense. We further propose an experimental setup designed to underscore the concept of this model-agnostic defense. We train a linear classifier using both adversarial and non-adversarial embeddings within the defended space. Subsequently, we assess its performance using adversarial embeddings from other models that are mapped to this space. Our approach achieves an AUROC of up to 0.99 for both CIFAR-10 and ImageNet datasets.

KEYWORDS

linear mapping, adversarial defense, embedded representations, embeddings spaces, cross-model defense, convolutional neural network architectures

1. Introduction

The rapid advancements in deep learning have led to remarkable breakthroughs in various tasks, such as image recognition, natural language processing, and autonomous driving. These achievements are widely attributed to increasingly innovative designs of neural network architectures, which are believed to enhance the quality of embedded representations. However, evidence from recent research into embedded representations has found results that counter this narrative. Specifically, McNeely-White et al. (2022) found that embedded representations of inputs within state-of-the-art models can be linked via a simple linear map. The existence of this simple map suggests that, despite the architectural diversity, the learned embedded representations may not be as distinctive as previously assumed.

In this study, we investigate the potential to harness this mapping to develop robust defenses against adversarial attacks (i.e., imperceptible perturbations added to input data that cause neural networks to incorrectly process inputs; Szegedy et al., 2014b). The crux of our proposed defense is that an adversarial defense can be established for specific neural network's embedded space—then, other neural network's embedded representations can be

linearly mapped to that embedding space, leading to the detection of adversarial attacks. We define the neural network with the defense's embedding space as the **canonical embedding space**.

In order for the defense designed for the canonical embedding space to generalize to mapped inputs from other networks, adversarial inputs into other networks would need to map to the canonical space. We believe our work is the first to investigate if such a mapping would be accurate—McNeely-White et al. (2022) only experimented with mapping in-domain, unperturbed inputs from datasets like ImageNet and IJB-C. Mapping adversarial inputs between embedding spaces is a difficult problem because adversarial inputs are typically generated for a specific network. Thus, a source image that is adversarially perturbed by two different networks, resulting in one image per network, is distinct because each generated image is designed to fool a specific network. This makes the mapping problem more difficult, and requires that any adversarial defense for a canonical embedding space be robust to these differences.

Our work investigated a defense proposed by Gorbett and Blanchard (2022), utilizing a linear SVM to detect adversarial inputs specific to a particular network. The SVM training necessitates creating a dataset of potential attacks. While Gorbett and Blanchard (2022) demonstrates robustness with sufficient data, a drawback lies in the dataset requirement. By considering one network as the canonical reference and mapping other networks to that space, we overcome this limitation, needing training data solely for the canonical network. Figure 1 provides a high-level illustration of the concept.

In this research paper, we present the following key contributions:

- Successful Linear Mapping of Adversarial Inputs: We successfully establish connections between adversarial inputs across diverse CNNs using a simple linear mapping. By applying this technique to adversarial versions of MNIST, CIFAR-10, and ImageNet datasets, we achieve Mean Squared Error (MSE) scores, of mapped adversarial embeddings, going as low as approximately 0, highlighting significant similarities in the adversarial image embeddings of various CNN architectures.
- Robust Cross-Model Adversarial Detection: We develop a straightforward yet effective adversarial defense mechanism based on a linear SVM approach. Remarkably, this defense method, initially constructed for one model's embeddings, proves to be highly adept at detecting adversarial embeddings from other models as well. The achieved Area Under the Receiver Operating Characteristic (AUROC) scores, reaching up to 0.99, demonstrate the robustness and generalizability of our defense approach across different CNN architectures.
- By integrating linear mapping to build adversarial detection method, ultimately, we propose a canonical adversarial defense that accurately identifies adversarial inputs from a range of networks and adversarially manipulated datasets.

Our paper adheres to the following structure: Section 2 presents a comprehensive review of related literature concerning adversarial defense and linear mapping. In Section 3, we outline the experimental setup, encompassing the definition of linear mapping, selection of datasets, implementation of adversarial attacks, and the chosen evaluation metrics. Sections 4, 5, and 7 contain the details of conducted experiments, analysis of obtained results, and discussion. Lastly, in Section 8, we provide concluding remarks summarizing the overall findings and contributions of our research.

2. Related work

2.1. Adversarial defense

In recent years, the vulnerability of deep neural networks (DNNs) to adversarial attacks has sparked significant interest, leading to a growing body of research focused on interpreting adversarial attacks (Han et al., 2023) and devising defense and detection mechanisms (Khamaiseh et al., 2022). Various proposed methods include augmenting input images to enhance robustness against adversarial attacks (Frosio and Kautz, 2023), mapping adversarial images back to the clean distribution (Li et al., 2023), and using vector quantization (Dong and Mao, 2023). Several studies have delved into gradient-based methods, including leveraging sparse representation to counter adversarial attacks (Gopalakrishnan et al., 2018), constraining the hidden space of DNNs (Mustafa et al., 2019), and reducing the space of potential adversarial examples (Xu et al., 2017).

In parallel, researchers have also explored the utilization of manifold-related properties to address adversarial attacks. Notably, Jha et al. (2018) observed that adversarial examples tend to deviate from the data manifold as the intensity of attacks increases, and this increasing distance can serve as a valuable cue for detection. Moreover, Crecchi et al. (2019) employed the non-linear dimensionality reduction technique t-SNE to identify adversarial examples by detecting images lying outside the manifold in localized pockets. Additionally, Feinman et al. (2017) introduced kernel density and Bayesian uncertainty estimation methods for adversarial detection, using the representations of unknown data points in the last hidden layer to measure their distance within that feature space. These manifold-based approaches present promising avenues for fortifying DNNs against adversarial perturbations.

Another intriguing area for advancing adversarial defense techniques lies in the development of pre-processing methods. Recent studies by Blau et al. (2022) and Nie et al. (2022) have introduced innovative defense strategies based on diffusion processes, which prove effective in countering adversarial attacks. Qiu et al. (2021) in their work, have put forth pre-processing techniques aimed at mitigating gradient-based attacks. Additionally, Zheng et al. (2020) have proposed a model-agnostic defense approach that leverages affine transformations applied to images as a pre-processing step. Our work distinguishes itself from traditional pre-processing defense methods. Instead of modifying the input data prior to model inference, we harness the output of trained models as a foundation for constructing our adversarial defense.

The utilization of information from latent layers for detecting adversarial attacks has also received extensive attention. Bendale and Boult (2016) introduced OpenMax as an alternative to the

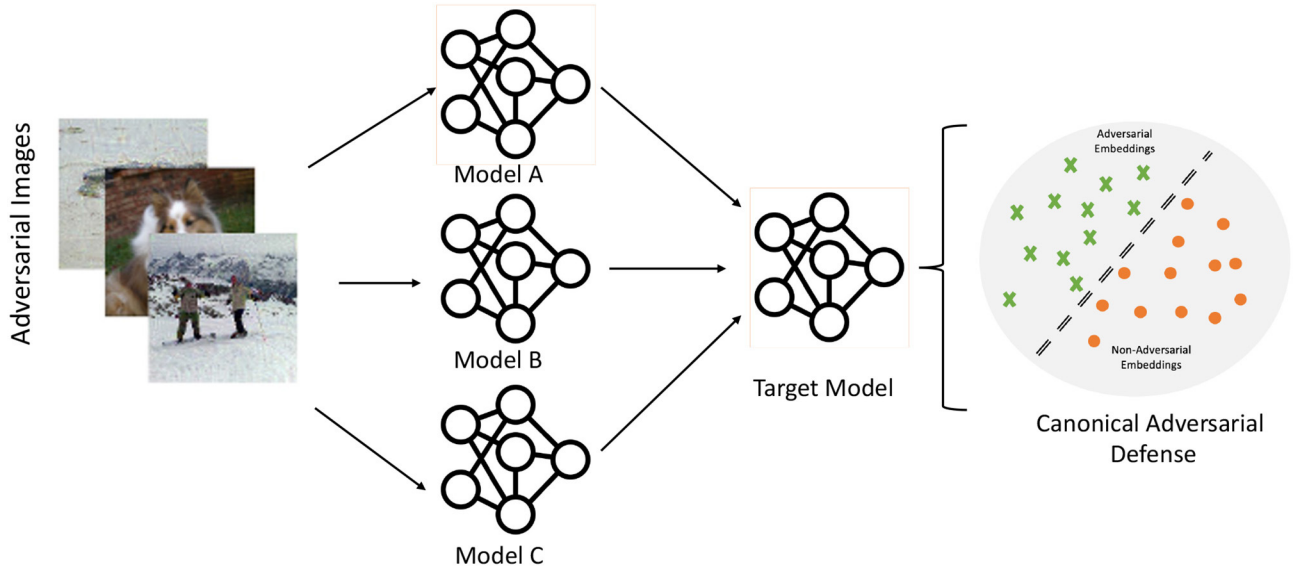


FIGURE 1
The idea of a model-agnostic defense: Adversarial embeddings from different models are mapped to a target model with a defended space. The mapped adversarial embeddings can then be effectively detected from non-adversarial embeddings using a simple linear classifier trained on the adversarial and non-adversarial embeddings of the target model.

softmax layer, leveraging penultimate layer information to identify unknown classes. Li and Li (2017) employed convolution layer outputs to develop a cascade method for detecting adversarial examples. In contrast, Gorbett and Blanchard (2022) demonstrated that only penultimate layers carry sufficient information to distinguish adversarial from non-adversarial images. Furthermore, Jamil et al. (2023) highlighted the utility of intermediate ReLU activation patterns for detecting adversarial images. These diverse approaches underscore the significance of using latent layer information for robust adversarial detection.

Our work aligns with the latter research endeavors, where we capitalize on information from the penultimate layer to construct a shared embedding space for various DNNs. This shared space exhibits potential as a robust fortress for adversarial defense. By mapping adversarial embeddings from different DNNs onto this canonical space, our aim is to create a generalized defense mechanism against adversarial attacks. This approach holds promise for strengthening the security and robustness of deep neural networks in the face of adversarial perturbations.

2.2. Linear mapping

Some researchers have directed their efforts toward emphasizing the commonalities existing between different DNN architectures concerning the learned features. For instance, in Lenc and Vedaldi (2015), a comparison of the hidden representations of DNNs in the convolutional layers was carried out through regression analysis. Notably, a series of studies conducted by McNeely-White et al. (2020), McNeely-White et al. (2021), and McNeely-White et al. (2022), established a relationship among DNNs by demonstrating that their hidden representations are essentially similar, up to a single linear transformation. McNeely-White et al. (2022) delved into the implications of these linear mappings in the context of biometric security.

In our work, we build upon this concept and extend it to the domain of adversarial attacks. We aim to investigate whether the hidden network representations for adversarial data can also be effectively mapped from one model's embedding space to another model's space. By exploring the feasibility of such cross-model mappings, we seek to uncover potential insights that may facilitate the development of more robust adversarial defense strategies.

3. Experimental setup

This section presents the methodology for establishing a linear mapping between a source network and a target network (see Section 3.1). Additionally, it encompasses the details of the datasets utilized in this study (see Section 3.2), the employed adversarial attacks (see Section 3.3), and the evaluation metrics (see Section 3.4) used to assess the transferability of adversarial features.

3.1. Linear mapping

Given f_A and f_B as the source and target networks, respectively, and X as the set of input images, we define a linear map denoted by $M_{A\to B}$ as follows:

$$\tilde{f}_B(X) = M_{A\to B} f_A(X), \tag{1}$$

where $\tilde{f}_B(X)$ is the best approximation to $f_B(x)$ across a given dataset, and the linear mapping $M_{A\to B}$ is computed by solving a

least square regression problem as follows:

$$\underset{\tilde{M}_{A\to B}}{\text{minimize}} \sum_{i=1}^{m} \|\tilde{M}_{A\to B} f_A(x) - f_B(x)\|_2. \quad (2)$$

3.2. Datasets

To assess the transferability of adversarial information between different architectures, we conduct a series of experiments. We begin by testing with a simple dataset such as MNIST, and subsequently, we validate its generalizability by extending the evaluation to more complex datasets like CIFAR-10 and ImageNet. This section provides an overview of the datasets utilized in the experiments, as well as a detailed explanation of the methodology used for generating the adversarial attacks.

3.2.1. MNIST

The MNIST dataset comprises a collection of 70,000 grayscale images of handwritten digits, each measuring 28 × 28 pixels. The dataset is further partitioned into a training set, containing 60,000 images, and a test set, containing 10,000 images. These images are categorized into 10 classes, representing the digits from 0 to 9.

3.2.2. CIFAR-10

The CIFAR-10 dataset comprises 60,000 images that are organized into 10 distinct classes. For training purposes, there are 50,000 images, and an additional 10,000 images are allocated for testing. Each image within the dataset measures 32 × 32 pixels.

3.2.3. ImageNet

In this study, we utilized the validation set from the ImageNet dataset, consisting of 50,000 images spread across 1000 classes, with 50 images per class.

For the MNIST and CIFAR-10 datasets, we employed the training set to train the models. Subsequently, the test set was utilized to calculate the linear mapping and assess the transferability of adversarial attacks.

As for the Imagenet dataset, we performed a train-test split on the validation set. The training set was utilized to compute the linear mapping, while the test set was employed to evaluate the transferability of adversarial attacks.

3.3. Adversarial attacks

We generate adversarial datasets corresponding to MNIST, CIFAR-10, and ImageNet datasets using the following adversarial attack techniques.

3.3.1. Fast gradient sign method

The Fast Gradient Sign Method (FGSM) (Szegedy et al., 2014b) is an efficient one-step adversarial attack technique. It introduces small perturbations δ to the input data x based on the gradient ∇_x of the loss function J with respect to the input. The perturbations are scaled by a small positive scalar ϵ, and their direction is determined by the sign of the gradient. This method causes misclassification by the targeted machine learning model. The mathematical representation is as follows:

$$\delta = \epsilon \cdot \text{sign}\left(\nabla_x J\left(\Phi, x, y\right)\right), \quad (3)$$

where ϵ is the scaling factor and sign denotes the sign function. This technique is widely used for crafting adversarial examples to evaluate the robustness of machine learning models.

3.3.2. Projected gradient descent

In contrast to FGSM, Projected Gradient Descent (PGD) (Madry et al., 2017) is an iterative adversarial attack method that computes perturbations using gradients and then restricts them within a specified perturbation bound. This iterative approach leads to more robust attacks as the perturbations are constrained to remain within an acceptable range. The iterative perturbation can be expressed as follows, where α represents the perturbation step size:

$$\delta_{t+1} = \text{clip}_\epsilon(\delta_t + \alpha \cdot \text{sign}\left(\nabla_x J\left(\Phi, x + \delta_t, y\right)\right)). \quad (4)$$

Here, clip_ϵ denotes a function that ensures the perturbations stay within the specified range, ϵ.

3.3.3. Carlini and Wagner attack

The Carlini and Wagner attack (Carlini et al., 2019) is an optimization-based adversarial technique that efficiently finds small perturbations to cause misclassification in a fixed input image. By minimizing a distance metric between the original and perturbed images, the attack strikes a balance between misclassification confidence, perturbation size, and the distance norm. This approach aims to generate potent and subtle adversarial perturbations for evading machine learning models effectively.

3.3.4. DAmageNet

DAmageNet (Chen et al., 2019) presents a transferable adversarial attack strategy that leverages attention heatmaps to create universal adversarial samples. By directing the attention to irrelevant regions in the image, it induces misclassification, taking advantage of shared attention patterns across diverse deep neural networks. This technique has proven to be effective in causing misclassification across a wide range of models, demonstrating its potency in generating robust adversarial samples.

In our experimental setup, we employed the Fast Gradient Sign Method (FGSM) attack to create adversarial datasets for the MNIST, CIFAR-10, and ImageNet validation datasets. For each dataset, we set the perturbation magnitude (ϵ) to the values of 0.02, 0.05, and 0.01, respectively.

However, for the ImageNet experiments, we extended our evaluation to include more sophisticated attacks, such as PGD, C&W attack, and the DamageNet attack. The adversarial dataset corresponding to the ImageNet validation set, generated with the DamageNet attack, was directly obtained from reliable source on the web.

3.4. Testing and evaluation metrics

3.4.1. Mean squared error

To evaluate the effectiveness of a linear mapping, we use the Mean Squared Error (MSE) metric. This involves calculating the MSE between the target embeddings and their corresponding linearly mapped source embeddings. For each pair of embeddings, we compute the squared differences between their elements, sum up these squared differences, and then take the average across all pairs. This resulting average MSE score represents how well the linear mapping transforms embeddings from one space to another. Lower MSE values indicate a more accurate and robust linear mapping.

3.4.2. Linear SVM classifier

To assess the effectiveness of linear mapping for deep neural networks with adversarial images, we adopt a linear SVM classifier. As demonstrated in a prior study (Gorbett and Blanchard, 2022), adversarial image embeddings can be distinguished from non-adversarial image embeddings using a linear SVM. To evaluate the mapping's efficacy, we train a linear SVM classifier on embeddings generated by one model to discern between adversarial and non-adversarial image embeddings. Additionally, we investigate whether the mapped adversarial embeddings, linearly transformed from a different network's embedding space to the target network's space, remain distinguishable from the non-adversarial image embeddings. To quantify the SVM's performance, we measure the area under the receiver operating characteristic curve (AUROC) metric.

4. Experiments

For CIFAR-10 and MNIST datasets, each comprising two sets: train and test. These sets consist of original images and their corresponding adversarial counterparts generated using the FGSM attack mentioned in Section 3.4.1.

In the case of ImageNet, we utilize pretrained models, leading to a dataset that solely contains test data. This dataset encompasses both original images and their corresponding adversarial images crafted through all the mentioned adversarial attacks given in Section 3.3.

For MNIST, we use two straightforward architectures: one comprising convolution layers and the other a feed-forward neural network (FFNN). For CIFAR-10, we employ EfficientNet (Tan and Le, 2020), ResNet-18 (He et al., 2015), MobileNetV2 (Sandler et al., 2019), GoogLeNet (Szegedy et al., 2014a), and VGG-19 (Simonyan and Zisserman, 2015) architectures. These networks were trained solely on the original images from the MNIST and CIFAR-10 training sets, respectively. In the case of ImageNet, we use pre-trained models, namely ResNet-50, ResNet-101, ResNet-152 (He et al., 2015), VGG-19, Inception-v3 (Szegedy et al., 2014a) and Vision Transformer (Dosovitskiy et al., 2020), which have been trained on the ImageNet training dataset. The classification accuracy on original test dataset and corresponding FGSM adversarial dataset of each of these models, evaluated using the test set, along with the size of the penultimate layer, are summarized in Table 1.

In these experiments, we initially construct a linear classifier trained to discriminate between the adversarial embeddings and original embeddings of a target model. Subsequently, we demonstrate that the trained classifier also effectively distinguishes these original embeddings from adversarial embeddings that are mapped from a source model to this target model.

To conduct this investigation, we designate a model as the target model and proceed to map the adversarial and original features from the embedding spaces of other models to the embedding space of this target model using MNIST and CIFAR-10 datasets with specified neural architectures. We then expand this approach to the ImageNet dataset, employing one neural architecture as the target model and several other architectures as source models. The experiment involves dividing the validation dataset into three distinct splits: map, svm, and val. For each split, we calculate the embeddings of the networks from their penultimate layer, which are denoted as X^A_{map}, X^A_{svm}, X^A_{val}, with A representing the network under consideration.

Given two networks, a source network denoted by s and a target network denoted by t, we train a linear SVM using the embeddings X^t_{svm} from the target network. This SVM classifier is trained with binary labels to distinguish between adversarial and original data. Then, we learn a linear mapping denoted as $M_{s\to t}$ that aligns the embeddings X^s_{map} with X^t_{map} by solving model (2). Subsequently, we obtain the mapped embeddings $X^{'}_{\text{val}}$ by applying $M_{s\to t}$ to the validation data, X^s_{val}, i.e., $X^{'}_{\text{val}} = M_{s\to t}X^s_{\text{val}}$. We then measure the strength of this mapping using the MSE metric, as mentioned in Section 3.4.1.

To assess adversarial transferability, we replace the validation data of target network, X^t_{val}, with $X^{'}_{\text{val}}$ obtained from linear mapping process. Next, we evaluate the modified data on the linear SVM trained on the target model's embeddings. The extent of transferability is quantified by measuring the SVM classification accuracy on the data which is the source model's adversarial and original embeddings after being linearly mapped to the target model's embedding space (see Section 3.4.2). The procedure is formally described in Algorithm 1.[1]

5. Results

5.1. Linear mapping for MNIST

Initially, we investigated the feasibility of linearly mapping adversarial images from the MNIST dataset and using a linear SVM to identify the adversarial images. Using a convolutional neural network (CNN) and a feed-forward neural network (FFNN) and evaluating mapping between both, we found the SVM was able to accurately identify adversarial images with 99.4% accuracy (CNN $\to$ FFNN) and 99.5% accuracy (FFNN $\to$ CNN) with $\epsilon = 0.02$—notably, these results were consistent across various epsilon values. This mirrors the accuracy of the SVM's performance on the original embedding space (99.8% for CNN; 99.9% for FFNN).

1 The "+" in step 1 of the Algorithm 1 indicates the concatenation of two sets.

TABLE 1 Classification accuracies for DNNs trained and evaluated on MNIST, CIFAR-10, and ImageNet and their corresponding adversarial datasets.

Datasets	Model names	Latent layer dimension size	Classification accuracy	
			Original data (%)	Adversarial data (%)
MNIST	CNN	128	99.16	88.42
	FFNN	128	97.98	58.27
CIFAR-10	EfficientNet-B0	320	85.20	39.31
	ResNet-18	512	95.42	44.09
	VGG-19	512	93.51	48.41
	MobileNet-V2	1,280	92.85	41.13
	GoogLeNet	1,024	95.75	45.49
ImageNet	ResNet-50	2,048	75.68	48.50
	ResNet-101	2,048	76.92	64.41
	ResNet-152	2,048	78.08	66.49
	VGG-19	25,088	72.16	59.43
	Inception-V3	2,048	77.20	66.37
	Vision Transformer	768	81.02	74.18

The adversarial data is created using FGSM attack with $\epsilon = 0.02, 0.05,$ and 0.01, respectively.

Require: X^i_{map}, X^i_{svm}, X^i_{val} where $i = t, s$
Ensure: $\text{AUROC}_{\text{svm}}$
1) Identify:
$X^i_{\text{map}} = X\text{org}^i_{\text{map}} + X\text{adv}^i_{\text{map}}$
$X^i_{\text{svm}} = X\text{org}^i_{\text{svm}} + X\text{adv}^i_{\text{svm}}$
$X^i_{\text{val}} = X\text{org}^i_{\text{val}} + X\text{adv}^i_{\text{val}}$

2) Train a linear SVM with $X\text{adv}^i_{\text{svm}} + X\text{org}^i_{\text{svm}}$ for $i = t$
3) Learn a linear mapping $M_{s\to t}$ using (2) with $f_A(x) = X^s_{\text{map}}$ and $f_B(x) = X^t_{\text{map}}$, calculate $X'_{\text{val}} = M_{s\to t} X^s_{\text{val}}$
4) Evaluate SVM with $X'_{\text{val}} = X\text{adv}'_{\text{val}} + X\text{org}'_{\text{val}}$
5) Calculate $\text{AUROC}_{\text{svm}}$

Algorithm 1. Cross-network adversarial mapping and detection.

5.2. Linear mapping for CIFAR-10

Table 2 presents the MSE scores between the adversarial embeddings of the target space and the ones mapped from the source model embedding space to the target model embedding space. The recorded lowest MSE values are 0.003 and 0.004 when adversarial embeddings are mapped to the space of GoogLeNet and MobileNet, respectively. Even when other models are considered as the target model, the MSE values remain remarkably low, indicating the overall efficiency of the linear mappings.

The architectures employed in the experiments demonstrate varying accuracies on non-adversarial and adversarial data (Table 1). EfficientNet-B0, with an accuracy of only 85%, is particularly vulnerable to adversarial attacks, achieving a mere 39.31% accuracy when exposed to such perturbations. Table 3 illustrates that when adversarial embeddings are mapped to the low-performing EfficientNet-B0, its ability to distinguish adversarial images from non-adversarial ones decreases. On the contrary, for ResNet-18, which exhibits better classification accuracy on original data, the mapped adversarial embeddings retain more distinctive features, enabling their effective separation from original images. These results show that the separability of adversarial embeddings seems to be contingent on the model's ability to classify the original data accurately. In other words, if the model performs well in classifying the original data, it tends to achieve better separability of adversarial embeddings as well.

These findings were obtained using the FGSM adversarial attack with $\epsilon = 0.05$. However, the transferability of adversarial features appears to be less dependent on the perturbation level, consistent with our observations from the MNIST experiments.

5.3. Linear mapping for ImageNet

Building upon the insights gained from MNIST and CIFAR-10, where linear mapping for adversarial data between different CNN architectures proved effective, we propose a shared embedding space concept, leveraging the ImageNet dataset. Specifically, we utilize ResNet-152's penultimate layer as the shared space, mapping adversarial embeddings from other networks onto it. Table 4 shows the MSE scores between the adversarial embeddings generated by ResNet-152 and the ones mapped to it from various CNNs. The overall MSE values are very low, except for VGG-19 where we observe notably higher MSE scores. We hypothesize that this discrepancy can be attributed to the high dimensionality of the VGG-19 embedding space. This trend appears to persist across our subsequent experiments, suggesting a consistent challenge posed by the intricately layered embedding space of VGG-19. Interestingly, the attention based model, ViT, shows very low MSE scores, signifying linear mapping can be easily learnt between CNN and attention based architectures.

TABLE 2 MSE scores (3.4) for the trained linear mapping between *source* and *target* adversarial embeddings for CIFAR-10.

Source / Target	EfficientNet-B0	ResNet-18	VGG-19	MobileNet	GoogLeNet
EfficientNet-B0	0.000	0.017	0.022	0.017	0.018
ResNet-18	0.020	0.000	0.011	0.013	0.010
VGG-19	0.037	0.018	0.000	0.025	0.021
MobileNet	0.006	0.004	0.005	0.000	0.004
GoogLeNet	0.007	0.003	0.004	0.004	0.000

TABLE 3 AUROC scores for classification of mapped original and adversarial image embeddings from the *source* model using a linear SVM trained on *target* model's embeddings for CIFAR-10 dataset.

Source / Target	EfficientNet-B0	ResNet-18	VGG-19	MobileNet	GoogLeNet
EfficientNet-B0	0.907	0.942	0.886	0.899	0.922
ResNet-18	0.907	0.995	0.939	0.979	0.988
VGG-19	0.899	0.979	0.943	0.974	0.972
MobileNet	0.904	0.991	0.936	0.989	0.983
GoogLeNet	0.902	0.987	0.928	0.978	0.985

An AUROC score of 1 indicates the best performance, with values close to 1 considered indicative of good classification performance.

TABLE 4 MSE scores for the linear mapping trained on original and adversarial embeddings for ImageNet between various models and ResNet-152.

Test data	Adversarial attacks			
	FGSM	PGD	DamageNet	C&W
ResNet-50 → ResNet-152	0.0672	0.0871	0.0851	0.0882
ResNet-101 → ResNet-152	0.0583	0.0747	0.0792	0.0813
VGG-19 → ResNet-152	0.817	0.8439	0.8127	0.8971
Inception-V3 → ResNet-152	0.1058	0.1084	0.1124	0.1146
ViT → ResNet-152	0.1368	0.1423	0.1508	0.144

The arrow indicates the direction of the linear map, with ResNet-152 as the *target* model and all other models serving as *source* models.

TABLE 5 AUROC scores for classification of mapped original and adversarial image embeddings from various models using a linear SVM trained on ResNet-152 model's embeddings.

Test data	Adversarial attacks			
	FGSM	PGD	DAmageNet	C&W
ResNet-152	0.9885	0.9932	0.9995	0.999
ResNet-50 → ResNet-152	0.9874	0.9913	0.9991	0.999
ResNet-101 → ResNet-152	0.9862	0.9926	0.9993	0.999
VGG-19 → ResNet-152	0.7584	0.7453	0.8572	0.8764
Inception-V3 → ResNet-152	0.9635	0.9434	0.9721	0.9886
ViT → ResNet-152	0.9721	0.9686	0.9959	0.9955

The arrow indicates the direction of the linear map, with ResNet-152 as the *target* model and all other models serving as *source* models.

Moreover, Table 5 presents the AUROC values obtained from the linear SVM classification. Firstly, it shows the results for ResNet-152 generated using adversarial and non-adversarial embeddings (1st row). Subsequently, it demonstrates the performance of the linear SVM when applied to the adversarial embeddings mapped to ResNet-152 space from other models' embedding space. The AUROC scores range from 0.75 to 0.99, which represents an impressive range, highlighting the excellent performance of the linear SVM. These results demonstrate that different DNNs learn similar adversarial features, facilitating successful mapping and accurate detection using a binary classifier.

We also perform the similar experiment while making the ViT as the target model and mapped the embeddings from all CNN architectures to its space. The AUROC scores for the SVM detection is given is Table 6. It is interesting to observe that the attention based architecture does not affect the quality of learnt linear map and the results are consistent with when the embeddings were mapped to ResNet-152's space.

Remarkably, the linear mapping is trained exclusively on adversarial images; however, during detection, when differentiating adversarial embeddings from other models with non-adversarial embeddings from ResNet-152, the SVM performs comparably or even better. This observation indicates that adversarial information within an image can be linearly transferred, alongside the image embeddings themselves, to a different CNN space. Consequently, this idea hints at the potential for a unified and transferable representation of adversarial features across diverse DNN architectures within the ImageNet dataset.

5.4. Mapped embeddings adversarial classification accuracy

To assess whether the embeddings, once mapped to the target network, maintain their adversarial nature, we conducted an experiment to measure the classification accuracies achieved by the target models. Specifically, we recorded the classification accuracies of the target models when provided with embeddings mapped to their respective embedding spaces. The results are presented in Table 7, showcasing both the average accuracies across different source models and their standard deviations.

Table 7 reveals a notable trend—the classification accuracy of the target models closely aligns with their adversarial accuracies (as indicated in column 3 of Table 7). This observation underscores the effectiveness of our mapping approach in preserving adversarial characteristics during the transfer.

However, it's worth noting that when considering the ImageNet dataset, we observed a higher standard deviation, particularly due to the mapping process from the VGG-19 embedding space. This outcome can be attributed to the inherent challenges posed by the substantial disparity in dimensions between the source and target embeddings, a point previously discussed.

TABLE 6 AUROC scores for classification of mapped original and adversarial image embeddings from various models using a linear SVM trained on Vision Transformer (ViT) model's embeddings.

Test data	Adversarial attacks			
	FGSM	PGD	DAmageNet	C&W
ViT	0.9799	0.9755	0.9973	0.9972
ResNet-50 → ViT	0.9815	0.9837	0.9978	0.9992
ResNet-101 → ViT	0.9793	0.9850	0.9985	0.9991
ResNet-152 → ViT	0.9797	0.9849	0.9986	0.9992
VGG-19 → ViT	0.7083	0.7001	0.8096	0.8106
Inception-V3 → ViT	0.9610	0.9385	0.9707	0.9863

The arrow indicates the direction of the linear map, with ViT as the *target* model and all other models serving as *source* models.

6. Comparison with other method

Our proposed method is the first to demonstrate the existence of a linear mapping between adversarial image representations of two models and leverages this insight to construct a model-agnostic adversarial defense.

We illustrate the possibility of this linear mapping using a simple baseline—a linear Support Vector Machine (SVM)—to create a model-agnostic detection method. We compare our baseline with the adversarial detection method proposed by Harder et al. (2021). The experiments in this section provide a comparative analysis. Specifically, we calculate the magnitude and phase of Fourier transforms for the penultimate layer embeddings and utilize them to establish a linear mapping. We then proceed to train a linear SVM classifier, following the procedure outlined in Algorithm 1.

To facilitate a comprehensive comparison, we selected ResNet-152 and Vision Transformer (ViT) as our target models, and mapped the Magnitude Fourier Spectrum (MFS) and Phase Fourier Spectrum (PFS) embeddings from various source models into the spaces of these target models. Our findings, as presented in Tables 8, 9, offer valuable insights.

Notably, when utilizing the mapped MFS embeddings, we observed that linear SVM did not exhibit strong performance, as indicated by relatively low AUROC scores across all adversarial attacks. In contrast, our method, which involves directly learning the linear mapping using the model's native embeddings, consistently outperformed the mapped MFS approach.

Furthermore, our analysis reveals that PFS presents an intriguing facet. It demonstrates superior performance when compared to mapped MFS embeddings, suggesting that phase information is amenable to linear mapping. However, it's worth highlighting an interesting observation: while PFS performs well within the realm of CNN-based architectures, its efficacy diminishes when applied to the mapping from attention-based architectures to CNN-based ones, as evidenced by lower AUROC scores in Table 8.

Notably, when all models are mapped to ViT space, the performance of PFS exhibits a slight decrease (see Table 9) compared to our proposed method. This underscores the adaptability and robustness of our approach, particularly in scenarios involving attention-based architectures.

TABLE 7 Adversarial classification accuracies of embeddings from different models mapped to the target model for FGSM attack.

Datasets	Target models	Adversarial accuracy of target model	Mapped accuracy for adversarial dataset
Cifar-10	EfficientNet-B0	39.31	41.90 ± 1.48
	ResNet-18	44.09	40.55 ± 4.83
	VGG-19	48.41	39.6 ± 4.02
	MobileNet	41.13	42.125 ± 3.24
	GoogleNet	45.49	40.825 ± 4.134
ImageNet	ResNet-152	66.49	52.30 ± 19.21
	ViT	74.18	46.75 ± 20.01

Accuracies are represented as the average of various accuracies with the standard deviations. The high standard deviation in ResNet-152 and Vision Transformer comes from the poor performance of VGG-19 mapped to ResNet-152 and ViT having accuracies (16.6 and 8.9%), respectively.

TABLE 8 AUROC scores for classification of mapped original and adversarial image magnitude Fourier spectrum (MFS) and phase Fourier spectrum (PFS) from various models using a linear SVM trained on ResNet-152 model's embeddings.

Test data	Adversarial attacks							
	FGSM		PGD		DAmageNet		C&W	
	MFS	PFS	MFS	PFS	MFS	PFS	MFS	PFS
ResNet-152	0.823	0.966	0.816	0.967	0.958	0.995	0.986	0.998
ResNet-50 → ResNet-152	0.804	0.969	0.820	0.954	0.945	0.993	0.985	0.999
ResNet-101 → ResNet-152	0.793	0.961	0.778	0.961	0.942	0.995	0.978	0.999
VGG-19 → ResNet-152	0.562	0.692	0.548	0.692	0.658	0.779	0.702	0.811
Inception-V3 → ResNet-152	0.621	0.934	0.514	0.514	0.803	0.940	0.769	0.968
ViT → ResNet-152	0.606	0.894	0.544	0.868	0.803	0.980	0.721	0.965

The arrow indicates the direction of the linear map, with ResNet-152 as the *target* model and all other models serving as *source* models.

TABLE 9 AUROC scores for classification of mapped original and adversarial image magnitude Fourier spectrum (MFS) and phase Fourier spectrum (PFS) from various models using a linear SVM trained on Vision Transformer (ViT) model's embeddings.

Test data	Adversarial attacks							
	FGSM		PGD		DAmageNet		C&W	
	MFS	PFS	MFS	PFS	MFS	PFS	MFS	PFS
ViT	0.577	0.884	0.549	0.858	0.813	0.980	0.676	0.963
ResNet-50 → ViT	0.614	0.947	0.521	0.921	0.865	0.989	0.835	0.993
ResNet-101 → ViT	0.595	0.937	0.562	0.925	0.858	0.991	0.841	0.992
ResNet-152 → ViT	0.604	0.934	0.565	0.922	0.847	0.990	0.834	0.993
VGG-19 → ViT	0.524	0.639	0.499	0.616	0.594	0.737	0.562	0.741
Inception-V3 → ViT	0.549	0.895	0.483	0.835	0.704	0.935	0.703	0.950

The arrow indicates the direction of the linear map, with ViT as the *target* model and all other models serving as *source* models.

In summary, our exploration of MFS and PFS mappings reveals interesting results. While PFS demonstrates promise, especially within the CNN domain, our method of direct linear mapping using model embeddings consistently delivers superior performance across various model architectures and adversarial attacks.

We also observed a notable variation in results when using different adversarial attack methods. For instance, the performance is better with DAmageNet images, likely due to their higher level of perturbation (MSE = 2.97 across the dataset) compared to FGSM (0.013), PGD (1.83), and C&W (1.05).

7. Discussion

To our knowledge, this is the first work to establish that adversarial features can be efficiently mapped between diverse DNN architectures. This novel discovery indicates the feasibility of creating a robust canonical embedding space that is resistant to adversarial inputs. This involves mapping adversarial embeddings from other DNNs to this canonical embedding and utilizing the canonical defense for identifying adversarial inputs. In this work, we establish the feasibility of a simple model-agnostic defense using an SVM—however, future work needs to explore the feasibility of alternative solutions for adversarial defense.

It is important to note that while this mapping does require data from the source model during its establishment, it subsequently enables the efficient detection of adversarial inputs. This detection process involves a minimal computational overhead, primarily consisting of matrix multiplication. The distinct advantage of our approach lies in its model-agnostic nature, allowing multiple models to achieve robustness against adversarial attacks through a shared and efficient detection mechanism. In contrast, model-dependent defense methods, although also requiring access to data, are inherently tied to specific model architectures and demand customization for each model.

These linear mappings raise intriguing possibilities for understanding the learned representations across different modalities, such as linking vision and language representations. Moreover, the implications extend to leveraging these mappings for practical purposes. For instance, mapping image embeddings to language embeddings may enhance the performance of language models in their respective tasks.

Our work also provide valuable insights into how one could consider the influence of architecture on a learned representation. If high performing models have ultimately learned similar representations, areas like neural architecture search (NAS) may consider shifting their focus to identifying higher performing representations—there is some work in this domain, such as the hypothesis by Blanchard et al. (2019) that learned representations that mirror biology, by grouping similar-looking objects in the

embedded representations, enhance robustness. The methodology for evaluating this shift in focus has been established by works like Radford et al. (2021), who evaluated their learned representation by testing generalization to new tasks in a zero-shot context.

There are of course further investigations that need to be done for non-traditional training paradigms and architectures.[2] For example, what is the feasibility linear of mapping between generative models, such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and CNNs? Do linear methods suffice for mapping from VAE spaces to CNN spaces, or are non-linear methods required? Despite initial appearances suggesting differences in the organization of VAEs' latent spaces, exploring the degree of dissimilarity from a linear relationship could yield valuable insights. Understanding the connections between these distinct embedding spaces will open avenues for leveraging the respective strengths of generative models and CNNs. Are there hybrid approaches that capitalize on the unique capabilities of each architecture? How can hybrid approaches facilitate the development of more powerful and adaptable AI systems?

Overall, our novel findings offer valuable insights into the interplay between adversarial features and neural network embeddings. This work paves the way for investigating novel model-agnostic defense strategies that transcend the limitations of individual architectures. Such defenses may enable more robust and reliable deep learning systems in the face of adversarial challenges.

8. Conclusion

In this study, we showcase the remarkable shared commonality in representations of adversarial images across a diverse set of deep neural networks (DNNs). This interchangeability is made possible through a straightforward linear mapping technique, typically using the DNNs penultimate layers. To our knowledge, this is the first work to establish that adversarial inputs are mappable across DNNs. Further, we capitalize on our novel finding to introduce the concept of a model-agnostic adversarial defense that leverages the transferability of adversarial features across representations. We develop a canonical adversarial defense, map adversarial embeddings from other models to that canonical space, and show adversarial inputs can be accurately identified without any additional training. The feasibility of linearly transforming adversarial features presents promising prospects for developing a more robust model-agnostic adversarial defense, provides insights for understanding and evaluating learned representations, and opens the door for a wealth of future research that capitalizes on these linear mappings.

2 Assuming the "traditional" paradigm is architectures trained for classification.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

HJ: Conceptualization, Methodology, Writing—original draft, Writing—review & editing. YL: Conceptualization, Supervision, Writing—review & editing. NB: Conceptualization, Formal analysis, Supervision, Writing—review & editing. MK: Funding acquisition, Project administration, Supervision, Writing—review & editing. CP: Supervision, Writing—review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was supported by the DARPA Geometries of Learning Program under Award No. HR00112290074.

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.

Publisher's note

References

Bendale, A., and Boult, T. E. (2016). "Towards open set deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (LasVegas, NV).

Blanchard, N., Kinnison, J., RichardWebster, B., Bashivan, P., and Scheirer, W. J. (2019). "A neurobiological evaluation metric for neural network model search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA).

Blau, T., Ganz, R., Kawar, B., Bronstein, A., and Elad, M. (2022). Threat model-agnostic adversarial defense using diffusion models. *arXiv preprint arXiv:2207.08089*. doi: 10.48550/arXiv.2207.08089

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., et al. (2019). On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*. doi: 10.48550/arXiv.1902.06705

Chen, S., Huang, X., He, Z., and Sun, C. (2019). DAmageNet: a universal adversarial dataset. *arXiv preprint arXiv:1912.07160*. doi: 10.48550/arXiv.1912.07160

Crecchi, F., Bacciu, D., and Biggio, B. (2019). Detecting adversarial examples through nonlinear dimensionality reduction. *arXiv preprint arXiv:1904.13094*. doi: 10.48550/arXiv.1904.13094

Dong, Z., and Mao, Y. (2023). Adversarial defenses via vector quantization. *arXiv preprint arXiv:2305.13651*. doi: 10.48550/arXiv.2305.13651

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. doi: 10.48550/arXiv.2010.11929

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. (2017). Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*. doi: 10.48550/arXiv.1703.00410

Frosio, I., and Kautz, J. (2023). "The best defense is a good offense: adversarial augmentation against adversarial attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Vancouver, BC), 4067–4076.

Gopalakrishnan, S., Marzi, Z., Madhow, U., and Pedarsani, R. (2018). Combating adversarial attacks using sparse representations. *arXiv preprint arXiv:1803.03880*. doi: 10.48550/arXiv.1803.03880

Gorbett, M., and Blanchard, N. (2022). "Utilizing network features to detect erroneous inputs," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (Waikoloa, HI), 34–43.

Han, S., Lin, C., Shen, C., Wang, Q., and Guan, X. (2023). Interpreting adversarial examples in deep learning: a review. *ACM Comput. Surv.* 55, 1–38. doi: 10.1145/3594869

Harder, P., Pfreundt, F.-J., Keuper, M., and Keuper, J. (2021). "Spectraldefense: detecting adversarial attacks on CNNs in the fourier domain," in *2021 International Joint Conference on Neural Networks (IJCNN)* (Shenzhen: IEEE), 1–8.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*. doi: 10.1109/CVPR.2016.90

Jamil, H., Liu, Y., Caglar, T., Cole, C., Blanchard, N., Peterson, C., et al. (2023). "Hamming similarity and graph Laplacians for class partitioning and adversarial image detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Vancouver, BC), 590–599.

Jha, S., Jang, U., Jha, S., and Jalaian, B. (2018). "Detecting adversarial examples using data manifolds," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)* (Los Angeles, CA).

Khamaiseh, S. Y., Bagagem, D., Al-Alaj, A., Mancino, M., and Alomari, H. W. (2022). Adversarial deep learning: a survey on adversarial attacks and defense mechanisms on image classification. *IEEE Access* 10, 102266–102291. doi: 10.1109/ACCESS.2022.3208131

Lenc, K., and Vedaldi, A. (2015). "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Boston, MA), 991–999.

Li, J., Zhang, S., Cao, J., and Tan, M. (2023). Learning defense transformations for counterattacking adversarial examples. *Neural Netw.* 164, 177–185. doi: 10.1016/j.neunet.2023.03.008

Li, X., and Li, F. (2017). "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice).

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*. doi: 10.48550/arXiv.1805.12152

McNeely-White, D., Beveridge, J. R., and Draper, B. A. (2020). Inception and resnet features are (almost) equivalent. *Cogn. Syst. Res.* 59, 312–318. doi: 10.1016/j.cogsys.2019.10.004

McNeely-White, D., Sattelberg, B., Blanchard, N., and Beveridge, R. (2021). Exploring the interchangeability of CNN embedding spaces. *arXiv preprint arXiv:2010.02323*. doi: 10.48550/arXiv.2010.02323

McNeely-White, D., Sattelberg, B., Blanchard, N., and Beveridge, R. (2022). Canonical face embeddings. *IEEE Trans. Biometr. Behav. Identity Sci.* 4, 197–209. doi: 10.1109/TBIOM.2022.3155372

Mustafa, A., Khan, S., Hayat, M., Goecke, R., Shen, J., and Shao, L. (2019). "Adversarial defense by restricting the hidden space of deep neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (Seoul), 3385–3394.

Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. (2022). Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*. doi: 10.48550/arXiv.2205.07460

Qiu, H., Zeng, Y., Zheng, Q., Guo, S., Zhang, T., and Li, H. (2021). An efficient preprocessing-based approach to mitigate advanced adversarial attacks. *IEEE Trans. Comput.* doi: 10.1109/TC.2021.3076826

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., et al. (2021). "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning* (Seoul: PMLR), 8748–8763.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2019). MobileNetV2: Inverted residuals and linear bottlenecks. *arXiv preprint arXiv:1801.04381*. doi: 10.1109/CVPR.2018.00474

Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. doi: 10.48550/arXiv.1409.1556

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2014a). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*. doi: 10.48550/arXiv.1409.4842

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., et al. (2014b). *Intriguing Properties of Neural Networks*. Technical report.

Tan, M., and Le, Q. V. (2020). EfficientNet: rethinking model scaling for convolutional neural e networks. *arXiv preprint arXiv:1905.11946*. doi: 10.48550/arXiv.1905.11946

Xu, W., Evans, D., and Qi, Y. (2017). Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*. doi: 10.48550/arXiv.1704.01155

Zheng, H., Zhang, Z., Gu, J., Lee, H., and Prakash, A. (2020). Efficient adversarial training with transferable adversarial examples. *arXiv preprint arXiv:1912.11969*. doi: 10.48550/arXiv.1912.11969

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 02 November 2023
DOI 10.3389/fcomp.2023.1253682

OPEN ACCESS

EDITED BY
Yunye Gong,
SRI International, United States

REVIEWED BY
Ajay Divakaran,
SRI International, United States
Michael Yao,
Stony Brook University, United States, in collaboration with reviewer AD
Chao Chen,
Stony Brook University, United States
Ruyi Lian,
Stony Brook University, United States, in collaboration with reviewer CC

*CORRESPONDENCE
Peter Tu
✉ tu@ge.com

RECEIVED 05 July 2023
ACCEPTED 16 October 2023
PUBLISHED 02 November 2023

CITATION
Tu P, Yang Z, Hartley R, Xu Z, Zhang J, Fu Y, Campbell D, Singh J and Wang T (2023) Probabilistic and semantic descriptions of image manifolds and their applications. *Front. Comput. Sci.* 5:1253682. doi: 10.3389/fcomp.2023.1253682

Probabilistic and semantic descriptions of image manifolds and their applications

Peter Tu[1]*, Zhaoyuan Yang[1], Richard Hartley[2], Zhiwei Xu[2], Jing Zhang[2], Yiwei Fu[1], Dylan Campbell[2], Jaskirat Singh[2] and Tianyu Wang[2]

[1]Computer Vision and Machine Learning Laboratory, General Electric Research, Niskayuna, NY, United States, [2]School of Computing, College of Engineering, Computing and Cybernetics, Australian National University, Canberra, ACT, Australia

This paper begins with a description of methods for estimating probability density functions for images that reflects the observation that such data is usually constrained to lie in restricted regions of the high-dimensional image space—not every pattern of pixels is an image. It is common to say that images lie on a lower-dimensional manifold in the high-dimensional space. However, although images may lie on such lower-dimensional manifolds, it is not the case that all points on the manifold have an equal probability of being images. Images are unevenly distributed on the manifold, and our task is to devise ways to model this distribution as a probability distribution. In pursuing this goal, we consider generative models that are popular in AI and computer vision community. For our purposes, generative/probabilistic models should have the properties of (1) sample generation: it should be possible to sample from this distribution according to the modeled density function, and (2) probability computation: given a previously unseen sample from the dataset of interest, one should be able to compute the probability of the sample, at least up to a normalizing constant. To this end, we investigate the use of methods such as normalizing flow and diffusion models. We then show how semantic interpretations are used to describe points on the manifold. To achieve this, we consider an emergent language framework that makes use of variational encoders to produce a disentangled representation of points that reside on a given manifold. Trajectories between points on a manifold can then be described in terms of evolving semantic descriptions. In addition to describing the manifold in terms of density and semantic disentanglement, we also show that such probabilistic descriptions (bounded) can be used to improve semantic consistency by constructing defenses against adversarial attacks. We evaluate our methods on CelebA and point samples for likelihood estimation with improved semantic robustness and out-of-distribution detection capability, MNIST and CelebA for semantic disentanglement with explainable and editable semantic interpolation, and CelebA and Fashion-MNIST to defend against patch attacks with significantly improved classification accuracy. We also discuss the limitations of applying our likelihood estimation to 2D images in diffusion models.

KEYWORDS

image manifold, normalizing flow, diffusion model, likelihood estimation, semantic disentanglement, adversarial attacks and defenses

1. Introduction

Understanding the complex probability distribution of the data is essential for image authenticity and quality analysis, but is challenging due to its high dimensionality and intricate domain variations (Gomtsyan et al., 2019; Pope et al., 2021). Seen images usually have high probabilities on a low-dimensional manifold embedded in the higher-dimensional space of the image encoder.

Nevertheless, the phenomenon that image embeddings encoded using methods such as a pretrained CLIP encoder (Ramesh et al., 2020) lie within a narrow cone of the unit sphere instead of the entire sphere (Gao et al., 2019; Tyshchuk et al., 2023), which degrades the aforementioned pattern of probability distribution. Hence, on such a manifold, it is unlikely that every point can be decoded into a realistic image because of the unevenly distributed probabilities. Therefore, it is important to compute the probability in the latent space to indicate whether the corresponding image is in a high-density region of the space (Lobato et al., 2016; Chang et al., 2017; Hajri et al., 2017; Grover et al., 2018; Papamakarios et al., 2021; Coeurdoux et al., 2022; Klein et al., 2022). This helps to distinguish seen images from unseen images, or synthetic images from real images. Some works train a discriminator with positive (real) and negative (synthetic) examples in the manner of contrastive learning (Liu et al., 2022) or analyze their frequency differences (Wang et al., 2020). However, they do not address this problem using the probabilistic framework afforded by modern generative models.

In this work, we calculate the exact log-probability of an image by utilizing generative models that assign high probabilities to seen images and low probabilities to unseen images. The confidence of such probabilities is usually related to image fidelity, we hence also introduce efficient and effective (with improved semantic robustness) generation strategies using hierarchical structure and large sampling steps with the Runge-Kutta method (RK4) (Runge, 1895; Kutta, 1901) for stabilization. Specifically, we use normalizing flow (NF) (Rezende and Mohamed, 2016; Papamakarios et al., 2021) and diffusion models (DMs) (Ho et al., 2020; Song et al., 2021; Luo, 2022) as image generators. NF models learn an image embedding space that conforms to a predefined distribution, usually a Gaussian. In contrast, DMs diffuse images with Gaussian noise in each forward step and learn denoising gradients for the backward steps. A random sample from the Gaussian distribution can then be analytically represented on an image manifold and visualized through an image decoder (for NF models) or denoiser (for diffusion models). In prior works, NF for exact likelihood estimation (Rezende and Mohamed, 2016; Kobyzev et al., 2019; Zhang and Chen, 2021) and with hierarchical structure (Liang et al., 2021; Hu et al., 2023; Voleti et al., 2023) have been explored in model training. To the best of our knowledge, however, it has not been studied by investigating such likelihood distribution of seen and unseen images with a hierarchical structure (without losing the image quality) from the manifold perspective. This is also applied to the diffusion models noting the difficulty of combining such exact likelihood with the mean squared error loss in diffusion training.

Samples from these image generators can be thought of having several meaningful semantic attributes. It is often desirable that these attributes be orthogonal to each other in the sample latent space so as to achieve a controllable and interpretable representation. In this work, we disentangle semantics in the latent space by using a variational autoencoder (VAE) (Kingma and Welling, 2013) in the framework of emergent languages (EL) (Havrylov and Titov, 2017; Kubricht et al., 2020; Pang et al., 2020; Tucker et al., 2021; Mu et al., 2023). This allows the latent representation on the manifold to be more robust, interpretable, compositional, controllable, and transferable. Although some VAE variant models such as β-TCVAE (Chen et al., 2018), GuidedVAE (Ding et al., 2020), and DCVAE (Parmar et al., 2021) achieve qualified semantic disentanglement results, we mainly focus on understanding the effectiveness of the emergent language framework for VAE based disentanglement inspired by Xu et al. (2022) and emphasizing the feasibility of our GridVAE (with mixture of Gaussian priors) under such an EL framework to study semantic distributions on the image manifold. We also evaluate their semantic robustness on such a manifold against adversarial and patch attacks (Carlini and Wagner, 2016; Brown et al., 2017; Tramer et al., 2017; Madry et al., 2018; Chou et al., 2019; Liu et al., 2020; Xiang et al., 2021; Hwang et al., 2023) and defend against the same attacks using semantic consistency with a purification loss.

We organize this paper into three sections, each with their own experiments: log-likelihood estimation for a given image under normalizing flows and diffusion models (see Section 2), semantic disentanglement in emergent languages for a latent representation of object attributes, using a proposed GridVAE model (see Section 3), and adversarial attacks and defenses in image space to preserve semantics (see Section 4).

2. Likelihood estimation with image generators

We evaluate the log-probability of a given image using (1) a hierarchical normalizing flow model, (2) a diffusion model adapted to taking large sampling steps, and (3) a diffusion model that uses a higher-order solution to increase generation robustness.

2.1. Hierarchical normalizing flow models

Normalizing flow (NF) refers to a sequence of invertible functions that may be used to transform a high-dimensional image space into a low-dimensional embedding space corresponding to a probability distribution, usually Gaussian. Dimensionality reduction is achieved via an autoencoding framework. For the hierarchical model, the latent vector corresponding to the image $\mathbf{x}_i$ at each level i is computed as

$$\mathbf{z}_i = g_i(\mathbf{y}_i) = g_i \circ f_i(\mathbf{x}_i) \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) , \tag{1}$$

and the inversion of this process reconstructs the latent $\mathbf{z}'_i$ to $\mathbf{x}'_i$ as

$$\mathbf{x}'_i = f'_i \circ g'_i(\mathbf{z}'_i) , \tag{2}$$

where the decoder f'_i and flow inverse function g'_i are inversions of the encoder f_i and flow function g respectively, and $\mathbf{z}'_i$ can be $\mathbf{z}_i$ or randomly sampled from $\mathcal{N}(\mathbf{0}, \mathbf{1})$. We illustrate hierarchical autoencoders and flows for rich and high-level spatial information with conditioning variables in either image space or latent space. In Figure 1, we show a 4-level hierarchical normalizing flow model, where each set of functions (f_i, g_i, g'_i, f'_i) corresponds to one level and where g'_i and f'_i are conditioned on the higher-level reconstruction, that is

$$\mathbf{x}'_1 = f'_1 \circ g'_1(\mathbf{z}'_1 | f'_2 \circ g'_2(\mathbf{z}'_2 | f'_3 \circ g'_3(\mathbf{z}'_3 | f'_4 \circ g'_4(\mathbf{z}'_4)))) . \tag{3}$$

The model is learned in two phases: joint learning of all autoencoders $\{f_i, f'_i\}$ and then joint learning of all flows $\{g_i, g'_i\}$ with

the pretrained autoencoders, for all $i \in \{1, 2, 3, 4\}$. The loss function for autoencoder learning, denoted $\mathcal{L}_{\text{ae}}$, is the mean squared error (MSE) between the reconstructed data and the processed data, and for the learning of flows the objective is to minimize the negative log-probability of $\mathbf{y}_i$, denoted $\mathcal{L}_{\text{flow}}$, such that the represented distribution of the latent variable is modeled to be the standard Gaussian distribution, from which a random latent variable can be sampled for data generation. Given N pixels and C channels ($C = 3$ for an RGB image and $C = 1$ for a greyscale image), $\mathbf{x}_i$ at level i can be represented as $\mathbf{x}_i = \{\mathbf{x}_{ij}\}$ for all $j \in \{1, ..., N\}$, the autoencoder loss is then given by

$$\mathcal{L}_{\text{ae}}(\mathbf{x}'_i, \mathbf{x}_i) = \frac{1}{CN} \sum_{j=1}^{N} \|\mathbf{x}'_{ij} - \mathbf{x}_{ij}\|^2 , \tag{4}$$

and the flow loss for the latent at level i is the negative log-probability of $\mathbf{y}_i$, that is $\mathcal{L}_{\text{flow}}(\mathbf{y}_i) = -\log p_Y(\mathbf{y}_i)$, using the change of variables as

$$\begin{aligned} \log p_Y(\mathbf{y}_i) &= \log p_Z(\mathbf{z}_i) + \log \left| \det \nabla_Y g_i(\mathbf{y}_i) \right| \\ &= \log p_Z(\mathbf{z}_i) + \log \left| J_Y \left(g_i(\mathbf{y}_i) \right) \right| , \end{aligned} \tag{5}$$

where

$$\begin{aligned} \log p_Z(\mathbf{z}_i) &= -\frac{1}{d_i} \log \frac{1}{\left(\sqrt{2\pi}\right)^{d_i}} \exp\left(-\frac{1}{2} \|\mathbf{z}_i\|^2 \right) \\ &= \frac{1}{2} \log 2\pi + \frac{1}{2d_i} \|\mathbf{z}_i\|^2 , \end{aligned} \tag{6}$$

d_i is the dimension of the ith latent and $J_X(\cdot)$ computes the Jacobian matrix over the partial derivative X. Similarly, the log-probability of $\mathbf{x}_i$ at level i is

$$\begin{aligned} \log p_X(\mathbf{x}_i) &= \log p_Z(\mathbf{z}_i) + \log \left| \det \nabla_X \left(g_i \circ f_i(\mathbf{x}_i) \right) \right| \\ &= \log p_Z(\mathbf{z}_i) + \log \left| \det J_Y(g_i(\mathbf{y}_i)) \right| + \log \left| \det J_X(f_i(\mathbf{x}_i)) \right| . \end{aligned} \tag{7}$$

Then, the log-probability of an image at level i with hierarchical autoencoders and flows from multiple downsampling layers, $\mathbf{x}_{i+1} = d(\mathbf{x}_i)$ at level i, can be calculated with the chain rule as

$$\log p(\mathbf{x}_i) = \sum_{j=1}^{i} \log p_X(\mathbf{x}_j) + \log \left| \det J_X(d(\mathbf{x}_{j-1})) \right| \cdot \mathbf{1}\left[j > 1 \right] , \tag{8}$$

where $[\cdot]$ is a binary indicator.

2.2. Diffusion models

Differently from normalizing flow models that sample in a low-dimensional embedding space due to the otherwise large computational complexity, diffusion models diffuse every image pixel in the image space independently, enabling pixelwise sampling from the Gaussian distribution. We outline below a strategy and formulas to allow uneven or extended step diffusion in the backward diffusion process.

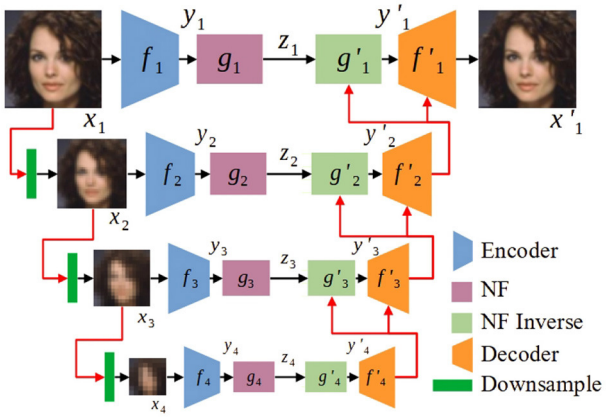


FIGURE 1
A 4-level hierarchical normalizing flow model, where each level involves the functions (f_i, g_i, g'_i, f'_i). The normalizing flow (NF) model is based on Glow (Kingma and Dhariwal, 2018); the downsampling block decreases image resolution by a factor of two; and the output of each higher ($i > 1$) level is conditioned on the output of the lower level. We first train all autoencoders $\{f_i, f'_i\}$ jointly, then train all flows $\{g_i, g'_i\}$ jointly, to obtain the generated image x'_1. The latent variable z_i conforms to the standard Gaussian distribution $\mathcal{N}(0, 1)$ during training; at test time, z_i is sampled from $\mathcal{N}(0, 1)$ for image generation.

2.2.1. Multi-step diffusion sampling

2.2.1.1. Forward process

The standard description of denoising diffusion model (Ho et al., 2020) defines a sequence of random variables $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$ according to a forward diffusion process

$$\mathbf{x}_{t+1} = \sqrt{\alpha_t}\, \mathbf{x}_t + \sqrt{\beta_t}\, \epsilon , \tag{9}$$

where $\beta_t = 1 - \alpha_t$, $\mathbf{x}_t$ is a sample from a random variable X_t, and ϵ is a sample from the standard (multidimensional) Gaussian. The index t takes integer values between 0 and T, and the set of random variables form a Markov chain.

The idea can be extended to define a continuous family of random variables according to the rule

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{\bar{\beta}_t}\, \epsilon , \tag{10}$$

where $\bar{\beta}_t = 1 - \bar{\alpha}_t$, and for simplicity, we can assume that x_t is defined for t taking continuous values in the interval $[0, 1]$. Here, the values $\bar{\alpha}_t$ are a decreasing function of t with $\bar{\alpha}_0 = 1$ and $\bar{\alpha}_1 = 0$. It is convenient to refer to t as *time*.

It is easily seen that if $\{0 = t_0, t_1, ..., t_T = 1\}$ are an increasing set of time instants between 0 and 1, then the sequence of random variables $\{X_{t_0}, \ldots X_{t_T}\}$ form a Markov chain. Indeed, it can be computed that for $0 \le s < t \le 1$, the conditional probabilities $p(\mathbf{x}_t|\mathbf{x}_s)$ are Gaussian

$$p(\mathbf{x}_t|\mathbf{x}_s) = \mathcal{N}\big(\mathbf{x}_t \mid \sqrt{\bar{\alpha}_{st}}\mathbf{x}_s,\ \bar{\beta}_{st}\big) . \tag{11}$$

where $\bar{\alpha}_{st} = \bar{\alpha}_t / \bar{\alpha}_s$ and $\bar{\beta}_{st} = 1 - \bar{\alpha}_{st}$. This is the isotropic normal distribution having mean $\sqrt{\bar{\alpha}_{st}}\mathbf{x}_s$ and variance $\bar{\beta}_{st}$. Similarly to Eq. (9), one has

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_{st}}\, \mathbf{x}_s + \sqrt{\bar{\beta}_{st}}\, \epsilon . \tag{12}$$

This applies in particular when s and t refer to consecutive time instants t_i and t_{i+1}. In this case, the joint probability of $\{X_{t_0}, \ldots, X_{t_T}\}$ is given by

$$p(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_T}) = p(\mathbf{x}_{t_0}) \prod_{i=1}^{T} p(\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}) \,. \tag{13}$$

One also observes, from Eq. (10) that $p(\mathbf{x}_1)$ is a standard Gaussian distribution. A special case is where the time steps are chosen evenly spaced between 0 and 1. Thus, if $h = 1/T$, this can be written as

$$p(\mathbf{x}_0, \mathbf{x}_h, \mathbf{x}_{2h}, \ldots, \mathbf{x}_{Th}) = p(\mathbf{x}_0) \prod_{i=1}^{T} p(\mathbf{x}_{ih}|\mathbf{x}_{(i-1)h}) \,. \tag{14}$$

2.2.1.2. Backward process

The joint probability distribution is also a Markov chain, which can be written in the reverse order, as

$$p(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_T}) = p(\mathbf{x}_{t_T}) \prod_{i=1}^{T} p(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i}) \,. \tag{15}$$

This allows us to generate samples from X_0 by choosing a sample from $X_1 = X_{t_T}$ (a standard Gaussian distribution) and then successively sampling from the conditional probability distributions $p(\mathbf{x}_{t_{i-1}}|\mathbf{x}_{t_i})$.

Unfortunately, although the forward conditional distributions $p(\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ are known Gaussian distributions, the backward distributions are not known and are not Gaussian. In general, for $s < t$, the conditional distribution $p(\mathbf{x}_t|\mathbf{x}_s)$ is Gaussian, but the inverse $p(\mathbf{x}_s|\mathbf{x}_t)$ is not.

However, if $(t - s)$ is small, or more exactly, if the variance of the added noise, $\bar{\beta}_{st} = 1 - \bar{\alpha}_{st}$ is small, then the distributions can be accurately approximated by Gaussians with the same variance $\bar{\beta}_{st}$ as the forward conditionals. With this assumption, the form of the backward conditional $p(\mathbf{x}_s|\mathbf{x}_t)$ is specified just by determining its mean, denoted by $\mu(\mathbf{x}_s|\mathbf{x}_t)$. The training process of the diffusion model consists of learning (using a neural network) the function $\mu(\mathbf{x}_s|\mathbf{x}_t)$ as a function of $\mathbf{x}_t$. As explained in Ho et al. (2020), it is not necessary to learn this function for all pairs (s, t), as will be elaborated below.

We follow and generalize the formulation in Ho et al. (2020). The training process learns a function $\epsilon_\theta(\mathbf{x}_t, t)$ that minimizes the expected least-squared loss function

$$E_{\mathbf{x}_0 \sim X_0, \epsilon \sim \mathcal{N}}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2] \,, \tag{16}$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{\bar{\beta}_t}\epsilon$. As such it estimates (exactly, if the optimum function ϵ_θ is found) the expected value of the added noise, given $\mathbf{x}_t$ (note that it estimates the *expected value* of the added noise, and not the actual noise, which cannot be predicted). In this case, following Ho et al. (2020),

$$\mu(\mathbf{x}_{t-1}|\mathbf{x}_t) = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \,. \tag{17}$$

In this form, this formula is easily generalized to

$$\begin{aligned} \mu(\mathbf{x}_s|\mathbf{x}_t) &= \frac{1}{\sqrt{\bar{\alpha}_{st}}} \left(\mathbf{x}_t - \frac{1 - \bar{\alpha}_{st}}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \\ &= \frac{1}{\sqrt{\bar{\alpha}_{st}}} \left(\mathbf{x}_t - \frac{\bar{\beta}_{st}}{\sqrt{\bar{\beta}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \,. \end{aligned} \tag{18}$$

As for the variance of $p(\mathbf{x}_s|\mathbf{x}_t)$, in Ho et al. (2020) it is assumed that the $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is an isotropic Gaussian (although in reality, it is not exactly a Gaussian, nor exactly isotropic). The covariance matrix of this Gaussian is denoted by $\sigma_{st}^2\mathbf{I}$, and two possible choices are given, which are generalized naturally to

$$\sigma_{st}^2 = \bar{\beta}_{st} \quad \text{or} \quad \sigma_{st}^2 = \frac{\bar{\beta}_s \bar{\beta}_{st}}{\bar{\beta}_t} \,. \tag{19}$$

As pointed out in Ho et al. (2020) both of these are compromises. The first choice expresses the approximation that the variance of the noise added in the backward process is equal to the variance in the backward process. As mentioned, this is true for small time steps.

Thus, in our work, we choose to model the reverse conditional as follows,

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_s|\mu(\mathbf{x}_s|\mathbf{x}_t), \sigma_{st}^2\mathbf{I}) \,, \tag{20}$$

where $\mu(\mathbf{x}_s|\mathbf{x}_t)$ is given by Eq. (18) and σ_{st}^2 is given by Eq. (19). This is an approximation of the true conditional probability $p(\mathbf{x}_s|\mathbf{x}_t)$.

2.2.2. Probability estimation

In the following, we choose a finite set of T time instances (usually equally spaced) $\{0 = \tau_0, \tau_1, \ldots, \tau_T = 1\}$ and consider the Markov chain consisting of the variables X_{τ_t}, for $t = \{0, \ldots, T\}$, at these time instances. For simplicity, we use the notation X_t instead of X_{τ_t} and $\mathbf{x}_t$ a sample from the corresponding random variable. Then, the notation corresponds to the common notation in the literature, but also applies in the case of unevenly, or widely sampled time instants.

To distinguish between the true probabilities of the variables X_t and the modeled conditional probabilities, the true probabilities will be denoted by q (instead of p which was used previously). The modeled probabilities will be denoted by $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, and the probability distribution of X_T, which is Gaussian, will be denoted by $p(\mathbf{x}_T)$.

The image probability can be calculated by using the forward and backward processes for each step of a pretrained diffusion model. The joint probability $p(\mathbf{x}_{0:T})$ and the probability of clean input $\mathbf{x}_0$ can be computed using the forward and backward conditional probability, $q(\mathbf{x}_{t+1}|\mathbf{x}_t)$ and $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$ respectively. Each sampling pair $(\mathbf{x}_t, \mathbf{x}_{t+1})$ where $t \in S = \{0, 1, 2, ..., T - 1\}$, follows the Markov chain rule resulting in the joint probability

$$p(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t \in S} q(\mathbf{x}_{t+1}|\mathbf{x}_t) = p(\mathbf{x}_T) \prod_{t \in S} p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}) \,, \tag{21}$$

so

$$q(\mathbf{x}_0) = \frac{p(\mathbf{x}_T) \prod_{t \in S} p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})}{\prod_{t \in S} q(\mathbf{x}_{t+1}|\mathbf{x}_t)} \,. \tag{22}$$

The negative log-probability of the input image $\mathbf{x}_0$ is then

$$-\log q(\mathbf{x}_0) = -\log p(\mathbf{x}_T) + \sum_{t\in S}\left(\underbrace{\log q(\mathbf{x}_{t+1}|\mathbf{x}_t)}_{\text{forward process}} \underbrace{-\log p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})}_{\text{backward process}}\right). \tag{23}$$

Computing Eq. (23) can be decomposed into three steps:

(1) Calculating $\log p(\mathbf{x}_T)$. Since $\mathbf{x}_0$ is fully diffused after T forward steps, $\mathbf{x}_T$ follows the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$, and thus the negative log-likelihood only depends on the Gaussian noise.

(2) Calculating $\log q(\mathbf{x}_{t+1}|\mathbf{x}_t)$. Since $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is a Gaussian with known mean $\bar{\alpha}_t/\bar{\alpha}_{t-1}$, and variance $1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}$, the conditional probability is easily computed, as a Gaussian probability.

(3) Calculating $\log p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$. Similarly, the probability $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is modeled as a Gaussian, with mean and variance given by Eq. (18) and Eq. (19) (where $s = t-1$) the backward conditional probabilities are easily computed.

2.2.3. Higher-order solution

With the hypothesis that high-fidelity image generation is capable of maintaining image semantics, in each of the diffusion inversion steps the $\mathbf{x}_0$ estimation and log-likelihood calculation should be stable and reliable with a small distribution variance. The diffusion inversion, however, usually requires a sufficiently small sampling step h, where DDPM (Ho et al., 2020) only supports $h = 1$ and DDIM is vulnerable to h (Song et al., 2021) as evidenced in Figure 8. It is important to alleviate the effect of h on the generation step by stabilizing the backward process in diffusion models.

Without loss of generality, the Runge-Kutta method (RK4) (Runge, 1895; Kutta, 1901) can achieve a stable inversion process by constructing a higher-order function to solve an initial value problem. Different from the traditional RK4, the diffusion inversion requires inverse-temporal updates because of the denoising gradient direction from the initial noisy image at $t = T$ to the clean image at $t = 0$. We provide the formulation of traditional RK4 and our inverse-temporal version in the Supplementary material.

2.3. Experiments

For each of the hierarchical normalizing flows (NFs) and diffusion models (DMs), we first show the effectiveness of likelihood estimation to analyze the image distribution (on 2D images for NFs and point samples for DMs). For likelihood estimation with image fidelity, we then illustrate the quality of images generated by our generation models (sampling on the manifold from a Gaussian distribution as well as resolution enhancement in NFs and sampling step exploration with RK4 stabilization in DMs).

2.3.1. Experiments on hierarchical normalizing flow models

2.3.1.1. Probability estimation

Figure 2 illustrates the probability density estimation on level 3 for an in-distribution dataset CelebA (Liu et al., 2015) and an out-of-distribution dataset CIFAR10 (Krizhevsky, 2009). The distribution of the latent variable $\mathbf{z}_i$ of CelebA is concentrated on a higher mean value than that of CIFAR10 due to the learning of $\mathbf{z}_i$ in the standard Gaussian distribution. Similarly, this distribution tendency is not changed in the image space illustrated by $\log p(\mathbf{x}_i)$. In this case, outlier samples from the in-distribution dataset can be detected with a small probability in the probability estimation.

2.3.1.2. Random image generation

Image reconstructions with encoded latent variables and conditional images as well as random samples are provided in Figure 3. For the low-level autoencoder and flow, say at level 1, conditioned on the sequence of decoded $\mathbf{x}_i$ for $i = \{2, 3, 4\}$, the reconstruction of $\mathbf{x}_1$ is close to the processed images although some human facial details are lost due to the downsampling mechanism, see Figure 3A. While randomly sampling $\{\mathbf{z}_i\}$ from the normal distribution at each level, the generated human faces are smooth but with blurry details in such as hair and chin and lack a realistic background.

2.3.1.3. Image super-resolution

With the jointly trained autoencoders and flows on CelebA, the images with low resolution, $3 \times 8 \times 8$ (channel $\times$ height $\times$ width) and $3 \times 16 \times 16$, are decoded to $3 \times 64 \times 64$ with smooth human faces, see Figures 4A, B respectively. The low-resolution image $\mathbf{x}_i$ is used as a condition image for (1) NF inverse $\{g_i'\}$ to generate embedding code to combine with the randomly sampled $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and (2) decoders $\{f_i'\}$ to concatenate with all upsampling layers in each decoder. This preserves the human facial details from either high levels or low levels for realistic image generation. As the resolution of the low-resolution images increases, the embedding code contains richer details.

2.3.2. Experiments on diffusion models

2.3.2.1. Log-likelihood estimation on point samples

We evaluate the log-probability of each point of point samples (Pedregosa et al., 2011) including Swiss roll, circle, moon, and S shown in Figure 5. Given a pretrained diffusion model on Swiss roll samples with 100 forward steps with each diffused by random Gaussian noise (see Figure 5A, the log-probability of the samples in Figure 5B follows Eq. (23) with $h = 1$ and indicates higher probability and density on seen or similar samples than unseen ones. In Figures 5B, C, the mean value of the Swiss roll sample achieves a higher mean value, -0.933, and a higher histogram density, 0.7, than the others. As the difference in the sample shape from the Swiss roll increases, the log-likelihood decreases, as shown in the bar chart in Figure 5C. It indicates that sampling from a low-density distribution is unable to reverse the diffusion step to obtain a realistic sample from the training set.

2.3.2.2. DDPM sampling with large steps

While Figure 5 uses $h = 1$ as the standard DDPM sampling process, it is feasible to sample with a fairly large step without losing the sample quality. This enables sampling from the Gaussian distribution for the log-likelihood estimation with less running

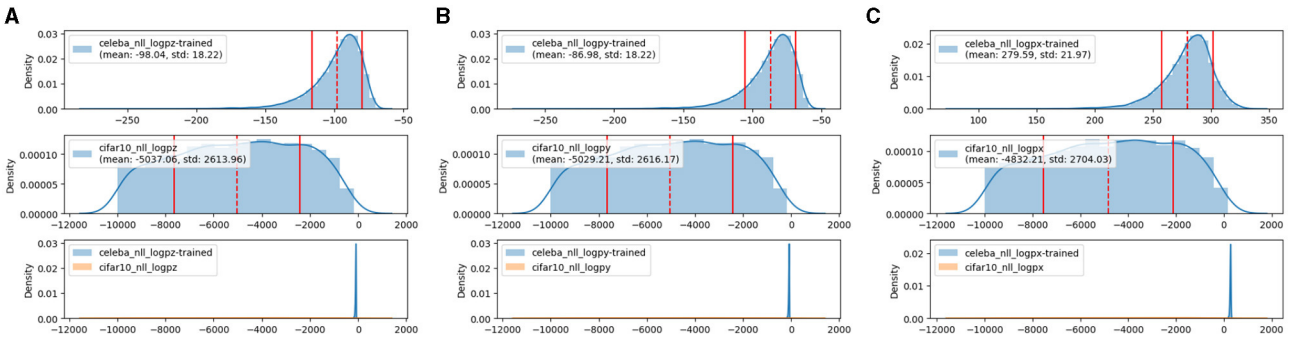


FIGURE 2
Log-likelihood estimation using hierarchical autoencoders and flows. The encoder and flow are trained on CelebA and evaluated on CelebA and CIFAR10. The x-axis is $\log p(\cdot)$ and the y-axis is the histogram density. In each subfigure, the first row is on the in-distribution dataset CelebA and the second row is on out-of-distribution CIFAR10, both are in the last row. In **(A)**, $\log p(z)$ can detect outlier samples, and adding $\log|\det(\cdot)|$ from NF and autoencoder does not significantly affect the distribution tendency, see **(B)** and **(C)**. For better visualization, samples with $\log p(\cdot)$ less than −10,000 are filtered out.

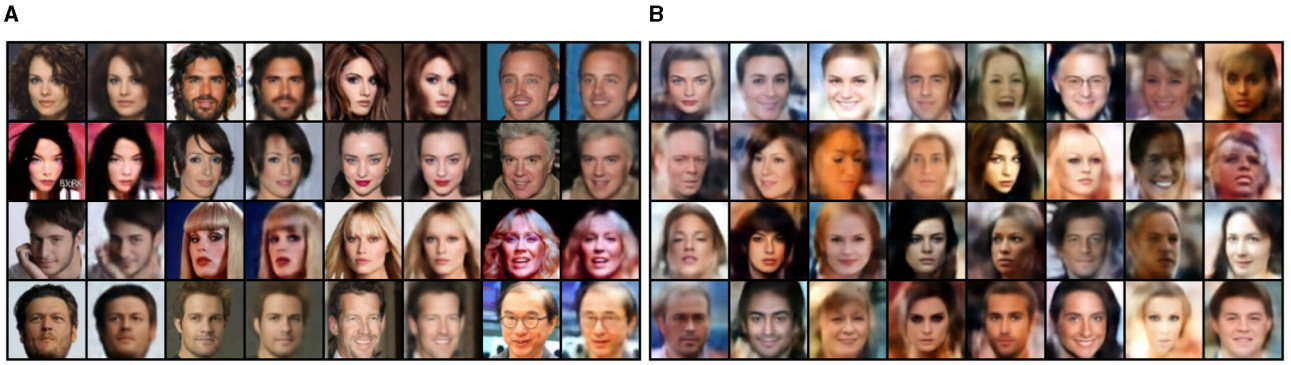


FIGURE 3
Image reconstruction and generation on the end-to-end training of 4-level autoencoders and flows. For each of two columns from left to right in **(A)**, the left is the real image and the right is the reconstructed image. **(A)** Reconstruction at level 1 with $\{z_i\}$ from encoders $\{g_i\}$ and conditioned on $\{f_i'\}$. **(B)** Random generation at level 1 with latent variables $\{z_i\} \sim \mathcal{N}(0, 1)$ and conditioned on $\{f_i'\}$.

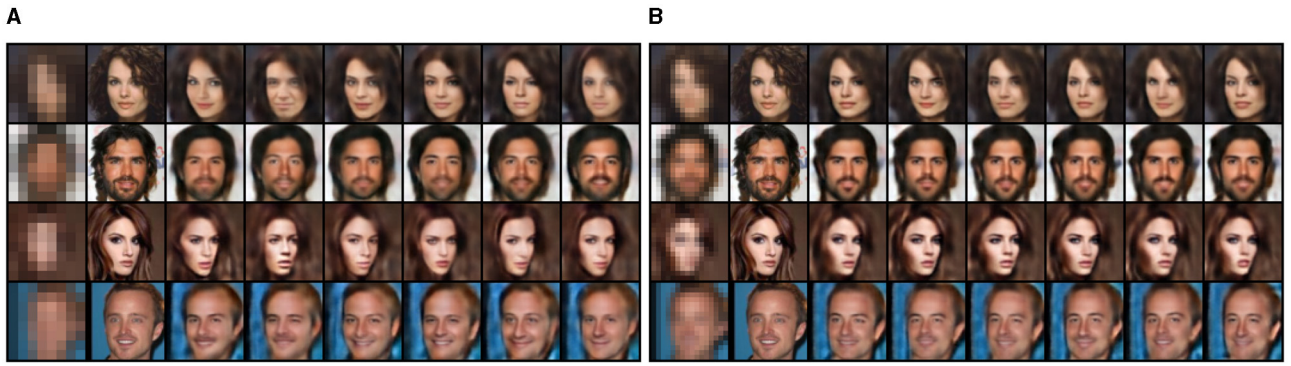


FIGURE 4
Image super-resolution on dataset CelebA. The first column is low-resolution images, the second column is real images, and the rest are high-resolution images with latent variables $\{Z_i\} \sim \mathcal{N}(0, 1)$ conditioned on the low-resolution images and temperature 1.0. **(A)** Resolution: $3 \times 8 \times 8$ to $3 \times 64 \times 64$. **(B)** Resolution: $3 \times 16 \times 16$ to $3 \times 64 \times 64$.

time. To visualize the image quality, we evaluate the samples on CelebA dataset by using a pretrained diffusion model with 1,000 forward diffusion steps. In Figure 6, the sampling has an increase step h in {2, 10, 100} while the samples have a high quality for $h = \{2, 10\}$ and a fair quality for $h = 100$.

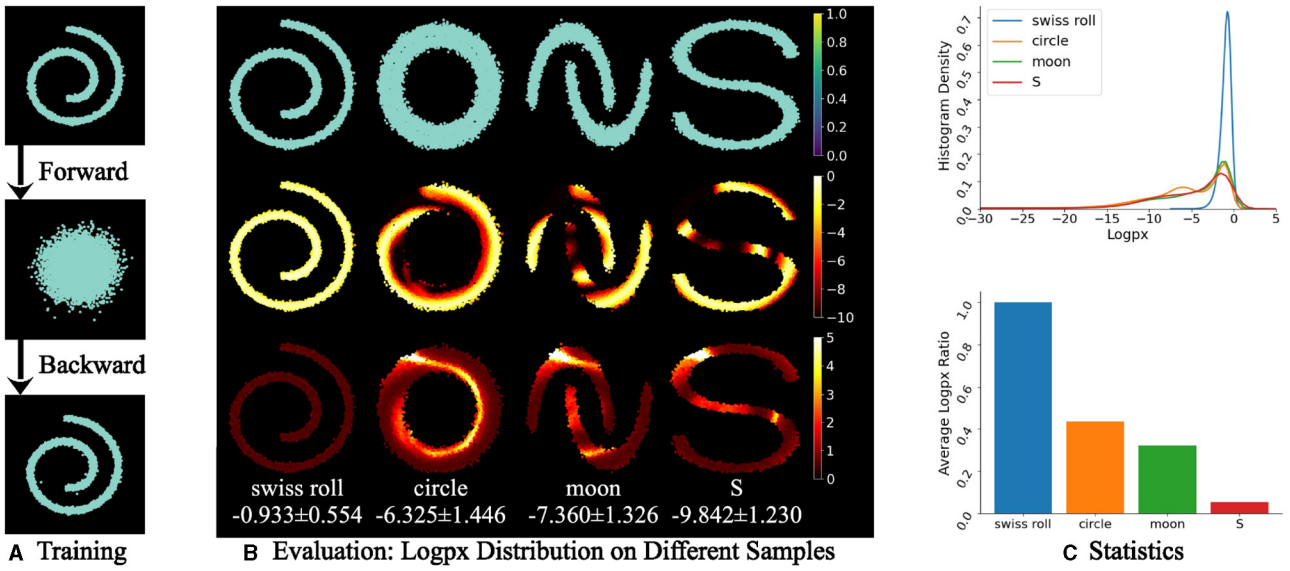


FIGURE 5
Evaluation of log-probability of x_0 on point samples with each of 10,000 points. **(A)** The training is on a Swiss roll sample and a diffusion model with forward (noising) and backward (denoising) processes. **(B)** At the evaluation phase, unseen samples, that is circle, moon, and S, have lower $\log p(x)$ values than the seen Swiss roll sample. In **(B)**, the first row is sampled points and the middle and last rows are the mean value and the standard deviation of $\log p(x)$ for each point on 100 random rounds respectively, which is represented as "mean ± SD." The randomness lies in the random noise in the forward and backward processes. A lighter color indicates a higher density. **(C)** Statistics indicates the higher density of a seen sample (Swiss roll) than an unseen one (circle, moon, or S) through the diffusion model by using the negative of Eq. (23) with $\log_{10}$.

2.3.2.3. Higher-order solution stabilizes sampling

While sampling with a large step h can sometimes cause bias from the one with a small h, RK4 effectively alleviates such a bias. We evaluate both the point samples and human face images from CelebA. In Figure 7, compared with the sample by using DDPM, RK4 with DDPM inference achieves less noise at $h = \{2, 5, 10\}$. For $h = 20$, RK4 performs expectedly worse because it only applies five sampling steps while the training is on ($T = 100$) diffusion steps. In Figure 8, we apply DDIM as the inference method for RK4 to deterministically compare the samples with DDIM. As h increases from 1 to 100, many of the samples using DDIM lose the image consistency with the samples at $h = 1$; however, most of the samples using RK4 still retain the image consistency. This indicates the robustness of applying RK4 with a large sampling step.

3. Semantic disentanglement on manifold

Semantics of object attributes are crucial for image distribution and spatial presentation. For instance, different shapes in Figure 5 represent different objects while those closer to the seen samples have high likelihood; in Figure 8 semantics such as human gender (see the 2nd row and 3rd column image with DDIM and RK4) are fundamental for controllable generation by sampling in high-density regions of specific semantic clusters on the manifold. These semantics, however, are usually entangled without independent distributions from each other for deterministic embedding sampling on the image manifold (Liu et al., 2018; Ling et al., 2022; Pastrana, 2022). Hence, regardless of image generation models, we exploit the popular and efficient variational autoencoder and introduce our GridVAE model for effective semantic disentanglement on the image manifold.

3.1. GridVAE for clustering and disentanglement

3.1.1. Formulation

A variational autoencoder (VAE) (Kingma and Welling, 2013) is a neural network that maps inputs to a distribution instead of a fixed vector. Given an input $\boldsymbol{x}$, the encoder with neural network parameters ϕ maps it to a hidden representation $\boldsymbol{z}$. The decoder with the latent representation $\boldsymbol{z}$ as its input and the neural network parameters as θ reconstructs the output to be as similar to the input $\boldsymbol{x}$. We denote the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. The hidden representation follows a prior distribution $p(\boldsymbol{z})$.

With the goal of making the posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ close to the actual distribution $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, we minimize the Kullback-Leibler divergence between these two distributions. Specifically, we aim to maximize the log-likelihood of generating real data while minimizing the difference between the real and estimated posterior distribution by using the evidence lower bound (ELBO) as the VAE loss function

$$\begin{aligned} L(\theta, \phi) &= -\log p_\theta(\boldsymbol{x}) + D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) || p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\ &= -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p_\theta(\boldsymbol{z})) \,, \end{aligned} \tag{24}$$

where the first term is the reconstruction loss and the second term is the regularization for $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be close to $p_\theta(\boldsymbol{z})$. The prior distribution of $\boldsymbol{z}$ is often chosen to be a standard unit isotropic

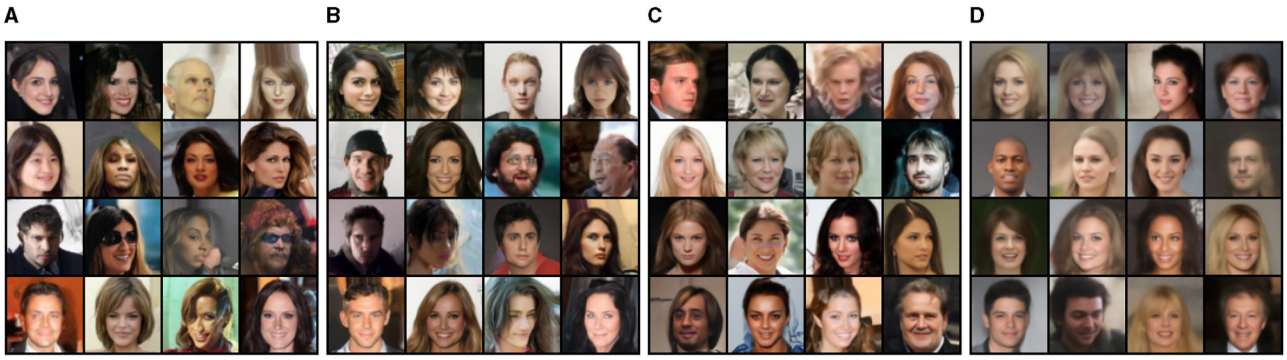


FIGURE 6
Image generation from our modified DDPM with step size h. Samples follow a Gaussian distribution. Fine details are obtained even for very large steps ($h = 100$). **(A)** $h = 1$. **(B)** $h = 2$. **(C)** $h = 10$. **(D)** $h = 100$.

FIGURE 7
Sampling robustness of DDPM and RK4 @ step h. With h being 5 or 10, RK4 still achieves clear sampling compared with DDPM. If h is too large, for instance 20, RK4 fails as expected. **(A)** DDPM. **(B)** RK4@2. **(C)** RK4@5. **(D)** RK4@10. **(E)** RK4@20.

FIGURE 8
Random image generation using DDIM and RK4 with DDIM as inference @ time step h={1, 2, 10, 100}. The RK4 sampling method is more robust than DDIM, especially at $h = 100$, with a higher image consistency than those at $h = 1$. **(A)** DDIM@1. **(B)** DDIM@2. **(C)** DDIM@10. **(D)** DDIM@100. **(E)** RK4@1. **(F)** RK4@2. **(G)** RK4@10. **(H)** RK4@100.

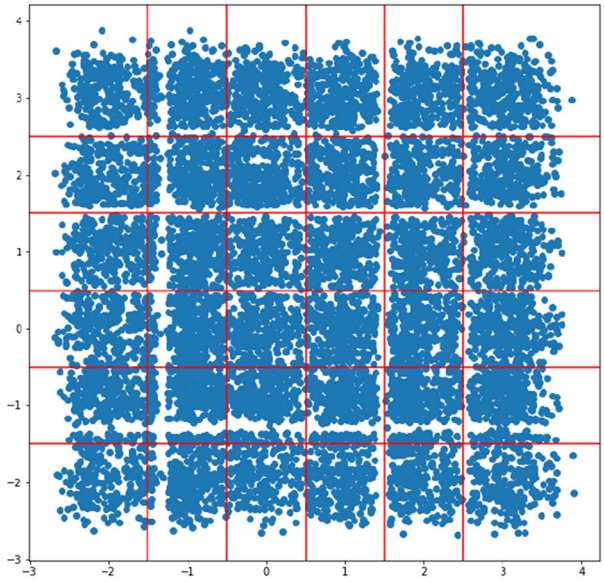


FIGURE 9
Scatter plot of test set latent space with an MoG prior.

Gaussian, which implies that the components of $\boldsymbol{z}$ should be uncorrelated and hence disentangled. If each variable in the latent space is only representative of a single element, we assume that this representation is disentangled and can be well interpreted.

Emergent language (EL) (Havrylov and Titov, 2017) is hereby introduced as a language that arises spontaneously in a multi-agent system without any pre-defined vocabulary or grammar. EL has been studied in the context of artificial intelligence and cognitive science to understand how language can emerge from interactions between agents. EL has the potential to be compositional such that it allows for referring to novel composite concepts by combining individual representations for their components according to systematic rules. However, for EL to be compositional, the latent space needs to be disentangled (Chaabouni et al., 2020). Hence, we integrate VAE into the EL framework by replacing the sender LSTM with the encoder of the VAE noting that the default LSTM encoder will entangle the symbols due to its sequential structure where the previous output is given as the input to the next symbol. In contrast, the symbols can be disentangled with a VAE encoder.

To achieve disentangled representations in EL, the VAE encoder must be able to cluster similar concepts into discrete symbols that are capable of representing attributes or concepts. The standard VAEs are powerful, but their prior distribution, which is typically the standard Gaussian, is inferior in clustering tasks, particularly the location and the number of cluster centers. In the EL setting, we desire a posterior distribution with multiple clusters, which naturally leads to an MoG prior distribution with K components

$$p(\boldsymbol{z}) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\boldsymbol{z}|\mu_k, \sigma_k^2)\,. \tag{25}$$

We choose the μ_k to be located on a grid in a Cartesian coordinate system so that the posterior distribution clusters can be easily determined based on the sample's distance to a cluster center. We refer to this new formulation as GridVAE, which is a VAE with a predefined MoG prior on a grid. The KL-divergence term in Eq. (24) can be re-written as

$$D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]\,. \tag{26}$$

The log probability of the prior can be easily calculated with the MoG distribution, and we only need to estimate the log probability of the posterior using a large batch size during training. By using a GridVAE, we can obtain a posterior distribution with multiple clusters that correspond to the same discrete attribute, while allowing for variations within the same cluster to generate different variations of the attribute.

3.1.2. Experiments

We evaluate the clustering and disentanglement capabilities of the proposed GridVAE model using a two-digit MNIST dataset (LeCun et al., 1998) consisting of digits 0 to 5. Each digit is from the original MNIST dataset, resulting in a total of 36 classes [00, 01, 02,..., 55].

To extract features for the encoder, we use a 4-layer ResNet (He et al., 2016) and its mirror as the decoder. The VAE latent space is 2-dimensional (2D), and if the VAE learns a disentangled representation, each dimension of the 2D latent space should represent one of the digits. We use a 2D mixture of Gaussian (MoG) as the prior distribution, with six components in each dimension centered at integer grid points from $[-2, -1, 0, 1, 2, 3]$, that is the coordinates for the cluster centers are $[(-2, -2), (-2, -1),..., (3, 3)]$. The standard deviation of the mixture of Gaussian is $1/3$.

After training the model, we generate a scatter plot of the test set latent space, as shown in Figure 9. Since the prior is a mixture of Gaussian on the grid points, if the posterior matches the prior, we can simply draw a boundary in the middle of two grid points, illustrated by the red lines in Figure 9.

With the trained model, one can sample in the latent space for image generation. In Figures 10A, B, when we decode from the cluster centers (i, j): in **(A)** we keep $j = 0$ and change i from -2 to 3, while in **(B)** we keep $i = 0$ and change j from -2 to 3. The latent space is disentangled with respect to the two digits - the first dimension of the latent space controls the first digit, while the second dimension controls the second digit. Each of the cluster centers corresponds to a different number.

Figures 10C, D show images generated within the cluster centered at $(1, 1)$, that is the pairs of number "44". If we slightly modify one of the dimensions, it corresponds to different variations of the number "4" along this dimension, while keeping the other digit unchanged.

Overall, these results demonstrate the effectiveness of the proposed GridVAE model in clustering and disentangling the latent space on the two-digit MNIST dataset.

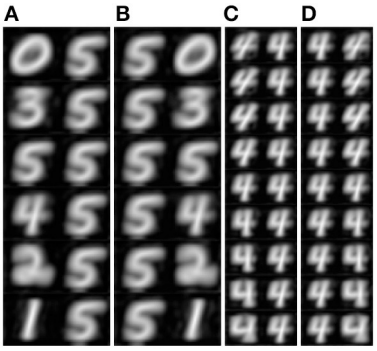


FIGURE 10
Generated images from sampling the latent space. **(A)** The second dimension is fixed at 0, changing the first dimension from −2 to +3. **(B)** The first dimension is fixed at 0 and the second dimension is changed from −2 to +3. **(C)** Around the cluster center(1, 1), keep the second dimension fixed and change the first dimension. **(D)** Around the cluster center(1, 1), keep the second dimension fixed and change the first dimension.

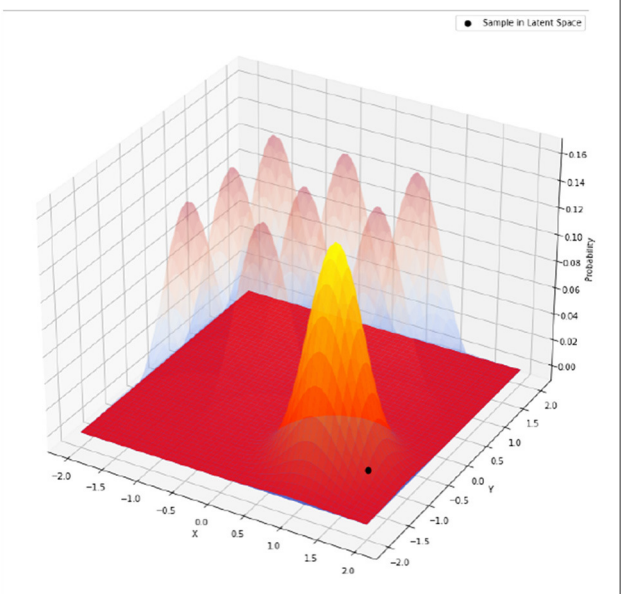


FIGURE 11
When calculating the KL-divergence, only the mixture component closest to the data (darker shade) is considered. Other components (lighter shade) are ignored. This can be generalized to multiple dimensions and multiple components in each dimension.

3.2. Scaling up GridVAE

In Section 3.1, the two-digit MNIST dataset lies in a 2-dimensional latent space. However, many real-world datasets would require a much higher dimensional space.

3.2.1. Addressing higher dimensional latent space

Discretizing a continuous space, such as in GridVAE, is challenging due to the curse of dimensionality (Bellman, 1957). This refers to the exponential growth in the number of clusters as the number of dimensions increases, which leads to a computational challenge when dealing with high-dimensional latent space. For example, when applying GridVAE to reconstruct images of the CelebA (Liu et al., 2015) dataset to learn the 40 attributes, we need a 40-dimensional latent space with two clusters in each dimension to represent the presence or absence of a given attribute. Firstly, parametrizing the mixture of Gaussian prior $p(\boldsymbol{z}) = \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{z}|\mu_k, \sigma_k^2)/K$ over 40 dimensions is prohibitively expensive as $K = 2^{40} \approx 1.1 \times 10^{12}$. Secondly, the assumption of equal probability for the components, which was appropriate for the simple 2-digit MNIST dataset, is no longer valid. This is because the attributes in the CelebA dataset are not uniformly distributed, and some combinations may not exist. For instance, the combination of "black hair" + "blonde hair" + "brown hair" + "bald" is impossible due to attribute conflicts. To address this issue, we use the proposed loss function in Eq. (24) incorporating relaxation.

To avoid pre-parametrizing $p(\boldsymbol{z})$ over 40 dimensions, we have implemented a dynamic calculation of the KL-divergence between q_ϕ and p_θ, whereby only the cluster that is closest to the latent space representation is considered, as illustrated in Figure 11. This means that clusters to which the data point does not belong do not affect its distribution, and the MoG distribution is simplified to a multivariate Gaussian as

$$D_{\mathrm{KL}}(p_1 \,||\, p_2) = \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - n + \mathrm{tr}\big(\Sigma_2^{-1}\Sigma_1\big) + (\mu_2-\mu_1)^{\intercal}\Sigma_2^{-1}(\mu_2-\mu_1)\right], \tag{27}$$

where $p_1 = q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mu_1, \Sigma_1)$, $\Sigma_1 = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$, $p_2 = \mathcal{N}(\mu_2, \Sigma_2)$, $\mu_2 = \mathsf{R}(\mu_1)$, and $\Sigma_2 = \mathrm{diag}(\sigma_0^2, \ldots, \sigma_n^0)$ with the round function $\mathsf{R}(\cdot)$ for the closest integer.

The key step here is that the round function dynamically selects the cluster center closest to μ_1, and σ_0 is a pre-defined variance for the prior distribution. It should be chosen so that two clusters next to each other have a reasonable degree of overlap, for example, $\sigma_0 = 1/16$ in some of our following experiments. The KL-divergence term becomes

$$\begin{aligned} D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) &= \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - n + \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2-\mu_1)^{\intercal}\Sigma_2^{-1}(\mu_2-\mu_1)\right] \\ &= \frac{1}{2}\left[\log\prod_i \sigma_0^2 - \log\prod_i \sigma_i^2 - n + \sum_i \frac{\sigma_i^2}{\sigma_0^2} + \sum_i \frac{(\mu_i - \mathsf{R}(\mu_i))^2}{\sigma_0^2}\right] \\ &= \frac{1}{2}\left[\sum_{i=1}^{n}\left(\log\sigma_0^2 - \log\sigma_i^2 - 1\right) + \sum_{i=1}^{n}\frac{\sigma_i^2 + (\mu_i - \mathsf{R}(\mu_i))^2}{\sigma_0^2}\right]. \end{aligned} \tag{28}$$

By adopting Eq. (28), we can significantly reduce the computational complexity of the model, even for a high-dimensional latent space,

bringing it to a level comparable to that of a standard VAE. It is worth noting that the global disentanglement may no longer be guaranteed. Rather, the model only provides local disentanglement within the proximity of each cluster.

Upon training the GridVAE with a 40-dimensional latent space by using the proposed Eq. (28) on the CelebA dataset, we observe some intriguing disentanglement phenomena. Figure 12 showcases the disentanglement of two latent space dimensions, where the first dimension governs one attribute and the second dimension determines another one. Combining these two dimensions leads to simultaneous attribute changes in the generated images.

An inherent limitation of this unsupervised approach is that while the latent space appears to be locally disentangled for each image, the same dimension may have different semantic interpretations across different images. To address this issue, we introduce all 40 attributes of the dataset during the training. This should establish an upper bound on the disentanglement.

3.2.2. From unsupervised to guided and partially guided GridVAE

To this end, we described an unsupervised approach to learning the latent space representation of images. However, for datasets like CelebA with ground truth attributes, we can incorporate them into the latent space to guide the learning. Specifically, we extract the 40-dimensional attribute vector indicating the presence or absence of each feature for each image in a batch and treat it as the ground truth cluster center μ_i^{gt}. Hence, instead of rounding the latent space representation μ_i in Eq. (28), we replace it with μ_i^{gt}.

One limitation of this approach is the requirement of the ground truth attributes for all images, which may not always be available or feasible. Additionally, it is important to note that while we refer to this approach as "guided," the given attribute information only serves in the latent space as the cluster assignment prior, and the VAE reconstruction task remains unsupervised. This differs from classical supervised learning, where the label information is the output. Furthermore, in our approach, no specific coordinate in the latent space is designated for the input. Instead, we provide guidance that the sample belongs to a cluster centered at a certain point in the latent space.

This guided learning framework can be extended to a subset of the 40 attributes or a latent space with more dimensions. For clarity, we will refer to the latter as "partially guided" to distinguish it from the commonly used "semi-supervised" by using a subset of the labeled dataset.

We conduct the experiments using attribute information as latent space priors and obtain the following findings for the guided approach: (a) GridVAE is able to cluster images accurately based on their attributes and the same dimension has the same semantic meaning across different images. For instance, dimension 31 represents "smile". (b) GridVAE could not generate images for clusters that have little or no representation in the training set. For example, the attempt to generate an image of a bald female by constraining GridVAE to the "female" and "bald" clusters is not achievable for an accurate representation. (c) Some attributes are more universal across different images, such as their ability to add a smile to almost any face. However, other attributes, such as gender, are not always modifiable. This could be caused by attributes that are not independent and can be influenced by others. Universal attributes, such as "smile," seem to primarily located locally in the image region without interruption from the other attributes, see Figure 13.

To further illustrate the incompleteness and correlation among the attributes in the CelebA dataset, we use a subset of the given attributes. We choose 38 out of the 40 attributes, excluding attributes 20 (female/male) and 31 (not smiling/smiling). Figure 14 shows that the GridVAE cannot learn the omitted attributes. This highlights the interdependence of different attributes in the latent space.

3.3. Combining manifolds of GridVAE disentangled attribute and facial recognition

After achieving a disentangled latent space, one may still wonder about the usefulness of a semantic description of a manifold. One can consider the scenario where another manifold, such as a facial recognition manifold, is learned. By studying these two manifolds jointly, we can gain insights to make the models more explainable and useful. One potential application is to better understand the relationship between facial attributes and facial recognition. By analyzing the disentangled latent space of facial attributes and the manifold learned for facial recognition, we can potentially identify which attributes are the most important for recognizing different faces. This understanding can then be used to improve the performance of facial recognition models as well as explain the model decisions.

For instance, FaceNet (Schroff et al., 2015) directly learns a mapping from face images to a compact Euclidean space where distances correspond to a measure of face similarity. To discover the semantic structure of this manifold with x as binary attributes, we can follow these steps:

1. Build a face recognition manifold using contrastive learning.
2. Use the CelebA dataset with ground truth attribute labels (40 binary values).
3. Insert CelebA samples onto the recognition manifold.
4. Find the nearest neighbor for each CelebA sample using the face recognition manifold coordinates.
5. For each attribute in $\mathbf{x}$, compute $p(\boldsymbol{x})$ over the entire CelebA dataset.
6. For each attribute in $\mathbf{x}$, compute $p(\boldsymbol{x}|\boldsymbol{x}$ of nearest neighbor $= 0)$.
7. For each attribute in $\boldsymbol{x}$, compute the KL divergence between $p(\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{x}$ of nearest neighbor $= 0)$.
8. Identify attributes with the largest KL divergence.

Figure 15 demonstrates that the KL Divergence between $p(\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{x}$ of nearest neighbor $= 0)$ is significantly larger for certain attributes, such as "male," "wearing lipstick," "young" and "no beard," than the others. This indicates that the neighborhood structure of the facial recognition manifold is markedly different from the distribution of these attributes in the entire dataset. These findings highlight the importance of the joint study of

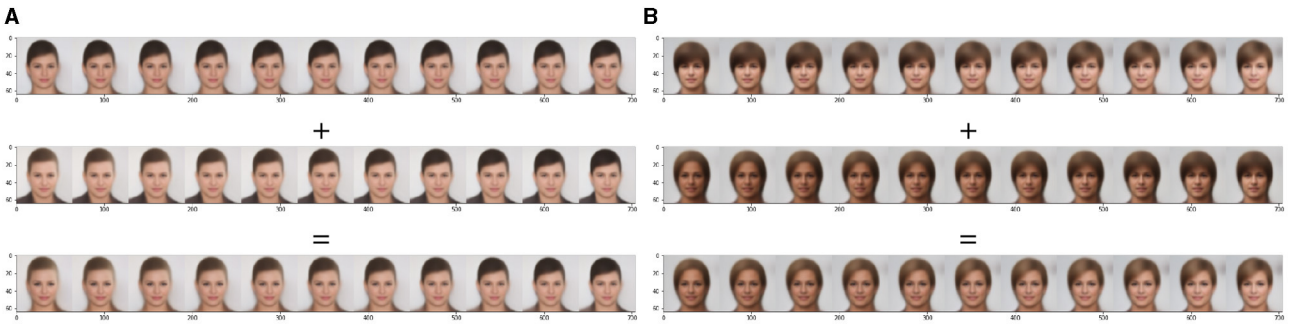


FIGURE 12
Two generated examples using linear sampling in the latent space. The top row fixes the dimensions and changes the first one, collar in **(A)** and skin color in **(B)**, from −0.5 to +1.5. The middle row fixes the dimensions and changes the second one, hair color in **(A)** and hairstyle in **(B)**, from −0.5 to +1.5. The bottom row changes the first and second dimensions from −0.5 to +1.5.

different manifolds to gain a more profound understanding of the relationship between the attributes and the recognition tasks. By incorporating it into the models, we can potentially improve the performance of facial recognition models and also enhance their interpretability.

4. Application to defend patch attacks

To this end, interpretable and controllable samplings from each semantic distribution on the manifold can be achieved by using the semantic disentanglement in Section 3 toward high-fidelity and diverse image generation and probability distribution analysis in Section 2. It is also of strong interest to enhance the robustness of such semantic samplings under certain attacks. In this section, we present an adversarial robustness framework by enforcing the semantic consistency between the classifier and the decoder for reliable density estimation on the manifold.

4.1. Adversarial defense with variational inference

In Yang et al. (2022), adversarial robustness can be achieved by enforcing the semantic consistency between a decoder and a classifier (adversarial robustness does not exist in non-semantically consistent classifier-decoder). We briefly review the adversarial purification framework below. We define the real-world high-dimensional data as $\mathbf{x} \in \mathbb{R}^n$ which lies on a low-dimensional manifold $\mathcal{M}$ diffeomorphic to $\mathbb{R}^m$ with $m \ll n$. We define an encoder function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a decoder function $f^\dagger : \mathbb{R}^m \to \mathbb{R}^n$ to form an autoencoder. For a point $\mathbf{x} \in \mathcal{M}$, $f^\dagger$ and f are approximate inverses. We define a discrete label set $\mathcal{L}$ of c elements as $\mathcal{L} = \{1, ..., c\}$ and a classifier in the latent space as $h : \mathbb{R}^m \to \mathcal{L}$. The encoder maps the image $\mathbf{x}$ to a lower-dimensional vector $\mathbf{z} = f(\mathbf{x}) \in \mathbb{R}^m$ and the functions f and h together form a classifier in the image space $h(\mathbf{z}) = (h \circ f)(\mathbf{x}) \in \mathcal{L}$.

A classifier (on the manifold) is a semantically consistent classifier if its predictions are consistent with the semantic interpretations of the images reconstructed by the decoder. Despite that the classifiers and decoders (on the manifold) have a low input dimension, it is still difficult to achieve high semantic consistency between them. Thus, we assume that predictions and reconstructions from high data density regions of $p(\mathbf{z}|\mathbf{x})$ are more likely to be semantically consistent and we need to estimate the probability density in the latent space with the variational inference.

We define three sets of parameters: (1) ϕ parametrizes the encoder distribution, denoted as $q_\phi(\mathbf{z}|\mathbf{x})$, (2) θ parametrizes the decoder distributions, represented as $p_\theta(\mathbf{x}|\mathbf{z})$, and (3) ψ parametrizes the classification head, given by $h_\psi(\mathbf{z})$. These parameters are jointly optimized with respect to the ELBO loss and the cross-entropy loss as shown in Eq. (29), where λ is the trade-off term between the ELBO and the classification. We provide the framework in Figures 16A, B for the two-stage procedure and the trajectory of cluster center change after introducing our purification over attacks in Figure 16C. By adopting this formulation, we notice a remarkable semantic consistency between the decoder and the classifier. Specifically, on Fashion-MNIST (Xiao et al., 2017), when making predictions on adversarial examples, if the predicted label is "bag," we observe that the reconstructed image tends to resemble a "bag" as well. This phenomenon is illustrated in Figures 16D, 17.

$$\max_{\theta,\phi,\psi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})]}_{\text{ELBO (lower bound of } \log p_\theta(\mathbf{x}))} + \lambda \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\mathbf{y}^\intercal \log h_\psi(\mathbf{z})]}_{\text{Classification loss}} . \tag{29}$$

To defend against image-level attacks, a purification vector can be obtained through the test-time optimization over the ELBO loss. For example, given an adversarial example $\mathbf{x}_{\text{adv}}$, a purified sample can be obtained by $\mathbf{x}_{\text{pfy}} = \mathbf{x}_{\text{adv}} + \boldsymbol{\epsilon}_{\text{pfy}}$ with

$$\boldsymbol{\epsilon}_{\text{pfy}} = \arg\max_{\boldsymbol{\epsilon}\in\mathcal{C}_{\text{pfy}}} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_{\text{adv}}+\boldsymbol{\epsilon})}\left[\log p_\theta(\mathbf{x}_{\text{adv}} + \boldsymbol{\epsilon}|\mathbf{z})\right] - D_{\text{KL}}[q_\phi(\mathbf{z}|\mathbf{x}_{\text{adv}} + \boldsymbol{\epsilon})\|p(\mathbf{z})] , \tag{30}$$

where $\mathcal{C}_{\text{pfy}} = \{\boldsymbol{\epsilon} \in \mathbb{R}^n \mid \mathbf{x}_{\text{adv}} + \boldsymbol{\epsilon} \in [0, 1]^n \text{ and } \|\boldsymbol{\epsilon}\|_p \leq \epsilon_{\text{th}}\}$ which is the feasible set for purification and ϵ_{th} is the purification budget. Since the classifier and the decoder are semantically consistent, the predictions from the classifier become normal to defend against the attacks upon normal reconstructions.

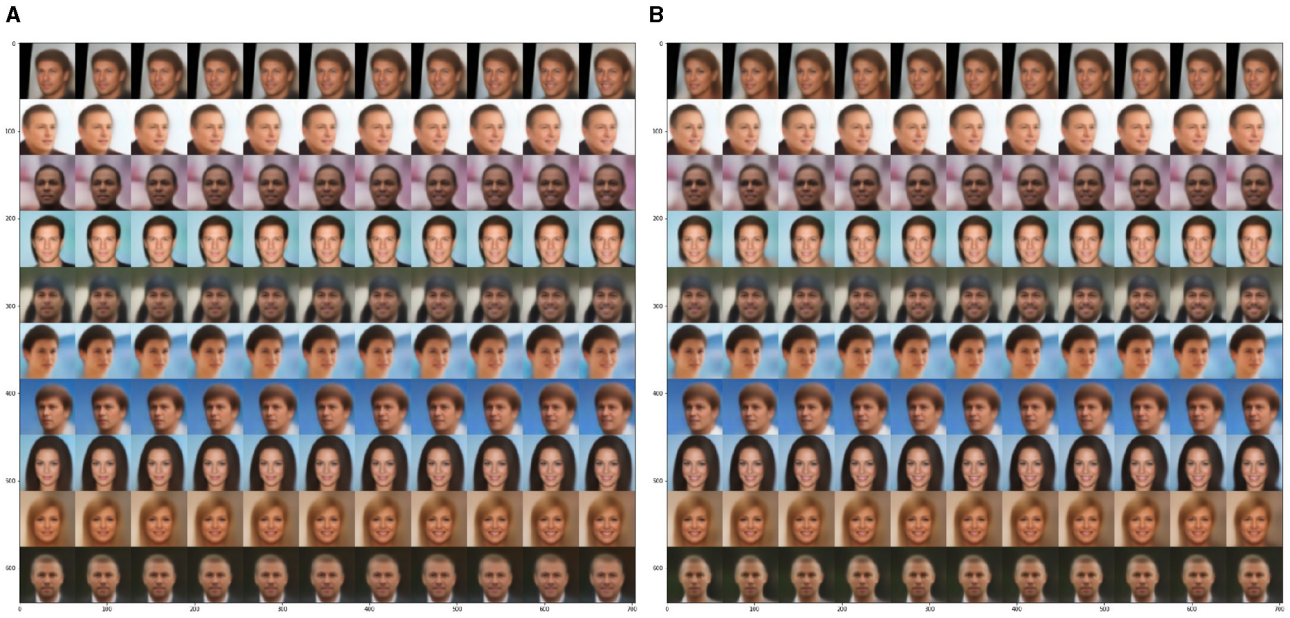


FIGURE 13
Generated images from sampling in the latent space. Keeping all other dimensions fixed and changing dimension **(A)** 31 (smile) from −0.5 to +1.5, or **(B)** 20 (male) from −0.5 to +1.5.

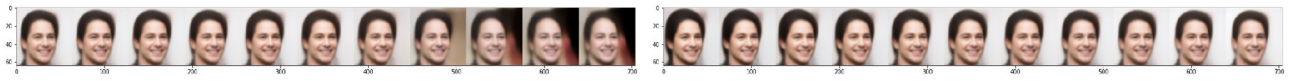


FIGURE 14
Partially guided GridVAE generation from the latent attributes which are not provided during training. The **left** and **right** subfigures (each with 11 images) are with the dimensions 20 and 31 respectively.

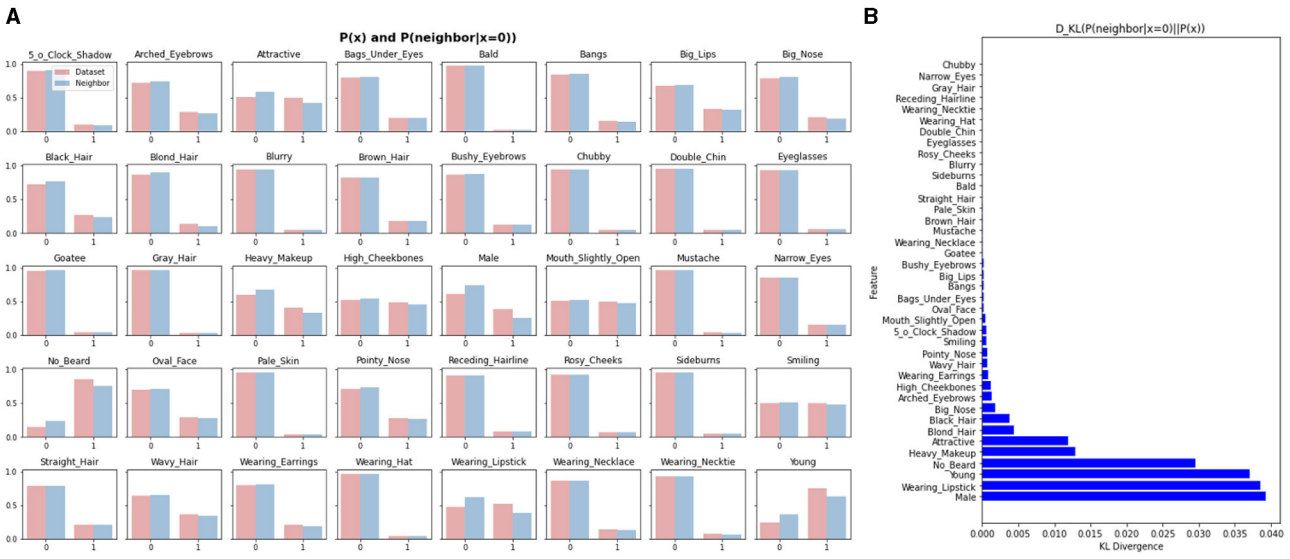


FIGURE 15
Semantic structure of the face recognition manifold by jointly studying the attribute manifold and the facial recognition manifold. **(A)** $p(x)$ and $p(x|x$ of nearest neighbor $= 0)$ distributions. **(B)** KL divergence.

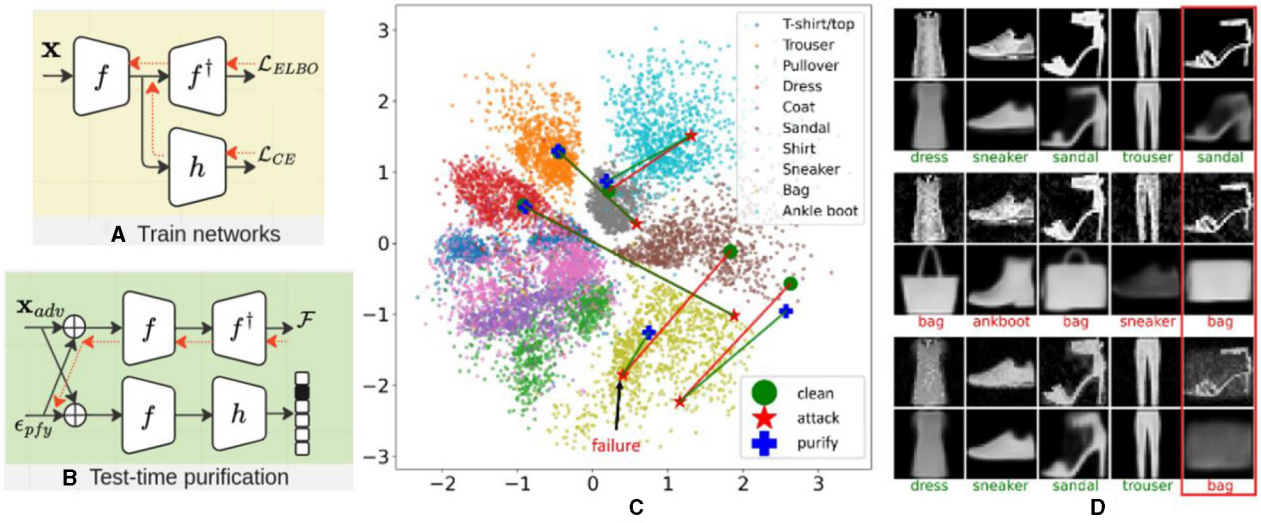


FIGURE 16
The framework of adversarial purification for image-level adversarial attacks. **(A)** Jointly train the classifier with the ELBO loss. **(B)** Test time adversarial purification with the ELBO loss. **(C)** Trajectories of clean (green)—attack (red)—purified (blue) images on a 2D latent space. **(D)** Input images and reconstruction images of samples in **(C)**. The top two rows are the input and reconstruction of clean images, the middle two rows are the input and reconstruction of adversarial images. The bottom two rows are the input and reconstruction of purified images. The text represents predicted classes with green color for correct predictions and red color for incorrect predictions. The red box on the right corresponds to the failure case (purified process fails).

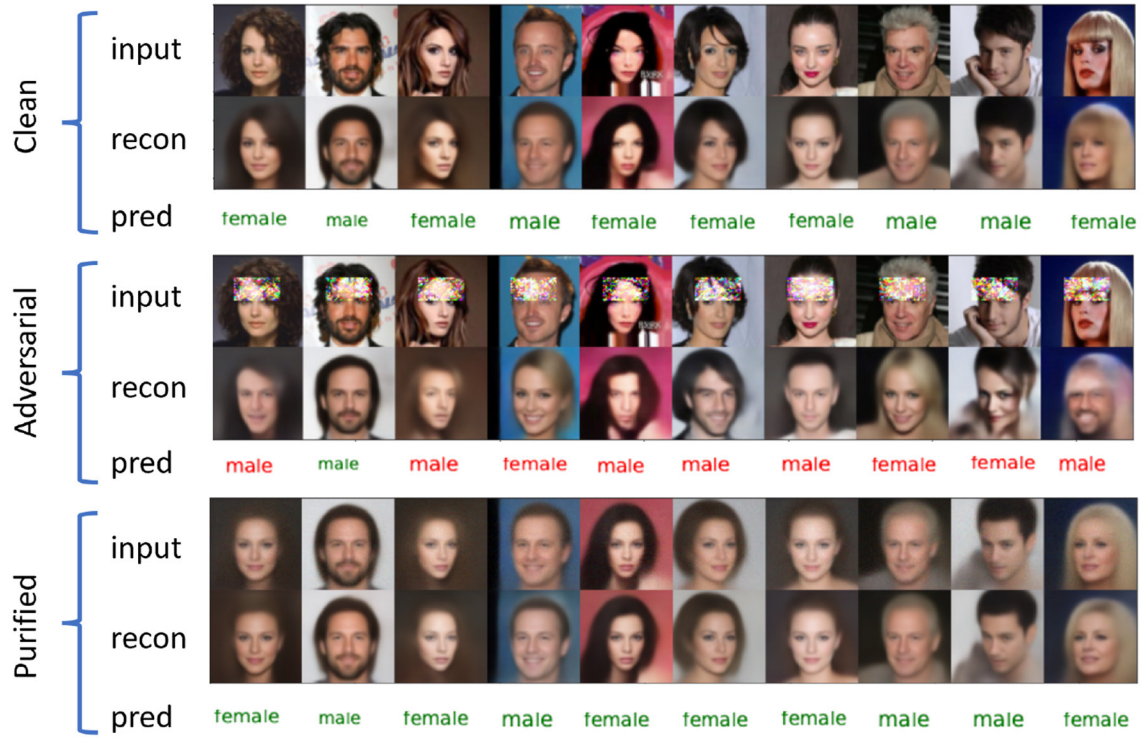


FIGURE 17
Class predictions from the VAE-Classifier models on clean, adversarial and purified samples of the CelebA gender attribute. The top two rows are the input and reconstruction of clean images, the middle two rows are the input and reconstruction of patch adversarial images. The bottom two rows are the input and reconstruction of purified images. The text represents the predicted classes with green color for correct predictions and red color for incorrect predictions. Since predictions and reconstructions from the VAE classifier are correlated, our test-time defenses are effective against adversarial attacks.

TABLE 1 Classification accuracy of the model on clean and adversarial (patch) examples.

Dataset (backbone)	VAE-CLF			+TTD (ELBO)		
	Clean	Patch-PGD	Patch-NAG	Clean	Patch-PGD	Patch-NAG
CelebA-Gender (ResNet-50)	97.86	13.14	6.83	91.20	75.75	76.75

4.2. Bounded patch attack

In this work, we focus on the ℓ_0-bounded attacks (Papernot et al., 2016; Brown et al., 2017) from the manifold perspective which is not investigated in the prior work. In contrast to full image-level attacks like ℓ_2 and ℓ_∞ bounded attacks (Madry et al., 2018), patch attacks, which are ℓ_0 bounded attacks, aim to restrict the number of perturbed pixels. These attacks are more feasible to implement in real-world settings, resulting in border impacts. Below, we conduct an initial investigation into the defense against patch attacks by leveraging the knowledge of the data manifold.

When compared to ℓ_∞ attacks, ℓ_0 attacks, such as the adversarial patch attacks, introduce larger perturbations to the perturbed pixels. Therefore, we decide to remove the purification bound for the patch-attack purification. Without these constraints, the purified examples can take on any values within the image space. A purification vector can then be obtained through the test-time optimization over the ELBO loss as shown in Eq. (30).

4.3. Experiments

We use the gender classification model (Yang et al., 2022) to demonstrate the adversarial purification of ℓ_0 bounded attacks. To ensure that the adversarial examples do not alter the semantic content of the images, we restrict the perturbation region to the forehead of a human face. The patch for perturbation is a rectangular shape measuring 16×32, see Figure 17. For the patch attacks, we conduct 2,048 iterations with step size 1/255 using PGD (Madry et al., 2018) and PGD-NAG (Nesterov Accelerated Gradient) (Lin et al., 2020). In Table 1, the purification is carried out through 256 iterations with the same step size.

5. Limitation

The current version of log-probability estimation in diffusion models has limitations in evaluating high-dimensional images. Specifically, at early denoising steps (when t is small) the diffusion model serves as a denoiser such that $\mathbf{x}_t$ and $\mathbf{x}_{t+h}$ are similar while at large steps (when t moves toward T), their difference is still small due to the high proportion of the Gaussian noise in $\mathbf{x}_t$. This leads to the proportion of the difference between $\mathbf{x}_t$ and $\mathbf{x}_{t+h}$ for effective out-of-distribution detection small compared with the $\log p$ accumulated in the processes. We keep this as an open problem for future work.

6. Conclusion

This work studies the image geometric representation from high-dimensional spatial space to low-dimensional latent space on the image manifold. To explore the image probability distribution with the assumption that real images are usually in a high-density region while not all samples from the distribution can be represented as realistic images, we incorporate log-likelihood estimation into the procedures of normalizing flows and diffusion models. Meanwhile, we explore the hierarchical normalizing flow structure and a higher-order solution in diffusion models for high-quality and high-fidelity image generation. For an interpretable and controllable sampling from the semantic distribution on the manifold, we then propose GridVAE model under an EL framework to disentangle the elements of the latent variable on the image manifold. To test the semantic and reconstruction robustness on the manifold, we first apply patch attacks and defenses in the image space and then effectively recover the semantics under such attacks with our purification loss. Experiments show the effectiveness of probability estimation in distinguishing seen examples from unseen ones, the quality and the efficiency with large sampling steps in image generation, meaningful representations of varying specific element(s) of the latent variable to control the object attribute(s) in the image space, and the well-preserved semantic consistency with patch attacks.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html, http://yann.lecun.com/exdb/mnist/, and https://github.com/zalandoresearch/fashion-mnist.

Ethics statement

Written informed consent was not obtained from the individual(s) for the publication of any potentially identifiable images or data included in this article because these human face images are from the public dataset CelebA, which is widely used in computer vision community.

Author contributions

PT: Writing—original draft, Writing—review & editing. ZY: Writing—original draft, Writing—review & editing. RH: Writing—original draft, Writing—review & editing. ZX: Writing—original draft, Writing—review & editing. JZ: Writing—original draft,

Writing—review & editing. YF: Writing—original draft, Writing—review & editing. DC: Writing—review & editing. JS: Writing—review & editing. TW: Writing—review & editing.

Funding

This work was supported by the DARPA geometries of learning (GoL) project under the grant agreement number HR00112290075.

Acknowledgments

We thank Amir Rahimi for his contribution to the code and discussion of the normalizing flow models.

Conflict of interest

PT, ZY, and YF are employed by General Electric.

The remaining 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.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2023.1253682/full#supplementary-material

References

Bellman, R. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. (2017). "Adversarial patch," in *Conference on Neural Information Processing Systems (NeurIPS)* (Long Beach). Available online at: https://nips.cc/Conferences/2017

Carlini, N., and Wagner, D. A. (2016). "Towards evaluating the robustness of neural networks," in *CoRR abs/1608.04644*.

Chaabouni, R., Kharitonov, E., Bouchacourt, D., Dupoux, E., and Baroni, M. (2020). Compositionality and generalization in emergent languages. *arXiv*. [preprint]. doi: 10.48550/arXiv.2004.09124

Chang, L., Borenstein, E., Zhang, W., and Geman, S. (2017). Maximum likelihood features for generative image models. *Ann. Appl. Stat*. 11, 1275–1308. doi: 10.1214/17-AOAS1025

Chen, R. T. Q., Li, X., Grosse, R., and Duvenaud, D. (2018). "Isolating sources of disentanglement in vaes," in *Conference on Neural Information Processing Systems (NeurIPS)* (Montreal, QC). Available online at: https://nips.cc/Conferences/2018

Chou, E., Tramer, F., and Pellegrino, G. (2019). "SentiNet: detecting localized universal attacks against deep learning systems," in Deep Learning and Security Workshop (DLSW) (San Francisco, CA: IEEE). doi: 10.1109/SPW50608.2020.00025

Coeurdoux, F., Dobigeon, N., and Chainais, P. (2022). "Sliced-Wasserstein normalizing flows: beyond maximum likelihood training," in *European Symposium on Artificial Neural Networks (ESANN)* (Bruges). doi: 10.14428/esann/2022.ES2022-101

Ding, Z., Xu, Y., Xu, W., Parmar, G., Yang, Y., Welling, M., et al. (2020). "Guided variational autoencoder for disentanglement learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA: IEEE). doi: 10.1109/CVPR42600.2020.00794

Gao, J., He, D., Tan, X., Qin, T., Wang, L., Liu, T. Y., et al. (2019). "Representation degeneration problem in training natural language generation models," *International Conference on Learning Representations (ICLR)* (New Orleans, LA). Available online at: https://iclr.cc/Conferences/2019

Gomtsyan, M., Mokrov, N., Panov, M., and Yanovich, Y. (2019). "Geometry-aware maximum likelihood estimation of intrinsic dimension," in *Proceedings of Machine Learning Research*. Asian Conference on Machine Learning (ACML). Available online at: https://www.acml-conf.org/2019/

Grover, A., Dhar, M., and Ermon, S. (2018). "Flow-GAN: combining maximum likelihood and adversarial learning in generative models," in *AAAI Conference on Artificial Intelligence (AAAI)*. (New Orleans, LA). Available online at: https://dblp.org/db/conf/aaai/aaai2018.html

Hajri, H., Said, S., and Berthoumieu, Y. (2017). "Maximum likelihood estimators on manifolds," in *International Conference on Geometric Science of Information* (Cham: Springer). doi: 10.1007/978-3-319-68445-1_80

Havrylov, S., and Titov, I. (2017). "Emergence of language with multi-agent games: learning to communicate with sequences of symbols," in *Conference on Neural Information Processing Systems (NeurIPS)* (Long Beach, CA), 30. Available online at: https://dblp.org/db/conf/nips/nips2017.html

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 770–778. doi: 10.1109/CVPR.2016.90

Ho, J., Jain, A., and Abbeel, P. (2020). *Denoising diffusion probabilistic models*. Conference on Neural Information Processing Systems (NeurIPS).

Hu, H. Y., Wu, D., You, Y. Z., Olshausen, B., and Chen, Y. (2023). RG-Flow: a hierarchical and explainable flow model based on renormalization group and sparse prior. *arXiv*. [preprint]. doi: 10.48550/arXiv.2010.00029

Hwang, R. H., Lin, J. Y., Hsieh, S. Y., Lin, H. Y., and Lin, C. L. (2023). Adversarial patch attacks on deep-learning-based face recognition systems using generative adversarial networks. *Sensors* 23, 853. doi: 10.3390/s23020853

Kingma, D. P., and Dhariwal, P. (2018). "Glow: generative flow with invertible 1x1 convolutions," in *Conference on Neural Information Processing Systems (NeurIPS)* (Montreal, QC). Available online at: https://dblp.org/db/conf/nips/nips2018.html.

Kingma, D. P., and Welling, M. (2013). Auto-encoding variational bayes. *arXiv*. [preprint]. doi: 10.48550/arXiv.1312.6114

Klein, S., Raine, J. A., and Golling, T. (2022). Flows for flows: training normalizing flows between arbitrary distributions with maximum likelihood estimation. *arXiv*. [preprint]. doi: 10.48550/arXiv.2211.02487

Kobyzev, I., Prince, S., and Brubaker, M. A. (2019). Normalizing flows: introduction and ideas. *arXiv*. [preprint]. doi: 10.48550/arXiv.1908.09257

Krizhevsky, A. (2009). "*Learning multiple layers of features from tiny images*," in Technical Report. University of Toronto. Available online at: https://learning2hash.github.io/publications/cifar2009learning/

Kubricht, J. R., Santamaria-Pang, A., Devaraj, C., Chowdhury, A., and Tu, P. (2020). Emergent languages from pretrained embeddings characterize latent concepts in dynamic imagery. *Int. J. Semant. Comput*. 14, 357–373. doi: 10.1142/S1793351X20400140

Kutta, W. (1901). Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys*. 46, 435–453.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*. doi: 10.1109/5.726791

Liang, J., Lugmayr, A., Zhang, K., Danelljan, M., Gool, L. V., Timofte, R., et al. (2021). "Hierarchical conditional flow: a unified framework for image super-resolution and image rescaling," in *IEEE International Conference on Computer Vision (ICCV)* (Montreal, QC: IEEE). doi: 10.1109/ICCV48922.2021.00404

Lin, J., Song, C., He, K., Wang, L., and Hopcroft, J. E. (2020). "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *International Conference on Learning Representations (ICLR)*. Available online at: https://iclr.cc/Conferences/2020

Ling, J., Wang, Z., Lu, M., Wang, Q., Qian, C., Xu, F., et al. (2022). "Semantically disentangled variational autoencoder for modeling 3D facial details," *Transactions on Visualization and Computer Graphics*. IEEE. doi: 10.1109/TVCG.2022.3166666. Available online at: https://ieeexplore.ieee.org/document/9756299

Liu, A., Wang, J., Liu, X., Cao, B., Zhang, C., Yu, H., et al. (2020). "Bias-based universal adversarial patch attack for automatic check-out," in *European Conference on Computer Vision (ECCV)* (Cham: Springer). doi: 10.1007/978-3-030-58601-0_24

Liu, B., Yang, F., Bi, X., Xiao, B., Li, W., Gao, X., et al. (2022). "Detecting generated images by real images," in *European Conference on Computer Vision (ECCV)* (New York, NY: ACM). doi: 10.1007/978-3-031-19781-9_6

Liu, Y., Wei, F., Shao, J., Sheng, L., Yan, J., Wang, X., et al. (2018). "Exploring disentangled feature representation beyond face identification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT: IEEE). doi: 10.1109/CVPR.2018.00222

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)* (Santiago: IEEE). doi: 10.1109/ICCV.2015.425

Lobato, G. A., Mier, P., and Navarro, M. A. A. (2016). Manifold learning and maximum likelihood estimation for hyperbolic network embedding. *Appl. Netw. Sci.* 1, 10. doi: 10.1007/s41109-016-0013-0

Luo, C. (2022). Understanding diffusion models: a unified perspective. *arXiv*. [preprint]. doi: 10.48550/arXiv.2208.11970

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*. (Vancouver, CA). Available online at: https://iclr.cc/Conferences/2018

Mu, Y., Yao, S., Ding, M., Luo, P., and Gan, C. (2023). "EC^2: emergent communication for embodied control," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vancouver, BC). doi: 10.1109/CVPR52729.2023.00648

Pang, A. S., Kubricht, J., Chowdhury, A., Bhushan, C., and Tu, P. (2020). "Towards emergent language symbolic semantic segmentation and model interpretability," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.* 2, 1–64. doi: 10.5555/3546258.3546315

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., Swami, A., et al. (2016). "The limitations of deep learning in adversarial settings," in *EuroS&P* (Saarbruecken: IEEE), 372–387. doi: 10.1109/EuroSP.2016.36

Parmar, G., Li, D., Lee, K., and Tu, Z. (2021). "Dual contradistinctive generative autoencoder," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Nashville, TN: IEEE). doi: 10.1109/CVPR46437.2021.00088

Pastrana, R. (2022). Disentangling variational autoencoders. *arXiv*. [preprint]. doi: 10.48550/arXiv.2211.07700

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830. doi: 10.5555/1953048.2078195

Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. (2021). "The intrinsic dimension of images and its impact on learning," in *International Conference on Learning Representations (ICLR)*. Available online at: https://iclr.cc/Conferences/2021

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., et al. (2020). "Zero-shot text-to-image generation," in *International Conference on Machine Learning (ICML)*. Available online at: https://icml.cc/Conferences/2020

Rezende, D. J., and Mohamed, S. (2016). "Variational inference with normalizing flows," in *International Conference on Machine Learning (ICML)*.

Runge, C. D. T. (1895). Über die numerische auflösung von differentialgleichungen. *Math. Annal.* 46, 167–178. doi: 10.1007/BF01446807

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). "Facenet: a unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA: IEEE), 815–823. doi: 10.1109/CVPR.2015.7298682

Song, J., Meng, C., and Ermon, S. (2021). "Denoising diffusion implicit models," in *International Conference on Learning Representations (ICLR)*. Available online at: https://iclr.cc/Conferences/2021

Tramer, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel,P., et al. (2017). "Ensemble adversarial training: attacks and defenses," in *International Conference on Learning Representations (ICLR)* (Toulon). Available online at: https://iclr.cc/archive/www/doku.php%3Fid=iclr2017:main.html

Tucker, M., Li, H., Agrawal, S., Hughes, D., Sycara, K., Lewis, M., et al. (2021). "Emergent discrete communication in semantic spaces," in *Conference on Neural Information Processing Systems (NeurIPS)*. Available online at: https://nips.cc/Conferences/2021

Tyshchuk, K., Karpikova, P., Spiridonov, A., Prutianova, A., Razzhigaev, A., Panchenko, A., et al. (2023). On isotropy of multimodal embeddings. *Information* 14, 392. doi: 10.3390/info14070392

Voleti, V., Voleti, V., Oberman, A., and Pal, C. (2023). Multi-resolution continuous normalizing flows. *Res. Sq.* Available online at: https://arxiv.org/abs/2106.08462

Wang, S.-Y., Wang, O., Zhang, R., Owens, A., and Efros, A. A. (2020). "CNN-generated images are surprisingly easy to spot... for now," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA: IEEE). doi: 10.1109/CVPR42600.2020.00872

Xiang, C., Bhagoji, A. N., Sehwag, V., and Mittal, P. (2021). "PatchGuard: a provably robust defense against adversarial patches via small receptive fields and masking," *USENIX Security Symposium 2021*. Available online at: https://www.usenix.org/conference/usenixsecurity21

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv*. [preprint]. doi: 10.48550/arXiv.1708.07747

Xu, Z., Niethammer, M., and Raffel, C. (2022). "Compositional generalization in unsupervised compositional representation learning: a study on disentanglement and emergent language," in *Conference on Neural Information Processing Systems (NeurIPS)* (New Orleans, LA). https://nips.cc/Conferences/2022

Yang, Z., Xu, Z., Zhang, J., Hartley, R., and Tu, P. (2022). Adaptive test-time defense with the manifold hypothesis. *arXiv*. [preprint]. doi: 10.48550/arXiv.2210.14404

Zhang, Q., and Chen, Y. (2021). "Diffusion normalizing flow," in *Conference on Neural Information Processing Systems (NeurIPS)*. Available online at: https://nips.cc/Conferences/2021

frontiers | Frontiers in Big Data

TYPE Original Research
PUBLISHED 14 November 2023
DOI 10.3389/fdata.2023.1274831

OPEN ACCESS

EDITED BY
Yunye Gong,
SRI International, United States

REVIEWED BY
Dzung Phan,
IBM Research, United States
Ajay Divakaran,
SRI International, United States

*CORRESPONDENCE
Turgay Caglar
✉ tcaglar@colostate.edu

†Deceased

RECEIVED 08 August 2023
ACCEPTED 25 October 2023
PUBLISHED 14 November 2023

CITATION
Liu Y, Caglar T, Peterson C and Kirby M (2023) Integrating geometries of ReLU feedforward neural networks. *Front. Big Data* 6:1274831. doi: 10.3389/fdata.2023.1274831

Integrating geometries of ReLU feedforward neural networks

Yajing Liu[1†], Turgay Caglar[2*], Christopher Peterson[1] and Michael Kirby[1]

[1]Department of Mathematics, Colorado State University, Fort Collins, CO, United States, [2]Department of Computer Science, Colorado State University, Fort Collins, CO, United States

This paper investigates the integration of multiple geometries present within a ReLU-based neural network. A ReLU neural network determines a piecewise affine linear continuous map, M, from an input space $\mathbb{R}^m$ to an output space $\mathbb{R}^n$. The piecewise behavior corresponds to a polyhedral decomposition of $\mathbb{R}^m$. Each polyhedron in the decomposition can be labeled with a binary vector (whose length equals the number of ReLU nodes in the network) and with an affine linear function (which agrees with M when restricted to points in the polyhedron). We develop a toolbox that calculates the binary vector for a polyhedra containing a given data point with respect to a given ReLU FFNN. We utilize this binary vector to derive bounding facets for the corresponding polyhedron, extraction of "active" bits within the binary vector, enumeration of neighboring binary vectors, and visualization of the polyhedral decomposition (Python code is available at https://github.com/cglrtrgy/GoL_Toolbox). Polyhedra in the polyhedral decomposition of $\mathbb{R}^m$ are neighbors if they share a facet. Binary vectors for neighboring polyhedra differ in exactly 1 bit. Using the toolbox, we analyze the Hamming distance between the binary vectors for polyhedra containing points from adversarial/nonadversarial datasets revealing distinct geometric properties. A bisection method is employed to identify sample points with a Hamming distance of 1 along the shortest Euclidean distance path, facilitating the analysis of local geometric interplay between Euclidean geometry and the polyhedral decomposition along the path. Additionally, we study the distribution of Chebyshev centers and related radii across different polyhedra, shedding light on the polyhedral shape, size, clustering, and aiding in the understanding of decision boundaries.

KEYWORDS

ReLU feedforward neural networks, binary vectors, polyhedral decomposition, geometries, Chebyshev center, Hamming distance

1 Introduction

ReLU feedforward neural networks (FFNNs) exhibit a number of interesting local and global geometric properties. These networks decompose the input space into convex polyhedra and assign to each data point within the same polyhedron a common linear affine function. This polyhedral decomposition offers a fundamental geometric framework, enabling researchers to comprehend the network's partitioning and modeling of the input space. By investigating these geometric properties, including the decomposition of the input space and the counting of linear regions, researchers can gain profound insights into the expressive power, generalization abilities, and limitations of the network. The following sections will present a thorough literature review focusing on the key aspects that have been extensively examined.

The exploration of neural network mappings, which encompass diverse architectures like convolutional neural networks, residual networks, skip connected networks, and recurrent neural networks, as max-affine spline operators, has been extensively investigated by Balestriero and Baraniuk (2018). Sattelberg et al. (2020) built intuition on how the polyhedral decomposition acts and both how they can potentially be reduced in number and how similar structures occur across different neural networks. The number of polyhedra present in the input space, or within a bounded region thereof, serves as a measure of the network's expressivity and complexity. Bounds, both upper and lower, on the maximum number of attainable polyhedra for a given ReLU FFNN architecture can be found in multiple studies such as Pascanu et al. (2013), Montufar et al. (2014), Raghu et al. (2017), Arora et al. (2018), Serra et al. (2018), Hanin and Rolnick (2019), Hinz and van de Geer (2019), and Safran et al. (2022). In an alternative approach, Wang (2022) investigated local properties of the polyhedra, such as inspheres, hyperplane directions, decision boundaries, and the relevance of surrounding regions, to analyze the behavior of neural networks. Masden (2022) has given a full encoding of the canonical polyhedral complex across all dimensions.

Various algorithms in Xiang et al. (2018), Yang et al. (2020), and Xu et al. (2021) have been devised to compute the precise polyhedral decomposition of the input space by employing a layer-by-layer linear inequality solving approach. For larger decomposition instances, an efficient method proposed by Vincent and Schwager (2021) and Liu et al. (2023), is available, which systematically enumerates all polyhedra within the input space by traversing the neighbors of each polyhedron.

Several studies have delved into the intricate geometric properties of ReLU feedforward neural networks. Notably, Zhang et al. (2018) established a profound connection between ReLU FFNNs and tropical geometry, showcasing their equivalence to tropical rational maps. Ergen and Pilanci (2021) focused their attention on finite-width two-layer ReLU networks and revealed that optimal solutions to the regularized training problem can be characterized as extreme points within a convex set, capitalizing on the advantageous attributes of convex geometry. Additionally, Novak et al. (2018) conducted meticulous sensitivity analyses on trained neural networks, scrutinizing the input-output Jacobian norm and the quantification of linear regions in the realm of image classification tasks.

The research conducted by Jamil et al. (2022) and Jamil et al. (2023) has presented a binary vector representation for individual polyhedra, emphasizing its ability to capture abundant information related to both the data and the neural network. Their research has compellingly demonstrated the practical utility of these binary vectors as highly effective tools for enhancing the explainability of neural networks and facilitating the detection of adversarial instances. Building upon the foundational research mentioned earlier, our primary objective is to construct a comprehensive toolbox that effectively leverages the binary vectors and the associated linear model for polyhedra. By harnessing these tools, we aim to delve into and analyze the intricate geometric properties exhibited by ReLU FFNNs.

In this manuscript, we make several key contributions. Firstly, we formulated the codebase for the toolbox as outlined in our prior work (Liu et al., 2023). This codebase is now accessible to the public, and it can be found at the following URL: https://github.com/cglrtrgy/GoL_Toolbox. Leveraging this toolbox, we delve into the intricate geometries of neural networks, utilizing the Hamming distance as a dissimilarity metric for binary vectors to gain insights into network geometry. Additionally, we employ the bisection method to generate samples with Hamming distances of 1, revealing network connectivity. We further explore Chebyshev centers and polyhedral radii, shedding light on polyhedral shape and size, network clustering, decision boundaries, and generalization capabilities. Our approach is validated through implementations on toy datasets, MNIST, and CIFAR-10 datasets, offering compelling insights into neural network geometries.

The remaining sections of the paper are organized as follows: Section 2 introduces the toolbox, providing a comprehensive overview of its functionalities. Section 3 details the methodologies employed for analyzing the geometries of neural networks. It covers the distance metric used, the application of the bisection method, and the utilization of Chebyshev center analysis. In Section 4, 5, the toolbox and the aforementioned analysis methods are demonstrated through illustrative examples using both toy datasets, the MNIST, and CIFAR-10 dataset. Finally, Section 6 provides a conclusive summary of the paper.

2 Definitions and methods

In this section, we will provide a comprehensive review of the following key aspects: model of ReLU FFNNs, the definition of binary vectors, the linear model for polyhedra decomposed by ReLU FFNNs, and the traversal method employed for listing these decomposed polyhedra. For more in-depth information and references, please refer to Liu et al. (2023).

2.1 ReLU feedforward neural network (FFNNs)

We consider an $(L+1)$-layer FFNN with an input space denoted as $\mathbb{R}^m$ and an output space denoted as $\mathbb{R}^n$. Each hidden layer consists of h_i nodes. The weight matrix and bias vector of layer i are denoted as $W_i \in \mathbb{R}^{h_i \times h_{i-1}}$ and $b_i \in \mathbb{R}^{h_i}$, respectively. The ReLU activation function is applied to the initial L layers, while the final layer does not have an activation function. For a given input $x \in \mathbb{R}^m$, the output in layer i is denoted as $F_i(x) \in \mathbb{R}^{h_i}$. The given notation represents the FFNN as follows:

$$\mathbb{R}^m \xrightarrow[\text{ReLU}]{(W_1,b_1)} \mathbb{R}^{h_1} \xrightarrow[\text{ReLU}]{(W_2,b_2)} \mathbb{R}^{h_2} \to \ldots \to \mathbb{R}^{h_{L-1}} \xrightarrow[\text{ReLU}]{(W_L,b_L)} \mathbb{R}^{h_L} \xrightarrow{(W_{L+1},b_{L+1})} \mathbb{R}^n. \quad (1)$$

The feedforward process of model (1) can be summarized as follows:

1). Layer 0 (Input Layer): Given a data point $x \in \mathbb{R}^m$, it serves as the input to Layer 1.

2). Layer 1 to L (Hidden Layers): The output of x at layer i can be expressed as:

$$F_i(x) = \mathrm{ReLU}(W_i F_{i-1}(x) + b_i) = \begin{bmatrix} \max\{0, w_{i,1}F_{i-1}(x) + b_{i,1}\} \\ \vdots \\ \max\{0, w_{i,h_i}F_{i-1}(x) + b_{i,h_i}\} \end{bmatrix}, \quad (2)$$

where $w_{i,j}$ is the jth row of W_i and $b_{i,j}$ is the jth entry of b_i.

3). Layer $L+1$ (Output Layer): The output of x at layer $L+1$ (also the output of the FFNN) is $W_{L+1}F_L(x) + b_{L+1}$. By iteration, this implies that an FFNN represents an affine mapping given a data point x.

2.2 Binary vector

A ReLU feedforward neural network performs a partitioning of the input space into convex polyhedra (Sattelberg et al., 2020), where each individual polyhedron is associated with a corresponding binary vector representation (Liu et al., 2023). The binary vector is defined based on the output of the ReLU activation function in each hidden layer of model (1). The definition is as follows:

For a given point $x \in \mathbb{R}^m$ to model (1), its binary vector at hidden layer i is defined as

$$s_i(x) = [s_{i,1}(x) \ \dots \ s_{i,h_i}(x)]^\top \in \mathbb{R}^{h_i},$$

where $s_{i,j}(x)$ (with $1 \le j \le h_i$) is defined as follows:

$$s_{i,j}(x) = \begin{cases} 1 & \text{if } w_{i,j}F_{i-1}(x) + b_{i,j} > 0, \\ 0 & \text{if } w_{i,j}F_{i-1}(x) + b_{i,j} \le 0. \end{cases} \quad (3)$$

The binary vector of x in model (1) is obtained by stacking its binary vectors from all hidden layers as follows:

$$s(x) = [s_1^\top(x) \ \dots \ s_L^\top(x)]^\top \in \mathbb{R}^h, \quad (4)$$

where $h = \sum_{i=1}^{L} h_i$ is the total number of nodes across all hidden layers.

It is worth noting that points residing within the same polyhedron exhibit identical binary vectors, thereby allowing each polyhedron to be represented by a single binary vector. Subsequently, the forthcoming section will provide a review of the linear model expressed in terms of the binary vector associated with each polyhedron.

2.3 Linear model for polyhedra

Given a data point x, we assume that its binary vector $s(x)$ is obtained using Equation (4). To ensure consistent directionality in expressing the linear inequalities, a sign vector $s'_i = [s'_{i,1} \ \dots \ s'_{i,h_i}]^\top$ is defined for each hidden layer i, where $s'_{i,j} = 1$ if $s_{i,j} = 0$ and $s'_{i,j} = -1$ if $s_{i,j} = 1$.

Let $\hat{W}_j = W_j \mathrm{diag}(s_{j-1})\hat{W}_{j-1}$ and $\hat{b}_j = W_j \mathrm{diag}(s_{j-1})\hat{b}_{j-1} + b_j$ for $2 \le j \le L$ with $\hat{W}_1 = W_1, \hat{b}_1 = b_1$. The linear model for the polyhedron associated with the binary vector $s(x)$ is expressed as follows:

$$Ax \le c, \quad (5)$$

where $A = [A_1^\top \ A_2^\top \dots \ A_L^\top]^\top$ and $c = [c_1^\top \ c_2^\top \dots \ c_L^\top]^\top$ with $A_j = \mathrm{diag}(s'_j)\hat{W}_j \in \mathbb{R}^{h_i \times m}$ and $c_j = \mathrm{diag}(s'_j)(-\hat{b}_j) \in \mathbb{R}^{h_i}$.

It's essential to highlight that, within the polyhedron defined by the bit vector s, the output of any input x is solely determined by a single affine mapping: $G(x) = W_{L+1}\mathrm{diag}(s_L)\hat{W}_L x + W_{L+1}\mathrm{diag}(s_L)\hat{b}_L + b_{L+1}$.

Each facet of the polyhedron corresponds to a unique linear inequality from (5), indicating the non-redundancy of these inequalities. An active bit within the ith entry of $s(x)$ indicates that the ith linear inequality is essential and not redundant. The following linear program can be used to determine whether the ith linear inequality of (5) is redundant or not.

Let

$$A = \begin{bmatrix} a_1 \ a_2 \ \dots \ a_h \end{bmatrix}^\top \quad \text{and} \quad c = \begin{bmatrix} c^1 \ c^2 \ \dots \ c^h \end{bmatrix}^\top$$

with $a_i \in \mathbb{R}^m$ and $c^i \in \mathbb{R}$. We define $\tilde{A}$ as the matrix obtained by removing the ith row $a_i^\top$ from A, and $\tilde{c}$ as the vector obtained by removing the ith element c^i from c. Consider the following linear program

$$\begin{aligned} &\underset{x}{\text{maximize}} \ \ a_i^\top x \\ &\text{s. t. } \tilde{A}x \le \tilde{c}. \end{aligned} \quad (6)$$

If the optimal objective value of (6) is less than or equal to c^i, it indicates that the ith linear inequality is redundant. In such cases, we can remove the row $a_i^\top$ and the corresponding element c^i from A and c, respectively. This iterative process of removing redundant constraints leads to the formation of the reduced set (A', c'), which represents the minimum set of constraints in (A, c). Moreover, the indices of the active bits in $s(x)$ can be obtained through this process. It is noteworthy that the number of active bits within a binary vector corresponds to the number of nonredundant inequalities present in its linear model (5).

By flipping an active bit in $s(x)$ (switching 1 to 0 or 0 to 1), a binary vector corresponding to a neighboring polyhedron that shares a facet with the given polyhedron can be obtained. The validity of this claim can be demonstrated through a proof by contradiction. This enables the derivation of the binary vectors that determine all polyhedra decomposed by the neural network, along with the corresponding derivation of the associated linear models. This method, known as the traversal method, will be reviewed in the subsequent section.

2.4 Traversal-and-Pruning method

In this section, we shall present the Traversal-and-Pruning method as outlined in Algorithm 1. This method systematically explores all bit vectors that define a polyhedron within the input space through the activation of specific bits.

The method employed in this approach originates from Liu et al. (2023), while the concept of flipping an active bit aligns with

the findings presented in Vincent and Schwager (2021). However, the key distinction lies in the fact that this method generates binary vectors for all polyhedra, which holds significant importance in the subsequent section as it facilitates the exploration of the neural network's underlying geometry. Furthermore, Liu et al. (2023) also demonstrates the capability to enumerate polyhedra within a bounded region, although specific details are omitted in this context.

Require: A pretrained (L+1)-layer ReLU FFNN with weights and biases $\{W_i, b_i\}_{i=1}^{L}$ for all L hidden layers and a random point $x^{(0)}$ in the input space $\mathbb{R}^m$. Denote by $p(x^{(0)})$ the polytope to which $x^{(0)}$ belongs and by $s(x^{(0)})$ the specific bit string that uniquely identifies $p(x^{(0)})$.

Ensure: $\mathcal{P}$: the set of bit vectors that determine all polyhedra in the input space, and the size of $\mathcal{P}$ as the number of polyhedra in the input space.

1) Initialize an empty set $\mathcal{P}$ to store all bit vectors that determine a polyhedron and define a label for each stored bit vector with 1 denoting if we have found all polyhedra adjacent to $p(x^{(0)})$ and 0 otherwise.

2) Calculate the bit vector $s(x^{(0)})$ using (2) and (4), add $s(x^{(0)})$ to the set $\mathcal{P}$, and give a label of 0 to $s(x^{(0)})$.

3) Traverse all the polyhedra adjacent to $p(x^{(0)})$, save their bit vectors to $\mathcal{P}$, and update these bit vectors' labels using the following process:

- Find the active bits of $s(x^{(0)})$ by solving the LP model (6). Without loss of generality, say there are q active bits.
- Flip each of these q active bits of $s(x^{(0)})$ (one at a time), producing q new bit strings $\{\hat{s}^{(j)} : 1 \leq j \leq q\} =: \hat{\mathcal{S}}$, which all differ from $s(x^{(0)})$ by only one bit. Add the elements of $\hat{\mathcal{S}}$ to $\mathcal{P}$ that are not already in $\mathcal{P}$.
- Label the added bit vectors with 0 and switch the label of $s(x^{(0)})$ from 0 to 1, indicating that we have found all of the neighbors of $p(x^{(0)})$ and have added their bit vectors to $\mathcal{P}$.

4) Repeat 3 for bit vectors in $\mathcal{P}$ with label 0 until all the bit vectors in $\mathcal{P}$ have label 1.

Algorithm 1. Traversal-and-Pruning method.

The traversal method exhaustively enumerates all binary vectors, and this assertion can be substantiated through the following argument: Consider a set of binary vectors represented as vertices in a graph, where two vertices are connected if they differ by flipping a single active bit.

- We begin by selecting an initial binary vector arbitrarily. The method identifies the active bits in this vector, flips one active bit at a time to generate neighbors, and continues this process iteratively until all possible neighbors are explored.
- To prove completeness, we employ mathematical induction. In the base case, we establish that the method successfully traverses all binary vectors within a small neighborhood of the initial vector. By the inductive hypothesis, we assume that for any binary vector within a certain radius of the initial vector, the traversal method can reach it through a sequence of active bit flips.
- For the inductive step, we show that the method can extend this reach to binary vectors within an expanded radius. By flipping active bits, we demonstrate that it's possible to reach any binary vector within this extended region. This ensures that the method systematically explores the entire space of binary vectors.
- Moreover, the graph formed by these binary vectors is connected because any binary vector in the graph can be transformed into any other binary vector by a series of single-bit flips, ensuring the existence of a path between any two binary vectors.

As a result, we conclude that the traversal method, starting from an arbitrary binary vector, effectively enumerates all binary vectors in the defined space by flipping active bits. This proof establishes the method's capability to traverse and enumerate all binary vectors systematically.

3 Geometric aspects and methodologies of neural networks

In this section, utilizing the toolbox developed in Section 2, our primary aim is to thoroughly investigate the intricate geometries that underlie neural networks. To achieve this, we leverage the Hamming distance, which serves as a dissimilarity metric based on the binary vectors. Moreover, we employ the bisection method to identify the sample points along the shortest Euclidean path between two given data points, imposing the constraint that neighboring sample points must exhibit a Hamming distance of 1. Additionally, we explore the Chebyshev centers and the corresponding radii of the polyhedra, providing valuable insights into the characteristics of the polyhedral structures.

3.1 Distance metric

The Euclidean distance or L_2 norm is widely adopted as the primary distance metric between two data points, including images. Alternatively, for binary data, the Hamming distance is commonly employed, quantifying the dissimilarity between two binary strings by counting the differing positions. In this manuscript, we establish the Hamming distance between two data points or polyhedra as the count of dissimilar entries in their respective binary vectors, which are obtained using Equation (4) based on a pre-trained FFNN.

It is important to highlight that the Hamming distance between polyhedra serves as an approximation for quantifying the

minimum number of steps required to transition between two polyhedra. This observation establishes the Hamming distance as a valuable metric for capturing the geometric relationship and connectivity among polyhedra. Notably, the effectiveness of the Hamming distance in unveiling the underlying mechanisms of neural network functionality has been substantiated in Jamil et al. (2023). Inspired by these findings, we leverage the Hamming distance between data points or polyhedra to probe the geometric characteristics of a pretrained FFNN.

3.2 Bisection method

The Hamming distance serves as an estimate for determining the shortest path between vertices on the dual graph of the polyhedral decomposition. In our case, we focus on finding the samples along the shortest Euclidean path given two data points, ensuring that the Hamming distance between adjacent samples is precisely 1. To achieve this, we introduce a bisection method, described in Algorithm 2, that allows us to generate data points and their corresponding binary vectors between any two given data points, while satisfying the following properties:

(1) The Hamming distance between two adjacent data points is exactly 1.

(2) The sampled data points align with the same line defined by the initial two data points.

Formulating this mathematically, we consider two data points, denoted as A and B, within the input space. Our objective is to identify a series of points $\{A_i\}_{i=1}^n$ that meet the following criteria:

(1) $A_i = A + \delta_i * (B - A)$, where $\delta_i \in (0, 1)$ for $1 \leq i \leq n$. Here, we set $A_0 = A$ and $A_{n+1} = B$.

(2) The Hamming distance between the bit vectors of A_i and A_{i-1}, for $1 \leq i \leq n + 1$, is equal to 1.

Require:
- A, B: Two given data points.
- Flag $= 1$: Determines whether the algorithm stops.
- $A_{\text{left}} = A$: The starting point of $\{A_i\}$ from 0 to $n+1$.
- $a = 0, b = 1$: Defines the search interval for δ_i.
- δ_{list}: An empty list to store $\{\delta_i\}_{i=1}^n$.

Ensure: $\{\delta_i\}_{i=1}^n$.

While Flag $= 1$:

1) Perform the following calculations:

- Calculate the middle point, δ, as $\frac{a+b}{2}$;
- Calculate C as $A + \delta \cdot (B - A)$;
- Compute the binary vectors for A_{left}, C, and B;
- Compute the Hamming distance between the binary vectors of A_{left} and C, denoted by d_1.

2) Check the Hamming distance, d_1:

- If $d_1 = 1$, return δ;
- If $d_1 > 1$, set $b = \delta$, and return to step 1;
- If $d_1 = 0$, set $A_{\text{left}} = C$ and $a = \delta$, then return to step 1.

Append δ to δ_{list}.

3) Update $a = \delta$ and $A_{\text{left}} = A + \delta \cdot (B - A)$.

4) Compute the Hamming distance between the binary vectors of A_{left} and B, denoted by d_2; If $d_2 = 1$, set Flag $= 0$ to exit the loop; Otherwise, repeat the above process.

Algorithm 2. Bisection method.

3.3 Chebyshev center

In our exploration of convex polyhedra in $\mathbb{R}^N$, various statistical characteristics are of interest. For example, determining the number of d-dimensional faces in P or calculating its volume is a fundamental pursuit. The count of $(N - 1)$-dimensional faces, analogous to the number of active bits, is computable via linear programming. On the contrary, pinpointing the number of zero-dimensional faces (i.e., vertices) poses challenges due to combinatorial complexities. To illustrate, a convex polytope with 30 faces in $\mathbb{R}^{15}$ can possess over 150,000 vertices, rendering calculations intractable in $\mathbb{R}^{1000}$. Estimating the radius and center of the largest inscribed sphere (the Chebyshev center) is achievable through linear programming, while determining the radius and center of the smallest bounding sphere remains infeasible. Similarly, exact volume calculations elude us, but the Chebyshev center provides a coarse approximation. The Gaussian mean width offers another proxy for volume but relies on probabilistic algorithms. Leveraging the Chebyshev center as a representation of the polyhedron's "center" and the associated sphere's radius as a volume indicator, we gain insights into polyhedral attributes, network clustering, decision boundaries, and generalization capabilities.

Consider a bounded set Q. The Chebyshev center refers to the center of the largest inscribed ball within Q, as defined in Boyd and Vandenberghe (2004). In our case, we aim to determine the Chebyshev center and its corresponding radius for a polyhedron that has been decomposed by a pretrained FFNN.

Assume that $A'x \leq c'$ represents a minimal set defining a bounded polyhedron resulting from the decomposition performed by an FFNN. Here,

$$A' = \begin{bmatrix} a'_1 \; a'_2 \; \dots \; a'_l \end{bmatrix}^\top \in \mathbb{R}^{l \times m} \quad \text{and } c' = \begin{bmatrix} c'_1 \; c'_2 \; \dots \; c'_l \end{bmatrix}^\top \in \mathbb{R}^l.$$

Note: In the case of an unbounded polyhedron, the inclusion of bounds on each dimension can be implemented to render it bounded.

To describe the center of the ball inscribed within the polyhedron $A'x \leq c'$, we introduce $x_c \in \mathbb{R}^m$ and $r \in \mathbb{R}$, where x_c represents the center and r denotes the radius. Any point within the ball can be expressed as $x_c + t$, with $||t||_2 \leq r$, and it must satisfy the constraints:

$$a'^\top_i (x_c + t) \leq c'_i \text{ for } 1 \leq i \leq l.$$

We know that $\sup_{\|t\|_2 \le r}\{a'^{\top}_i t\} = r||a'_i||_2$ for $1 \le i \le l$, which allows us to rewrite the constraints as:

$$a'^{\top}_i x_c + r||a'_i||_2 \le c'_i \text{ for } 1 \le i \le l.$$

Therefore, the problem of maximizing the radius r can be formulated as the following optimization problem:

$$\begin{aligned} &\underset{r,x_c}{\text{maximize}} \quad r \\ &\text{s. t. } a'^{\top}_i x_c + r||a'_i||_2 \le c'_i \text{ for } 1 \le i \le l. \end{aligned} \tag{7}$$

Define

$$x = \begin{bmatrix} r \\ x_c \end{bmatrix} \in \mathbb{R}^{m+1}, \quad e = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{m+1}, \text{ and } \hat{A} = \begin{bmatrix} ||a'_1||_2 & a'^{\top}_1 \\ ||a'_2||_2 & a'^{\top}_1 \\ & \vdots \\ ||a'_l||_2 & a'^{\top}_l \end{bmatrix} \in \mathbb{R}^{l\times(m+1)}.$$

The optimization problem (7) can then be reformulated as:

$$\begin{aligned} &\underset{x}{\text{minimize}} \quad -e^{\top}x \\ &\text{s. t. } \hat{A}x \le c'. \end{aligned} \tag{8}$$

Problem (8) is a linear program, which can be effectively solved using the cvxpy package in Python.

4 Examples

In this section, we initially showcase the efficacy of our toolbox using toy datasets and the MNIST dataset. Subsequently, we apply the bisection method and Chebyshev center analysis to the MNIST dataset, enabling a detailed investigation of the intricate geometries present in neural networks.

4.1 Basic FFNN

4.1.1 Toy examples 1: 20 nodes

We initially employed model (1) composed of two hidden layers, each consisting of 10 nodes, to approximate the function $f_1(x_1, x_2) = x_1^2 + x_2^2 - 0.4$. The training of this model was performed using PyTorch, a Python library known for its capabilities in deep learning (Paszke et al., 2019). To create a representative dataset, we uniformly sampled 10,000 points from the interval $[-1, 1]^2$. The training process iterated until a predefined early stopping criterion, based on the convergence of the validation loss, was satisfied.

Figure 1 is generated through the following procedure: (1) Enumerate all the binary vectors using Algorithm 1. (2) Determine the linear model associated with each polyhedron using equation (5). (3) Compute the vertices of each polyhedron using the Python package intvalpy and plot each polyhedron using the vertices.

Figure 1 illustrates the decomposition of polyhedra achieved by the aforementioned model. Within the bounded region $[-1, 1]^2$, Algorithm 1 yields a total of 218 polyhedra, contributing to the overall count of 237 polyhedra in the 2-dimensional space $\mathbb{R}^2$.

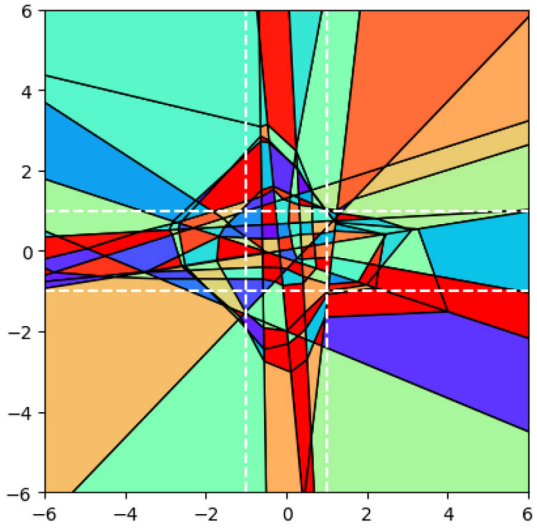


FIGURE 1
2-D visualization of polyhedral composition within $[-6, 6]^2$. The central bounded region, indicated by white dotted lines, encompasses $[-1, 1]^2$.

Among these polyhedra, 211 are classified as bounded, while 26 are deemed unbounded.

Observations: The size of the polyhedra within the training area is relatively small, and their size increases as they move farther away from the training area. Furthermore, for points located on the two white dotted lines at a fixed Euclidean distance, their Hamming distance is greater within the training area and decreases as the points move away from it. To illustrate this, consider the following example: the Hamming distance between the points $(0, -1)$ and $(0, 1)$ is greater than the Hamming distance between the points $(6, -1)$ and $(6, 1)$. This trend highlights how the Hamming distance varies with respect to the proximity to the training area.

4.1.2 Toy examples 2: different model structures

The relationship between the number of polyhedra decomposed by an FFNN and the network depth/width has been extensively explored in prior studies such as Pascanu et al. (2013), Montufar et al. (2014), Raghu et al. (2017), and Hanin and Rolnick (2019).This study aims to investigate the relationship between the mean square error (MSE) of the objective function on the validation dataset and the number of polyhedra. The examination involves exploring different network structures while keeping a consistent number of nodes or layers. By analyzing this relationship, we aim to gain insights into the influence of network structure on the performance of the model.

We conducted a series of experiments where we varied the number of hidden layers while maintaining a consistent number of 5 nodes per layer. Additionally, we adjusted the number of nodes in each hidden layer while keeping a consistent total number of 3 layers. To assess the stability and consistency of the results, we repeated each scenario 50 times with different initializations. The

TABLE 1 Comparison of model architectures: polyhedra count and MSE statistics on validation dataset.

Models		# Polyhedra stats					Model training stats	
# layer	# nodes	Avg	SD	Median	Min	Max	Avg MSE	SD MSE
2	5	55.10	8.28	56	37	70	0.001389	0.000618
4	5	163.33	43.99	156.5	101	277	0.001086	0.001553
6	5	260.70	108.29	249	114	664	0.000947	0.001926
8	5	472.53	232.09	392	265	1,081	0.000593	0.000518
10	5	588.80	181.59	539.5	378	1,072	0.000635	0.000711
3	5	110.20	27.35	105	75	210	0.000651	0.000300
3	10	392.20	64.65	379	302	532	0.000081	0.000021
3	15	788.83	138.14	774	572	1,051	0.000036	0.000014
3	20	1,405.30	233.28	1,376	970	1,938	0.000019	0.000003
3	25	2,225.60	315.68	2,245.5	1,592	3,153	0.000014	0.000002

experimental setup aligns with the details described in Section 4.1.1. The results on the polyhedra count and MSE statistics are listed in Table 1.

The results presented in Table 1 reveal significant trends. Firstly, when the number of nodes is held constant, increasing the number of layers leads to improved average performance and an increase in the number of polyhedra. Moreover, maintaining a fixed number of layers while increasing the number of nodes also results in improvements in both the number of polyhedra and performance.

4.1.3 Toy examples 3: visualizing polyhedral compositions in 3D

Following the same procedure outlined in Section 4.1.1, we utilized a neural network model comprising three hidden layers, each containing 10 nodes, to approximate the function $f_1(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 - 3$. To ensure an accurate representation of the function, we randomly sampled 125,000 points from the range $[-1, 1]^3$. Figure 2 illustrates the decomposition of polyhedra in three-dimensional space. This decomposition encompasses a total of 2,212 polyhedra within the range of $[-1, 1]^3$ in all three dimensions.

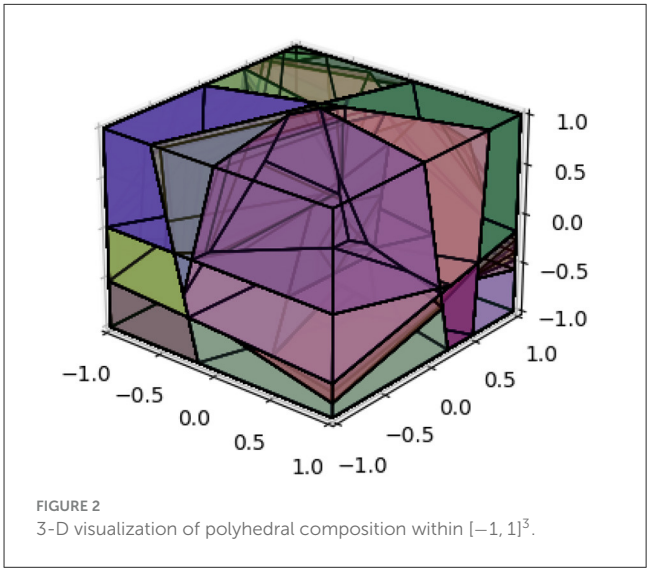


FIGURE 2
3-D visualization of polyhedral composition within $[-1, 1]^3$.

4.2 MNIST

In this experimental study, we conducted our analysis using the well-known MNIST dataset with dimensions of 28 × 28. Model (1), comprising five hidden layers, was trained on the training dataset using five distinct configurations. Notably, the loss function utilized was cross-entropy, optimization was conducted with the Adam optimizer, and the maximum training epochs were limited to 50. The node configurations for these layers were chosen as 300, 350, 400, 450, and 500, respectively, for each of the five structures, surpassing the input dimension of 784. Remarkably, all configurations consistently yielded training and test accuracies surpassing 98%. Subsequently, we randomly selected 30 images from the training dataset and computed their corresponding binary vectors and the linear model (5) representing the polyhedra they belong to for the five different structures. To determine the active bits for each binary vector across the five different model structures, we solved a varying number of instances of model (6), specifically 1,500, 1,750, 2,000, 2,250, and 2,500, by considering different values of i. The coefficient matrix $\tilde{A}$ in the constraint of model (6) had dimensions of $1,499 \times 784$, $1,749 \times 784$, $1,999 \times 784$, $2,249 \times 784$, and $2,499 \times 784$ for the respective instances. To optimize computational efficiency, we leveraged parallel processing on a Linux machine equipped with AMD EPYC 7,452 2.35 GHz processors, utilizing 48 CPUs to efficiently solve model (6) for the corresponding number of times: 1,500, 1,750, 2,000, 2,250, and 2,500, respectively.

Table 2 provides a comprehensive summary of the experimental results, presenting various metrics for different configurations represented by the "Variable nodes per hidden layer" columns. The table includes measurements such as the average (Avg), standard

TABLE 2 Results of active bits for different model structures.

	Variable nodes per hidden layer				
	300 nodes	350 nodes	400 nodes	450 nodes	500 nodes
Avg # active bits	658.30	740.57	780.00	886.87	975.00
SD # active bits	64.07	82.18	97.48	105.19	124.29
Min # active bits	520	525	556	636	719
Max # active bits	785	885	942	1,056	1,181

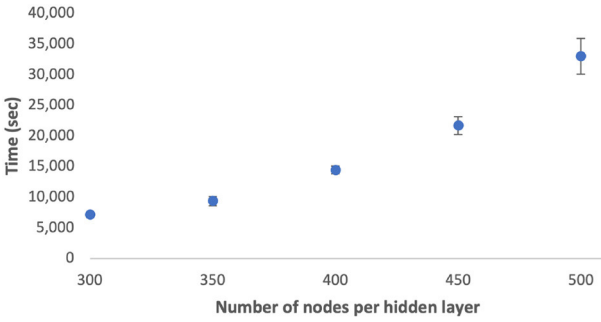


FIGURE 3
Time taken to find active bits vs. node sizes per hidden layer. Error bars represent 1 SD.

deviation (SD), minimum (Min), and maximum (Max) number of active bits.

The results summarized in Table 2 reveal the following insights: As the number of nodes per hidden layer increases (from 300 to 500), the average number of active bits also increases, indicating a positive correlation between the number of nodes and the number of polyhedra; The standard deviation of active bits shows some variation across different configurations, with larger numbers of nodes generally leading to higher variability. The minimum and maximum number of active bits demonstrate an increasing trend as the number of nodes per hidden layer increases.

The computational complexity for solving problem (6) can be described as $\mathcal{O}(m^3h)$. Figure 3 showcases the average and standard deviation of computation time across various node quantities using the Python cvxpy solver. The results demonstrate a positive correlation between the number of nodes per hidden layer and the average computation time. Furthermore, configurations with a larger number of nodes demonstrate increased variability in computation time. In future work, we plan to investigate the scalability of these computations on GPUs.

4.2.1 Hamming distance and bisection method

In this section, we begin by showcasing the efficacy of the Hamming distance in capturing data features and its properties across different source points. We then utilize the bisection method to generate samples along the shortest Euclidean distance path between two designated data points, subsequently analyzing the fluctuations in the number of polyhedra among nearest neighbors as the neural network's layer count undergoes variation.

TABLE 3 Classification accuracy rates of the nearest neighbors for training/test data using Euclidean and Hamming distances.

	Data points	
Metrics	Training	Test
Euclidean	97.37%	95.58%
Hamming for structure 1	99.50%	98.26%
Hamming for structure 2	99.52%	98.06%
Hamming for structure 3	99.58%	98.05%

4.2.1.1 Why using hamming distance?

In this section, we aim to elucidate the rationale behind utilizing the Hamming distance as a representation for the neural network, subsequently leveraging it to delve into the intricate geometric properties inherent within the network.

We examine three distinct structures of the neural network (1). The first comprises 3 layers with 200 nodes per layer, the second consists of 5 layers with 300 nodes per layer, and the third is composed of 5 layers with 500 nodes per layer. The training accuracies for the three structures are 99.08%, 99.01%, and 99.11%, respectively. Correspondingly, the test accuracies for these structures are 97.57%, 97.95%, and 97.64%. We compute the binary vectors corresponding to each training and test data point for the three structures. Subsequently, we determine the nearest neighbor for each data point in both the training and test sets, employing both Euclidean and Hamming distances. Additionally, we investigate whether the nearest neighbor belongs to the same class as the data point under consideration. Table 3 presents the classification accuracy rates of the nearest neighbors for training and test data points belonging to the same class, using both Euclidean distance and Hamming distance for the three structures.

From the table, we observe that the Hamming distance yields higher accuracy rates compared to the Euclidean distance for both training and test data. This suggests that the utilization of the Hamming distance measure leads to more precise classification of data points, resulting in superior classification accuracy rates compared to those obtained with the Euclidean distance measure. The superior performance of the Hamming distance can be attributed to its calculation based on binary vectors, which are derived from the pretrained neural networks. By considering only the differing entries between binary vectors, the

TABLE 4 Comparison of Hamming distance between data points from training/test/adversarial datasets and sampled data points with fixed Euclidean distance.

	Euclidean distance					
	0.5		0.75		1.0	
	Hamming distance		Hamming distance		Hamming distance	
Datasets	Avg	SD	Avg	SD	Avg	SD
Training (Structure 1)	52.97	13.80	102.76	30.90	131.82	35.73
Test (Structure 1)	52.48	13.66	101.40	29.71	130.51	34.43
FGSM (Structure 1)	75.27	13.37	128.01	35.27	151.09	36.89
Training (Structure 2)	99.97	41.88	173.59	69.61	251.73	93.69
Test (Structure 2)	99.36	42.65	170.45	67.51	246.99	90.01
FGSM (Structure 2)	178.57	83.69	264.66	99.79	339.07	106.60
Training (Structure 3)	129.98	56.91	234.34	92.26	329.89	107.36
Test (Structure 3)	129.05	56.73	242.05	87.36	344.97	106.26
FGSM (Structure 3)	219.92	99.16	351.10	113.61	439.77	118.09

TABLE 5 The Hamming distance vs. the number of polyhedra along shortest Euclidean path length.

Pairs number	1	2	3	4	5	6
Hamming distance	98	132	184	232	282	320
# of polyhedra along the shortest Euclidean path	100	135	193	252	299	351

Hamming distance captures crucial features that are highly relevant for determining similarity within the dataset. Consequently, it effectively discriminates between data points and contributes to the improved classification accuracy observed in the results.

Figure 1 illustrates a notable observation: data points that maintain a constant Euclidean distance can exhibit varying Hamming distances as they traverse the input space. This intriguing finding motivates us to explore the potential differences in the Hamming distance when comparing a data point to a sampled data point, while maintaining a fixed Euclidean distance. Specifically, we aim to explore whether the Hamming distance varies based on whether the data point is sourced from the training, test, or adversarial set.

We compute the Hamming distance between a given data point and a sampled data point, while maintaining a fixed Euclidean distance of 0.5, 0.75, and 1, respectively. This analysis encompasses data points sourced from the training, test, and adversarial datasets. The adversarial dataset is generated using the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) and is derived from the test dataset. The results presented herein are based on a dataset comprising 10,000 data points from the training set, as well as the entire test and corresponding adversarial datasets.

Table 4 demonstrates that the average Hamming distance between training data points and their sampled counterparts mirrors that of test data points and their corresponding samples. This consistent behavior underscores the Hamming distance's efficacy in capturing fundamental features and affinities across data points, regardless of their belonging to the training or test set.

However, a notable disparity emerges in the Hamming distances between adversarial data points and their corresponding samples, compared to those of training and test data points. This discrepancy suggests a distinctive and divergent relationship in terms of their binary vector representations. These observations highlight that adversarial examples manifest a substantially dissimilar geometric nature compared to the original training and test data points.

The larger Hamming distances between adversarial data points and their sampled counterparts signify heightened dissimilarity and divergence in their binary vectors. This underscores that adversarial data points occupy a distinct region in the input space separate from both training and test data.

Despite the average Hamming distance between training/test data points and their samples being smaller than that between adversarial data points and corresponding samples, an overlap within the range exists. Consequently, the Hamming distance alone cannot definitively discern the adversarial nature of a point.

4.2.1.2 Bisection method

In this section, we used the bisection method (Section 3.2) to systematically enumerate samples along the shortest Euclidean path between two specified data points. Initially, we analyzed the correlation between the Hamming distance and the number of polyhedra along this path, considering a given data point and its nearest neighbors in the training dataset. Furthermore, we explored how the number of polyhedra changes along the shortest Euclidean path between two nearest neighbors from the training dataset with increasing layer numbers.

Firstly, we aim to demonstrate that the Hamming distance of two given data points does not equate to the number of polyhedra along the shortest Euclidean path between these two

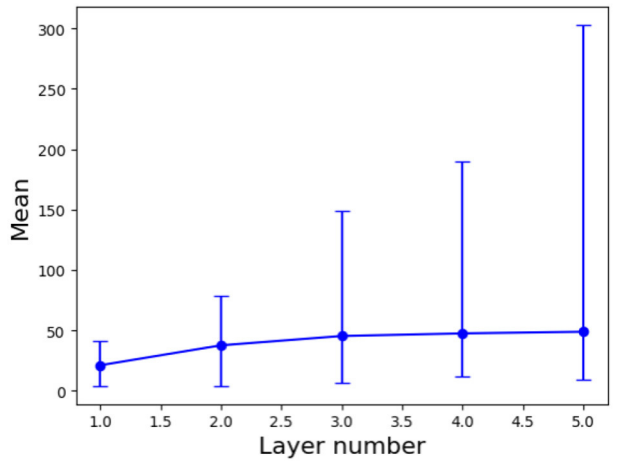


FIGURE 4
Mean values with error bars (maximum and minimum) for the number of polyhedra along the shortest Euclidean path for 1000 pairs of two nearest neighbors versus the number of layers.

given data points. To accomplish this, we computed the Hamming distance for six pairs of nearest neighbors within the train dataset and the results are listed in Table 5. The experiment was carried out using the pretrained model Structure 2, as discussed in the preceding section.

Table 5 illustrates the relationship between the Hamming distance and the number of polyhedra along the shortest Euclidean path between two nearest neighbors (in terms of the Hamming distance). It reveals that when the Hamming distance is small, there is a close correspondence between the Hamming distance and the number of polyhedra along the shortest Euclidean path. However, as the Hamming distance increases, the disparity between the Hamming distance and the number of polyhedra along the shortest Euclidean path becomes more pronounced.

Next, we apply the bisection method to 1000 pairs of nearest neighbors from the training dataset to investigate the variation in the number of polyhedra along the shortest Euclidean distance as the number of layers increases. For this analysis, we utilize the Euclidean distance to determine nearest neighbors, as it remains consistent regardless of any alteration in the neural network structure. We increase the number of layers from 1 to 5 and keep the number of nodes in each layer as 200 for the neural network (1).

Figure 4 illustrates the relationship between the number of layers and the number of polyhedra along the shortest Euclidean path between two nearest neighbors. The results reveal an exponential increase in the maximum number of polyhedra, ranging from 41 to 303, as the number of layers is increased. In contrast, the mean number of polyhedra shows a gradual rise from 21 to 49. This discrepancy suggests that for nearest neighbor pairs with larger Euclidean distances, the number of polyhedra changes significantly with the addition of layers, while most nearest neighbor pairs exhibit a relatively slow change in the number of polyhedra as the number of layers increases.

4.2.2 Chebyshev center

In this section, we employed the same pretrained models discussed in Section 4.2.1.1, along with the MNIST dataset, to conduct our analysis. Specifically, we randomly sampled 1000 data points from the training, test, and adversarial datasets. For each of these data points, we computed the linear models (6) corresponding to the polyhedra on which they reside. Additionally, we utilized model (8) to solve for the Chebyshev centers and their associated radii. The corresponding results for Structure 2 are presented in Figure 5 and Table 6. Additionally, Figure 6 showcases the distribution of radii for the 1000 data points across the three datasets. It is worth noting that the Chebyshev center for each polyhedron resides in a 784-dimensional space, while Figure 5 provides a visual representation limited to three dimensions.

Figure 5A visually depicts the close proximity of the Chebyshev center between the polyhedra containing randomly selected training and test samples. Additionally, Table 6 and Figure 6 present the similarity in size between the polyhedra for the training and test samples. Conversely, Figure 5B reveals a noticeable disparity in the Chebyshev center between the polyhedra encompassing the randomly selected training and adversarial data points. Furthermore, Table 6 and Figure 6 highlight the comparatively smaller size of the polyhedra housing the adversarial data samples in relation to the polyhedra encompassing the training and test data samples. These findings are consistent with the observations from Table 4, which indicates larger Hamming distances when sampling the original points from the adversarial dataset, while maintaining a fixed Euclidean distance.

The above observations provide insights into the behavior and characteristics of the neural network.

The close proximity of the Chebyshev centers and the similarity in size between the polyhedra containing the training and test samples suggest that the neural network exhibits consistent behavior and decision boundaries for these two datasets. This indicates that the network generalizes well and maintains stability in its predictions when presented with new test samples.

On the other hand, the noticeable disparity in the Chebyshev centers and the smaller size of the polyhedra for the adversarial data points indicate that the neural network behaves differently when faced with adversarial inputs. Adversarial examples are intentionally designed to mislead the network and exploit vulnerabilities in its decision-making process. The observed differences in the Chebyshev centers and polyhedra sizes suggest that the network's decision boundaries are more susceptible to manipulation and exhibit variations in response to adversarial inputs.

5 CIFAR-10

To demonstrate the practical applicability of our toolbox and the methodologies outlined in Sections 2 and 3, we conducted experiments using the CIFAR-10 dataset. CIFAR-10 is characterized by its inclusion of real-world images that exhibit heightened complexity, encompassing variations in lighting, orientation, and backgrounds, a notable departure from the MNIST dataset. For training purposes, we employed model (1) with eight hidden layers, each comprised of 400 nodes. The training

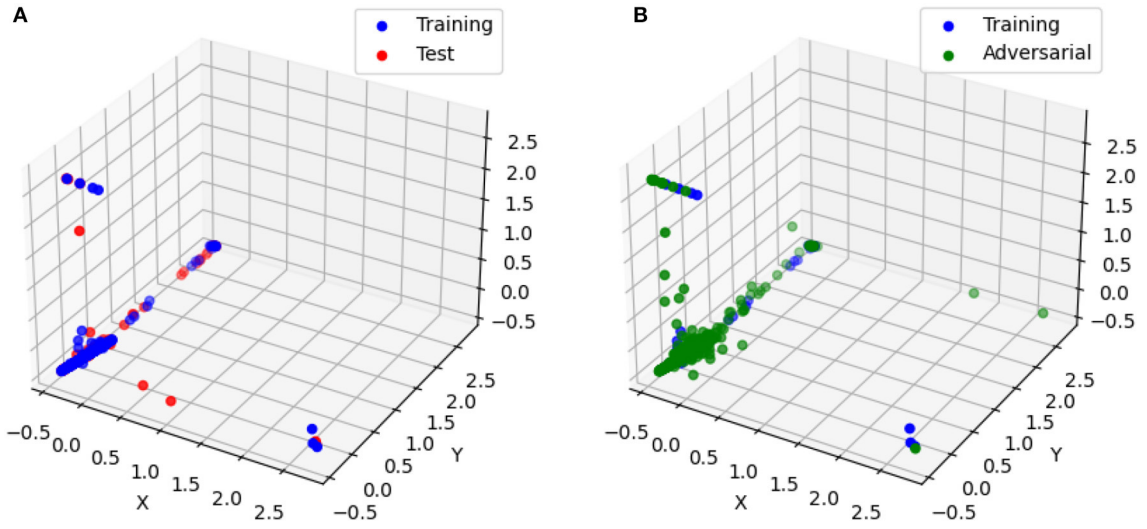


FIGURE 5
The Chebyshev center for randomly selected training, test, and adversarial samples. **(A)** The Chebyshev center for training and test samples. **(B)** The Chebyshev center for training and adversarial samples.

TABLE 6 **Statistics of the radius of the largest inscribed ball within polyhedra: training, test, and adversarial datasets.**

	Mean	SD	Max	Min
Train	0.23	0.081	0.43	0.016
Test	0.21	0.079	0.43	0.018
Adv	0.13	0.092	0.54	0.097

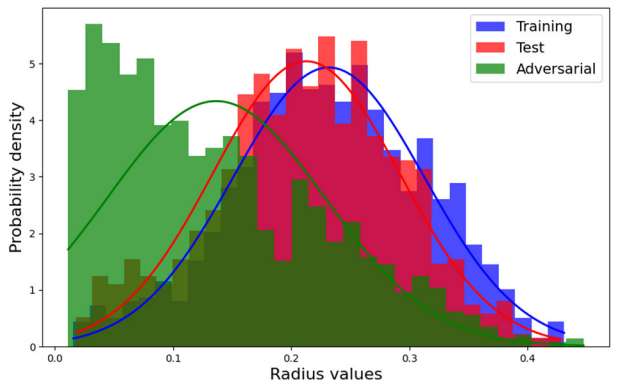


FIGURE 6
Histogram with Gaussian normal distribution for the radius of training, test, and adversarial datasets.

configuration, encompassing the choice of loss function, optimizer, and maximum number of training epochs, remained consistent with the parameters utilized in our MNIST experiments. The training process culminated in a remarkable training dataset accuracy of 99.39%, while the test dataset accuracy reached 53.59%.

In the initial phase of our experimentation, we computed the Hamming distance between a selected data point and a reference data point, maintaining a predefined fixed Euclidean distance of 0.1. These data points were sourced from the training, test, and adversarial datasets. It's essential to note that the adversarial dataset was generated using the same methodology applied to the MNIST dataset. Furthermore, the training dataset consisted of 10,000 data points, aligning with the identical number of data points present in the test and adversarial datasets.

The computed mean Hamming distances across the three datasets reveal values of 109.47, 130.19, and 142.80, accompanied by respective standard deviations of 40.44, 50.66, and 63.42 for the 10,000 data point pairs. Notably, these results diverge from our MNIST experiments, as they indicate a notable dissimilarity in the average Hamming distance between training data points and their corresponding samples compared to that observed in the test dataset. This discrepancy can be attributed to the suboptimal generalization of the trained neural network when applied to the test dataset, resulting in disparate Hamming distance profiles between the training and test datasets.

It's important to underscore the significance of Hamming distance as a lower boundary for quantifying the number of polyhedral boundaries traversed during the trajectory between two polyhedra. The variance in mean Hamming distances among the three datasets implies that, on average, a greater number of polyhedral boundaries exist between an adversarial dataset and its sampled data point.

Subsequently, we conducted a random sampling of 200 data points from the training, test, and adversarial datasets and computed the linear models of the polyhedra within which they

TABLE 7 Statistics of the radius of the largest inscribed ball within polyhedra: training, test, and adversarial datasets.

	Mean	SD	Max	Min
Train	0.038	0.015	0.081	0.015
Test	0.021	0.016	0.069	0.089
Adv	0.017	0.0095	0.051	0.0054

resided. We then calculated the Chebyshev centers and their corresponding radii. The statistics concerning the radii of the polyhedra containing the sampled training, test, and adversarial data points are presented in Table 7. The results shed light on the distinctive nature of the average polyhedral sizes across the three datasets. Notably, the training data points exhibited the most substantial polyhedral size, followed by the test data points, with the adversarial data points displaying the smallest polyhedral size. This observation aligns with the findings derived from the Hamming distance measurements.

Similar to the obersvation on MNIST dataset, the reduced polyhedral size within the adversarial dataset accentuates the network's decision boundaries' susceptibility to manipulation and their propensity to undergo variations when exposed to adversarial inputs. These insights underscore the intricate interplay between geometric characteristics and network behavior, reinforcing the critical need for comprehensive and robust neural network assessments.

The analysis on MNIST and CIFAR datasets has revealed valuable insights into the performance and adaptability of trained neural networks. By employing a combination of Hamming distance and the Chebyshev center method, we are capable of gauging a network's generalization capability and its resilience to real-world data variations and adversarial challenges. These insights not only enhance our understanding of neural network behavior but also provide practical guidance for creating more robust and versatile neural systems capable of effectively navigating the complexities of real-world data and adversarial scenarios.

6 Conclusion

In this work, we present a toolbox for exploring aspects of the polyhedral decomposition (and other associated geometries) of neural networks. Our toolbox allows for the calculation of binary vectors, derivation of polyhedron linear models, extraction of active bits, and enumeration of neighboring polyhedra. Leveraging this toolbox, we investigate the geometric properties of neural networks using the Hamming distance, bisection method, and Chebyshev centers. Through implementation on toy datasets and the MNIST dataset, we validate the effectiveness of our approach and gain insights into the underlying geometries of neural networks. Overall, our work provides a contribution to the understanding and analysis of ReLU neural network structures, decompositions, and behaviors. This paper serves as a proof of concept, laying the foundation for future endeavors. Subsequent work will extend the application of the toolbox and methodologies to conduct comprehensive geometric analyses on much larger real-world datasets together with much more intricate neural network architectures. This includes enhancing model generalization, optimizing model structures, and exploring the design of novel network architectures.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

YL: Conceptualization, Formal analysis, Investigation, Methodology, Software, Supervision, Writing—original draft, Writing—review & editing. TC: Data curation, Formal analysis, Software, Writing—original draft. CP: Supervision, Writing—review & editing. MK: Funding acquisition, Project administration, Supervision, Writing—review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was supported by the DARPA Geometries of Learning Program under Award No. HR00112290074.

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.

Publisher's note

References

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2018). "Understanding deep neural networks with rectified linear units," in *International Conference on Learning Representations*, 1–17.

Balestriero, R., and Baraniuk, R. (2018). "A spline theory of deep learning," in *Proceedings of the 35th International Conference on Machine Learning*, 374–383.

Boyd, S., and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge: Cambridge University Press.

Ergen, T., and Pilanci, M. (2021). Pconvex geometry and duality of over-parameterized neural networks. *J. Mach. Learn. Res.* 22, 1–63.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *arXiv*.

Hanin, B., and Rolnick, D. (2019). "Deep ReLU networks have surprisingly few activation patterns," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 361–370.

Hinz, P., and van de Geer, S. (2019). A framework for the construction of upper bounds on the number of affine linear regions of relu feed-forward neural networks. *IEEE Trans. Inf. Theory* 65, 7304–7324. doi: 10.1109/TIT.2019.2927252

Jamil, H., Liu, Y., Cole, C., Blanchard, N., King, E. J., Kirby, M., et al. (2022). "Dual graphs of polyhedral decompositions for the detection of adversarial attacks," in *2022 IEEE International Conference on Big Data (Big Data)*, 2913–2921.

Jamil, H., Liu, Y., Caglar, T., Cole, C., Blanchard, N., Peterson, C., et al. (2023). "Hamming similarity and graph Laplacians for class partitioning and adversarial image detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Vancouver, BC: IEEE, 590–599.

Liu, Y., Cole, C., Peterson, C., and Kirby, M. (2023). "ReLU neural networks, polyhedral decompositions, and persistent homology," in *the ICML 2023 Workshop on Topology, Algebra, and Geometry in Machine Learning*.

Masden, M. (2022). Algorithmic determination of the combinatorial structure of the linear regions of relu neural networks. *arXiv*.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). "On the number of linear regions of deep neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2924–2932.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv*.

Pascanu, R., Montufar, G., and Bengio, Y. (2013). On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 8026–8037.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). "On the expressive power of deep neural networks," in *International Conference on Machine Learning*, 2847-2854.

Safran, I., Vardi, G., and Lee, J. D. (2022). On the effective number of linear regions in shallow univariate ReLU networks: convergence guarantees and implicit bias. *arXiv*.

Sattelberg, B., Cavalieri, R., Kirby, M., Peterson, C., and Beveridge, R. (2020). Locally linear attributes of ReLU neural networks. *arXiv*.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. (2018). *Bounding and Counting Linear Regions of Deep Neural Networks*.

Vincent, J. A., and Schwager, M. (2021). "Reachable polyhedral marching (RPM): a safety verification algorithm for robotic systems with deep neural network components," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, 9029–9035.

Wang, Y. (2022). "Estimation and comparison of linear regions for ReLU networks," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 3544–3550.

Xiang, W., Tran, H.-D., Rosenfeld, J. A., and Johnson, T. T. (2018). "Reachable set estimation and safety verification for piecewise linear systems with neural network controllers," in *2018 Annual American Control Conference (ACC)*, 1574–1579.

Xu, S., Vaughan, J., Chen, J., Zhang, A., and Sudjianto, A. (2021). Traversing the local polytopes of ReLU neural networks: a unified approach for network verification. *arXiv*.

Yang, X., Tran, H.-D., Xiang, W., and Johnson, T. (2020). Reachability analysis for feed-forward neural networks using face lattices. *arXiv*.

Zhang, L., Naitzat, G., and Lim, L.-H. (2018). Tropical geometry of deep neural networks. *arXiv*.

frontiers | Frontiers in Artificial Intelligence

TYPE Original Research
PUBLISHED 23 November 2023
DOI 10.3389/frai.2023.1255192

OPEN ACCESS

EDITED BY
Yunye Gong,
SRI International, United States

REVIEWED BY
Takashi Kuremoto,
Nippon Institute of Technology, Japan
Daochen Zha,
Rice University, United States

*CORRESPONDENCE
Ben Sattelberg
ben.sattelberg@colostate.edu

RECEIVED 08 July 2023
ACCEPTED 20 October 2023
PUBLISHED 23 November 2023

CITATION
Sattelberg B, Cavalieri R, Kirby M, Peterson C and Beveridge R (2023) Locally linear attributes of ReLU neural networks.
Front. Artif. Intell. 6:1255192.
doi: 10.3389/frai.2023.1255192

Locally linear attributes of ReLU neural networks

Ben Sattelberg[1]*, Renzo Cavalieri[2], Michael Kirby[2], Chris Peterson[2] and Ross Beveridge[1]

[1]Department of Computer Science, Colorado State University, Fort Collins, CO, United States, [2]Department of Mathematics, Colorado State University, Fort Collins, CO, United States

A ReLU neural network functions as a continuous piecewise linear map from an input space to an output space. The weights in the neural network determine a partitioning of the input space into convex polytopes, where each polytope is associated with a distinct affine mapping. The structure of this partitioning, together with the affine map attached to each polytope, can be analyzed to investigate the behavior of the associated neural network. We investigate simple problems to build intuition on how these regions act and both how they can potentially be reduced in number and how similar structures occur across different networks. To validate these intuitions, we apply them to networks trained on MNIST to demonstrate similarity between those networks and the potential for them to be reduced in complexity.

KEYWORDS

neural networks, ReLU, linearization, linear mapping, polyhedral decomposition, Jacobian matrices

1 Introduction

Building a better understanding of neural network behavior is critically important. Neural networks are state-of-the-art in a variety of contexts, including facial recognition (Deng et al., 2019) and object recognition (Russakovsky et al., 2015). However, there is limited understanding of how these networks work or what they are truly doing to achieve such high performance. We present one path for building understanding and intuition by investigating the locally linear behavior of ReLU networks.

ReLU neural networks can be broken into linear region facets—the small polytopes where the network behaves as a linear function based on the activation pattern of the ReLU activation functions. These can be considered both through the underlying geometry of the polytope partitioning of the network and through the linear function associated with the network within each polytope. Prior work has been done on establishing theoretical bounds on the number of regions that it is possible for a network to have (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017) and on investigating metrics involving these structures (Novak et al., 2018).

Much of the original study dealing with the linear regions of ReLU neural networks has focused on investigating expressivity and complexity. It has previously been shown that networks are universal approximators, that is, subject to certain mild constraints, and they are able to approximate any well-behaved function to within arbitrary precision as the size of the network increases (Cybenko, 1989; Hornik, 1991; Hanin and Sellke, 2017; Lu et al., 2017; Lin and Jegelka, 2018). As meaningful as these results are, they are typically not applicable to practical neural networks and do not say anything about the expressivity of a *given* neural network. To assist with determining the expressivity of networks in practice, various groups found and improved bounds on the maximum number of linear regions that feedforward fully connected ReLU neural networks can attain as functions of their width, the number of

nodes in a given layer, and depth, that is, the number of layers (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017). The main result of this study is that the maximum number of linear regions a network can have grows polynomially in the width and exponentially in the depth (Raghu et al., 2017). This partially explains the success of the trend in many modern neural networks to go deeper, such as ResNet (He et al., 2016).

However, empirical investigations of the number of linear regions actually achieved by many neural networks have shown different results. Untrained neural networks after initialization have a number of linear regions that tends to grow linearly in the number of ReLU functions along any one-dimensional subspace of the input space (Hanin and Rolnick, 2019a). Furthermore, after training, the number of regions tends to grow polynomially in the number of ReLU nodes in the network and exponentially in the dimension of the inputs to the network (Hanin and Rolnick, 2019b).

These linear regions have also been used empirically to measure the sensitivity of neural networks. As will be discussed in Section 2.1, the Jacobian of a neural network at a point, together with the value of the neural network at the point, describes exactly the linear function that agrees with the network in a polytope around that point. Novak et al. (2018) utilized this fact to investigate the effect of hyperparameters on input sensitivity and found that overparameterization can help in generalization. Additionally, they Zhang and Wu (2019) investigated how the linear region structure can be used to predict the quality of a network.

Many of these studies have used visualization methods for the polytope structures of neural networks (Hanin and Sellke, 2017; Hanin and Rolnick, 2019a; Zhang and Wu, 2019). These visualizations are frequently done on MNIST or similar datasets using cross-sections of the input space to better understand how the polytope structure of networks evolves through training or through different training methodologies. We apply such visualizations to toy, two-dimensional input problems so that we can build intuition on problems where the entire relevant input space is viewable.

Liu et al. (2023) investigated the properties of the activation patterns of the ReLU functions as bit strings corresponding to these linear regions, although their method works only for fully connected networks, and they did not extend it to convolutional layers or max-pooling.

In the study by McNeely-White et al. (2019), it was shown that one can apply a linear map to the feature vector (the outputs of the pre-classification layer) of one network to obtain a vector, considered as a feature vector in the second network, that can then be used by the second network for classification while maintaining high accuracy.

Zhang et al. (2018) showed that due to the piecewise linear structure of these neural networks, and under certain assumptions, the set of ReLU neural networks, the set of piecewise linear functions, and the set of tropical rational functions are equivalent. We do not extend our results to the realm of tropical algebra, but we do take inspiration from the concept of the dual as commonly expressed in tropical algebra.

We investigate the behavior of linear region for small networks trained on toy problems where full visualization is possible to build intuition for the behavior and structure of both the polytope geometry and their associated linear functions. Insights from those small networks are extended to larger, more modern networks trained to recognize handwritten digits from the Modified National Institute of Standards and Technology database for handwritten digit recognition (MNIST) (LeCun et al., 1998; Szegedy et al., 2016; Lin and Jegelka, 2018). The first is that clustering these linear regions based on Euclidean distance between the weights of their linear functions can be carried out while preserving much of the original performance of the networks. This implies that networks have significant redundancy at the facet level, aligning with the success of methods for pruning and compressing networks (Frankle and Carbin, 2018; Blalock et al., 2020).

The second main result is that the linear functions associated with linear region of two different networks, trained or fine-tuned on the same problem, can be related by a linear map that maintains high accuracy. This implies that qualitatively different networks result in similar solutions when considered on the polytope level, while also providing a way to identify when two networks may identify different patterns in the input data that they exploit for classification. Identifying when networks converge to similar solutions allows for a stronger ability to determine where different architectures or training methods will be successful.

2 Materials and methods

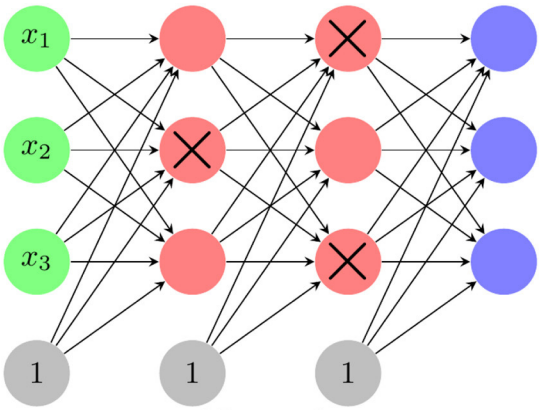
Neural networks with piecewise linear activation functions, such as ReLU, are continuous piecewise linear maps from the input space to the output space (Zhang et al., 2018). Additionally, each of the linear portions of this mapping is supported on a convex polytope defined by the boundaries along which the ReLU nodes activate. Visualizing and analyzing the structure of these linear regions allows for increased understanding of network behavior.

2.1 Linear regions definition

The piecewise linear and convex polytope structures of a ReLU neural network, $f : \mathbb{R}^d \to \mathbb{R}^o$ with inputs in $\mathbb{R}^d$ and o outputs, mean that it can be written as a piecewise linear function (Zhang et al., 2018). A representation of that is

$$f(x) = \begin{cases} \mathbf{W}_1 x + b_1, & \text{if } \mathbf{A}_1 x \leq c_1 \\ \mathbf{W}_2 x + b_2, & \text{if } \mathbf{A}_2 x \leq c_2 \\ \vdots & \\ \mathbf{W}_m x + b_m, & \text{if } \mathbf{A}_m x \leq c_m. \end{cases} \tag{1}$$

For each of the $1 \leq i \leq m$ linear regions, the affine mapping defined by $\mathbf{W}_i$ and b_i is valid on the convex polytope defined by $\mathbf{A}_i$ and c_i. One way to determine these parameters for a given input, x, starts with identifying which ReLU functions are activated for that input. Zeroing the weights in the network associated with deactivated ReLU nodes and converting activated ReLU functions to the identity function, $\mathbf{W}_i$ and b_i can be determined by multiplying through the resulting linear equation. The values of $\mathbf{A}_i$ and b_i can be determined by finding the zeros of the ReLU functions and setting inequalities based on their activation patterns.



Network:

$$f(x_1,x_2,x_3) = \begin{bmatrix} 7 & 9 & 6 \\ 4 & 0 & 9 \\ 7 & 3 & 7 \end{bmatrix} \text{ReLU}\left(\begin{bmatrix} 2 & 9 & 5 \\ 1 & 3 & 3 \\ 8 & 7 & 0 \end{bmatrix} \text{ReLU}\left(\begin{bmatrix} 3 & 0 & 4 \\ 5 & 9 & 1 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 7 \\ 2 \\ 9 \end{bmatrix} \right) + \begin{bmatrix} 7 \\ 7 \\ 8 \end{bmatrix} \right) + \begin{bmatrix} 5 \\ 5 \\ 2 \end{bmatrix}$$

Post ReLU:

$$\begin{aligned} f(x_1,x_2,x_3) &= \begin{bmatrix} 7 & 9 & 6 \\ 4 & 0 & 9 \\ 7 & 3 & 7 \end{bmatrix} \left(\begin{bmatrix} \cancel{2} & \cancel{9} & \cancel{5} \\ 1 & 3 & 3 \\ \cancel{8} & \cancel{7} & \cancel{0} \end{bmatrix} \left(\begin{bmatrix} 3 & 0 & 4 \\ \cancel{5} & \cancel{9} & \cancel{1} \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 7 \\ \cancel{2} \\ 9 \end{bmatrix} \right) + \begin{bmatrix} \cancel{7} \\ 7 \\ \cancel{8} \end{bmatrix} \right) + \begin{bmatrix} 5 \\ 5 \\ 2 \end{bmatrix} \\ &= \begin{bmatrix} 7 & 9 & 6 \\ 4 & 0 & 9 \\ 7 & 3 & 7 \end{bmatrix} \left(\begin{bmatrix} 0 & 0 & 0 \\ 1 & 3 & 3 \\ 0 & 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 3 & 0 & 4 \\ 0 & 0 & 0 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 7 \\ 0 \\ 9 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 7 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} 5 \\ 5 \\ 2 \end{bmatrix} \\ &= \begin{bmatrix} 42 & 0 & 120 \\ 20 & 0 & 27 \\ 336 & 21 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 695 \\ 376 \\ 562 \end{bmatrix} \end{aligned}$$

Region of validity:

$$\begin{bmatrix} 3 & 0 & 4 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 7 \\ 9 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 3 & 3 \end{bmatrix} \left(\begin{bmatrix} 3 & 0 & 4 \\ 5 & 9 & 1 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 7 \\ 2 \\ 9 \end{bmatrix} \right) + 7 \geq 0$$

$$\begin{bmatrix} 5 & 9 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + 2 < 0 \qquad \begin{bmatrix} 2 & 9 & 5 \\ 8 & 7 & 0 \end{bmatrix} \left(\begin{bmatrix} 3 & 0 & 4 \\ 5 & 9 & 1 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 7 \\ 2 \\ 9 \end{bmatrix} \right) + \begin{bmatrix} 7 \\ 8 \end{bmatrix} < \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

FIGURE 1
An illustration of how the ReLU activation pattern for an input determines the linear mapping used for that input. If a given input fails to activate the crossed out red nodes, the corresponding rows of matrices in the functional representation are zeroed. This leads to a single linear function of the input for that input. The region of validity refers to the possible x values for which this ReLU activation pattern exists and is determined by finding inequalities corresponding to the zeros of the ReLU function. The zero for a given node corresponds to the x values for which the output value of the associated matrix row is zero. All the equations must be satisfied.

An example of this process is shown in Figure 1. This piecewise linear mapping structure can be extended to various other common layers types, such as max and average pooling with additional work.

The $\mathbf{W}_i$ and $\mathbf{A}_i$ are linked—the $\mathbf{W}_i$ are selected based on which ReLU nodes are activated, and the $\mathbf{A}_i$ describe where ReLU nodes switch from activated to deactivated or vice-versa. This is partially illustrated in Figure 1 and a specific, smaller example of this is shown later in Equations 2 and 3. There are also similarities and relationships between different $\mathbf{W}_i$ or $\mathbf{A}_i$—because they are coming from the same network weight matrices with rows removed, there is an inherent structure in the specific values used to construct them.

An additional note to make is that the number of regions, m, has the potential to be very large, with exponential growth in the depth and polynomial growth in the width of the network (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017). Experimentally, trained networks have been shown to typically exhibit polynomial growth with the number of ReLU activations of the network, where the degree of the polynomial is the input dimension (Hanin and Rolnick, 2019b). Although this is polynomial, networks applied in domains such as image recognition frequently have inputs with at least 1,000 dimensions, so this still results in very large numbers of regions (Russakovsky et al., 2015).

The linear mapping network definition, Equation 1, highlights the fact that as long as one of the ReLU nodes does not switch from "activated" to "deactivated" or vice-versa, the behavior of the network is purely linear. Since the network is a composition of continuous linear and piecewise linear functions, it is itself a continuous piecewise linear function that splits the input space into disjoint polytopes, on each of which there is an associated affine mapping. This represents an unequivocally simple way to conceptualize what ReLU networks compute, but unfortunately, the typically extreme growth in the number of facets in Equation 1 means enumerating the full set of affine mappings is impractical for most modern networks.

Because of this difficulty in computation, Equation 1 is of conceptual value but, arguably by itself, not of much practical value; however, it leads to several distinct yet ultimately equivalent views of neural networks. Some of the relevant views are:

- The weight matrix, $\mathbf{W}_i$, is the Jacobian of the neural network in the region described by $\mathbf{A}_i$ and c_i. The j^{th} row of $\mathbf{W}_i$ is the gradient of the j^{th} output of the network. This fact has been utilized previously to consider sensitivity metrics for neural networks (Novak et al., 2018). This also allows for simple calculation of the $\mathbf{W}_i$ and b_i values, even in networks with unusual piecewise linear activation functions.
- The weight matrices, $\mathbf{W}_i$, and biases, b_i, form a set of linear maps which the neural network chooses from based on the value of the input. Each row of these $\mathbf{W}_i$ is a surface normal to the hyperplane used for classification.
- The choices are based on the location of the input in a set of connected polytopes induced by the ReLU structure of the network. We provide animations showing how these structures evolve as networks train in Section 3.1.
- Each row of $\mathbf{W}_i$ concatenated with the corresponding element of b_i forms a point in $\mathbb{R}^{d+1}$. These points can be considered as lying in a "dual" space to the corresponding output of the network, and their structure can be analyzed in that context to investigate the linear function behavior. We show how this space forms in Sections 2.2 and 3.1, and analyze this space for clustering and similarity of networks in Sections 3.2 and 3.3.

2.2 Example on XOR

For an example of how the piecewise linear nature of ReLU neural networks works, we consider an XOR problem and a ReLU neural network that solves it as presented in Figure 2. We choose XOR as it is a complex enough problem that it illustrates non-linear aspects of network behavior, but simple enough that full analysis of that behavior is feasible. Note that for the XOR function itself, shown in Figure 2A, zero is replaced with -1 to assist with training of networks. Figure 2B shows a network which solves the XOR problem. The functional form of that same network mapping from the two inputs x and y may be written as

$$f(x,y) = \begin{bmatrix} 2 & -4 \end{bmatrix} \max\left\{ \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} - 1. \quad (2)$$

As a function on $\mathbb{R}^2$, the network divides $\mathbb{R}^2$ into three linear regions with corresponding linear function/polytope pairs,

$$f(x) = \begin{cases} -2x - 2y + 1, & 0 \leq x + y, \text{ Both ReLUs activated} \\ 2x + 2y + 1, & -1 \leq x + y \leq 0, \text{ Top activated; bottom not activated} \\ -1, & x + y \leq -1, \text{ Neither ReLU activated} \end{cases} \quad (3)$$

These linear regions are shown in Figure 3. Even for this very simple example, a complication arises: there is actually a "fourth" region, $-4x - 4y - 1$, tied to the case where the lower ReLU node is activated and the top is not. However, that case occurs in the empty polytope $0 \leq x+y \leq -1$ which cannot occur for any values of x and y, and thus, in practical terms, this empty polytope does not exist. This is an example of a general phenomena where cases exist in principle but are unreachable regardless of input. Furthermore, the existence of such cases explains in part why the number of possible linear regions grows as it does and not simply as a power of two of the number of ReLU functions.

There are additional practical complications that can arise but do not on this network due to its simplicity—a network can be considered as a function on all of its input space, $\mathbb{R}^d$, but the data to which the network is actually applied lie in a bounded region within that space. Polytopes may exist outside of that region but not be meaningful for the purpose of the network. Furthermore, in many problems, the data used are a discrete subset of this bounded region. It is possible for the network to define polytopes lying in the bounded region but too small to contain any of the discrete data to which the network is applied. In general, we observe that the number of polytopes does typically grow beyond the number of actual training samples when considering high-dimensional input data and complex networks.

Returning to the regions shown in Figure 3, the weights and biases in these polytopes can be considered as $d + 1$-dimensional points existing in a "dual space" to the original neural network. For example,

$$-x - y = \begin{bmatrix} -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

and so the point $(-1, -1, 0)$ in the dual is induced by this region. Further examples of these duals are illustrated in Figure 4, which shows the decision boundary, numerical output, and dual points for three networks with varying numbers of nodes trained on the XOR problem. These can illustrate patterns in the behavior of the network, and as will be discussed in more detail, mapping

A

x	y	$f(x,y)$
-1	-1	-1
1	-1	1
-1	1	1
1	1	-1

B

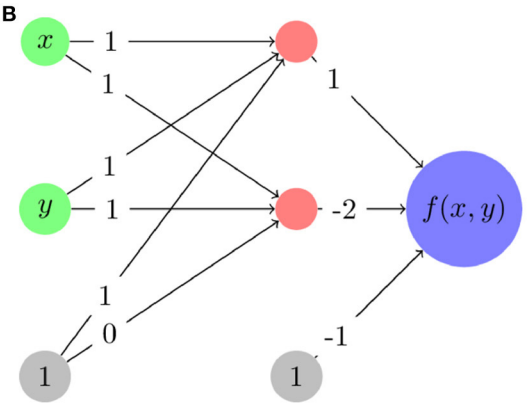


FIGURE 2
The modified XOR problem. **(A)** The input and output values—inputs and outputs are rescaled to be from −1 to 1 rather than from 0 to 1. **(B)** A network architecture and its associated weights that solves this problem. Nodes in red have ReLU applied after calculating their associated input values.

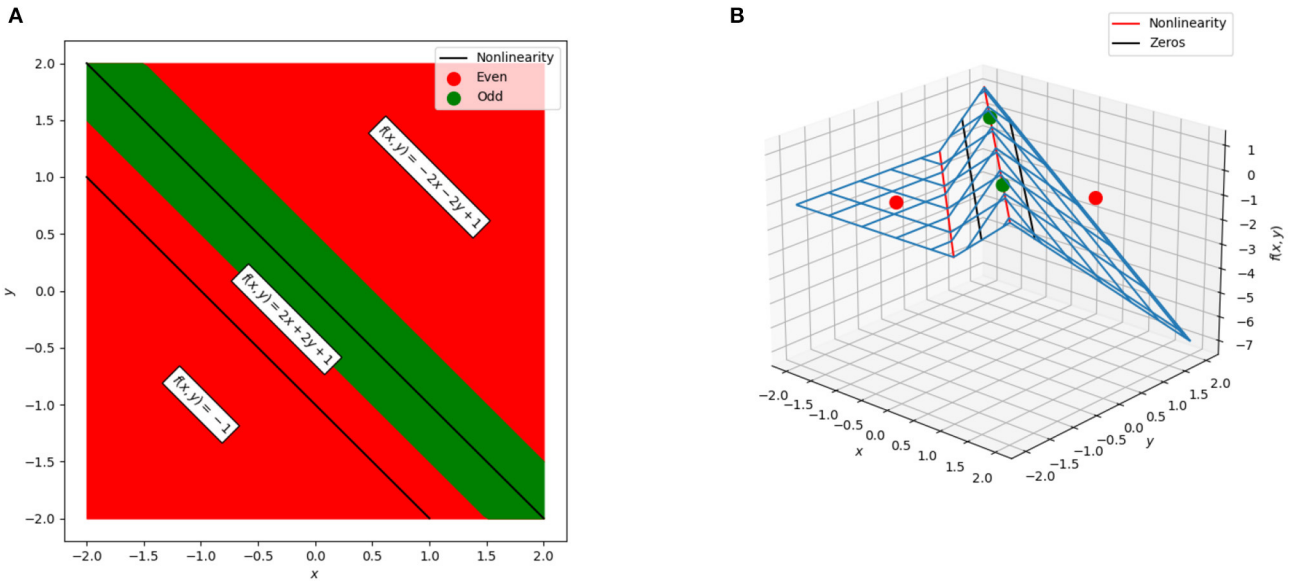


FIGURE 3
The polytopes and associated linear regions for a simple network to solve the XOR problem. **(A)** The cross-section of the network in the plane. Green corresponds to points that would be labeled in the positive class (neural network output greater than zero) and red corresponds to points that would be labeled in the negative class (neural network output less than zero). The black lines correspond to the points at which one of the two ReLU units "activates" or "deactivates" and switches the linear region used for classification. The three polytopes form bands in the plane. **(B)** The surface of the neural network. The points used for training are shown as green and red dots, the non-linearities are shown as red lines, and the decision boundary (zeros of the network) are shown as black lines.

between dual regions of networks or clustering in this space can identify similarity metrics and areas where the neural network gives potentially unnecessary complexity.

2.3 Polytope visualization

One way to think of the polytopes resulting from ReLU activation patterns is the way in which they arise as a consequence of the iterated perceptron structure inherent in this style of network. Each ReLU node in the network builds upon the non-linearities in the previous layers by having its activation boundary correspond to a line in the output space of the previous layer. An example of this is illustrated in Figure 5. This figure shows the decision surface, numerical output, dual points, and the boundaries of the linear regions induced by each node split by layer.

The boundaries induced by the first hidden layer of the network, bottom left of Figure 5, are relatively simple—each of the nodes in the first layer has a line in the original input space for an activation boundary where the output of that node switches

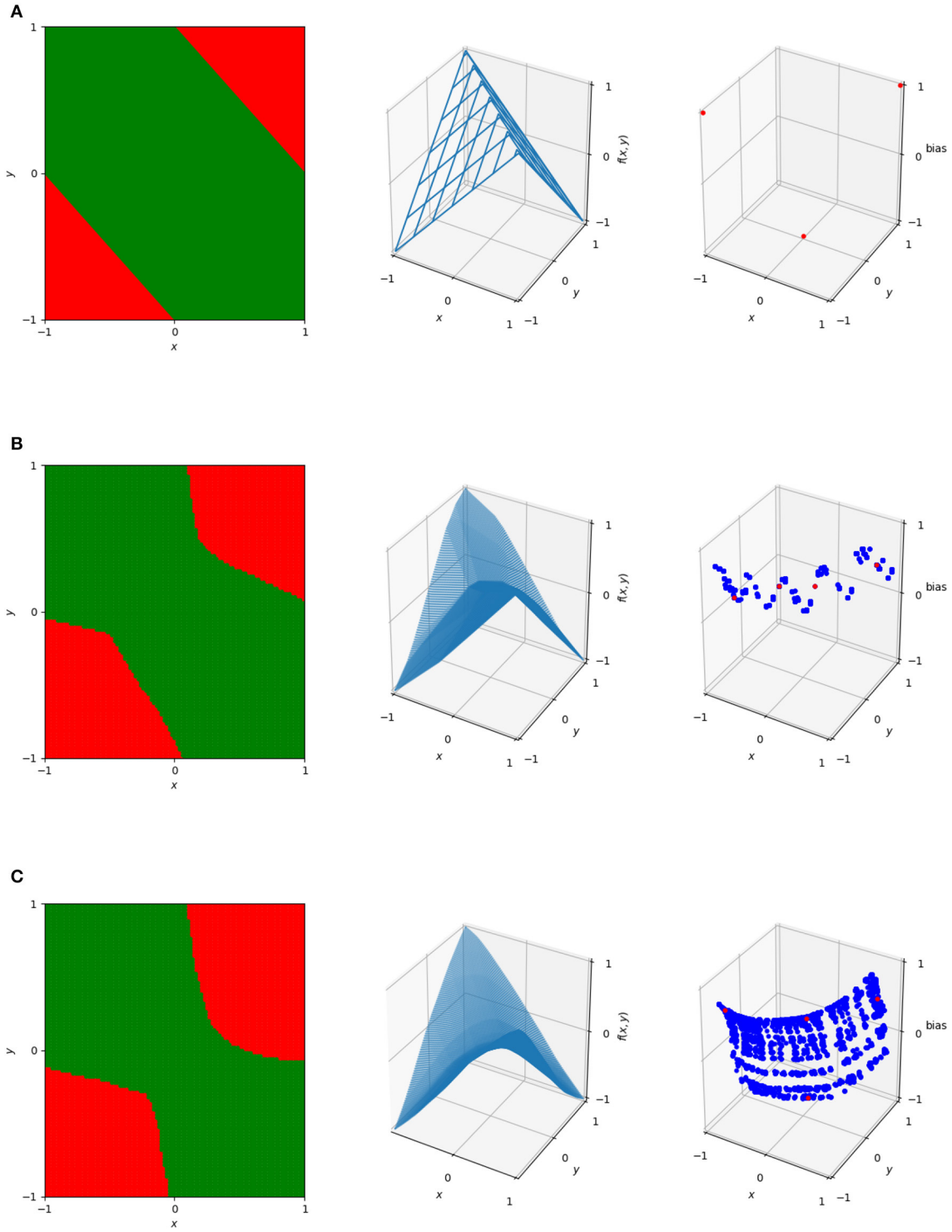


FIGURE 4
The decision boundaries (left), wireframe representations of output (center), and dual representations of the linear regions (right) for three networks designed to solve ReLU. The top network **(A)** is the simple one described previously. The center **(B)** and bottom **(C)** are single hidden layer neural networks with the center having 20 hidden nodes and the bottom having 100 hidden nodes. In the dual, blue dots represent linear regions used on the 101×101 uniform grid in $[-1, 1]^2$. The red dots represent the linear regions used for the actual classification of the four data points—note that the top image only has three dots corresponding to these, rather than four, as it only has a total of three linear regions.

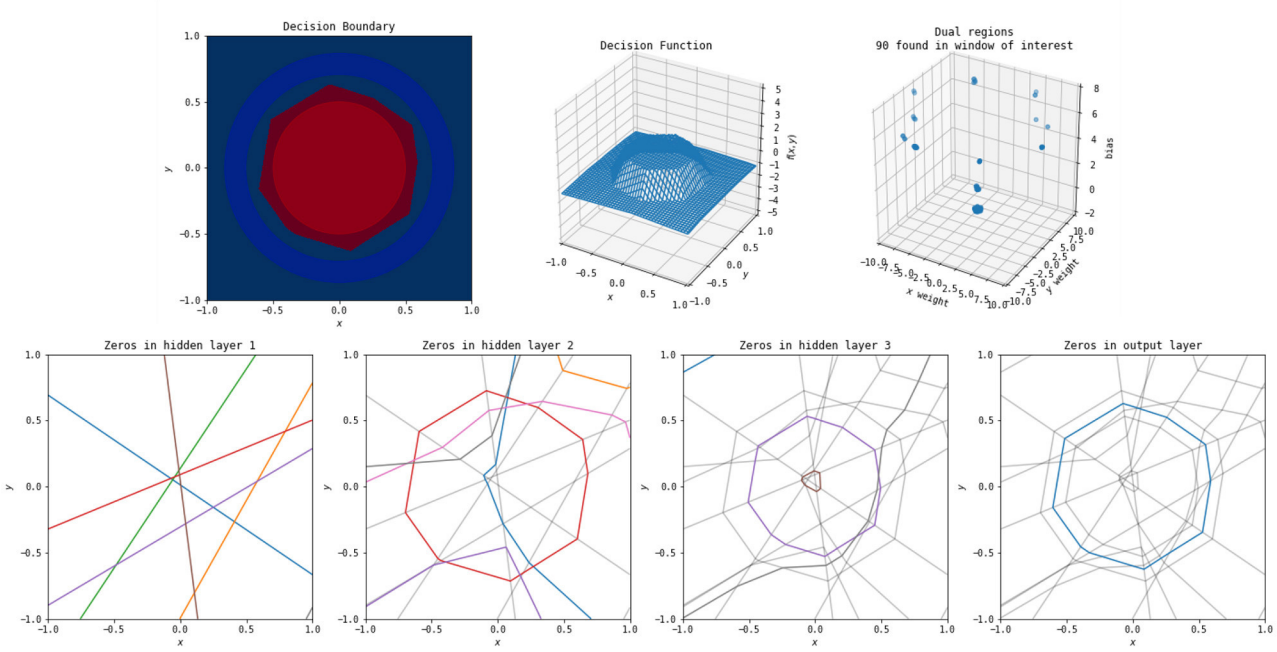


FIGURE 5
The polytopes resulting from the various layers of a simple network to classify a circle versus a surrounding annulus. **Top left**: the original problem and the decision boundary determined by the network. **Top middle**: the outputs of the network. **Top right**: the dual weights. **Bottom first**: the zeros for each of the perceptrons from the original input space to the first hidden layer—these decision boundaries are all lines, as the perceptrons at this stage are purely linear in the original input. Each color corresponds to one of eight nodes in this hidden layer (and the colors do not relate between each of the four bottom plots). Not every node has zeros occurring in the window shown. **Bottom second**: the zeros for each of the perceptrons from the original input to the second hidden layer, with the boundaries for the first hidden layer in light gray. These are lines in the output space of the first layer, but appear non-linear when shown in the original input space. Each boundary can only break at one of the lines from the previous layer. **Bottom third**: the zeros for each of the perceptrons from the original input to the third layer, with the boundaries of the first two layers in gray. Breaks in this layer can occur at any location where it crosses a zero of a previous layer. **Bottom fourth**: zeros in the output layer. This forms the decision boundary shown in the top left.

from positive to negative. Each subsequent layer builds upon the previous. However, the activation boundaries of ReLU nodes in subsequent layers form lines in *compressed* space. Within the polytopes formed by the activation boundaries of previous layers, the new activation boundaries are still linear, but when changing between those polytopes, the new activation boundaries are able to change angle. To illustrate this, the activation boundaries for each layer in Figure 5 are reproduced in subsequent layers in gray. The more complicated activation boundaries for each subsequent layer are always locally linear with changes in direction only arising where they intersect a boundary from a previous layer. This is a direct result and also illustration of the fact that the non-linearities of multi-layer ReLU networks must be built up from activation boundaries established by the previous layers in the network. Finally, notice in the bottom right of Figure 5 that the output layer of the network does as expected, constructing a valid piecewise linear decision boundary for the original classification task.

A few things can be noticed from Figure 5. The first is that the final decision boundary is not reliant on all of the activation boundaries from previous layers. This implies that some of the nodes in the network could be removed without qualitatively impacting the classification. Additionally, the final decision boundary corresponds closely to an activation boundary in the second hidden layer, suggesting that layers after the second are not necessary. These observations support the idea of the "lottery hypothesis," where networks have more nodes that necessary, and subnetworks that are initialized well can be the main driving force for network success (Frankle and Carbin, 2018; Blalock et al., 2020). Finally, the 90 identified linear function weights in the dual graph cluster into a small number of points, again suggesting simplification of the network is possible to remove such redundancy.

2.4 Region modification

To investigate extraneous complexity in networks, it is useful to consider the linear functions that arise on each polytopes. This is useful for a number of reasons, but the two simplest are that the visualizations done in the previous section cannot be done as simply with high-dimension input data, and that the complexity of the polytope partitioning increases significantly with the complexity of the network. Using the linear functions allows us to sample points from the input space and compare the behavior of a network or networks across those points without having to worry about the polytope structure between those points. Even for relatively simple image classification datasets such as MNIST, small networks have nearly every image in the dataset lying on a unique polytope (Novak et al., 2018). Additionally, by calculating the linear functions using the Jacobian, we can largely treat the

network structure as a black box and avoid difficulties that arise from considering more complex activation functions (Liu et al., 2023).

For investigating these affine mappings, there are two useful steps to take: constructing notation to allow us to refer to the set of affine mappings potentially used for a specific output of a network, and considering only the affine mappings that are used for training or testing to reduce the number to something computationally manageable.

In terms of notation, the $\mathbf{W}_i$, which is the Jacobian of the network within the region defined by $\mathbf{A}_i$ and c_i, and b_i described in Equation 1 can be written as

$$\mathbf{W}_i = \begin{bmatrix} w_{i,1}^T \\ w_{i,2}^T \\ \vdots \\ w_{i,o}^T \end{bmatrix} \text{ and } b_i = \begin{bmatrix} b_{i,1} \\ b_{i,2} \\ \vdots \\ b_{i,o} \end{bmatrix}, \tag{5}$$

Where each $w_{i,j}$ and $b_{i,j}$ correspond to the affine mapping in polytope $1 \leq i \leq m$ for the $1 \leq j^{\text{th}} \leq o$ output of the network. Then, it is possible to construct the matrix containing the set of affine mappings used for a given output, j, as

$$\overline{\mathbf{C}}_j = \begin{bmatrix} w_{1,j}^T & b_{1,j} \\ w_{2,j}^T & b_{2,j} \\ \vdots & \\ w_{m,j}^T & b_{m,j} \end{bmatrix} \in \mathbb{R}^{m \times (d+1)}. \tag{6}$$

In practice, it is computationally infeasible to calculate all m linear regions, so for the purpose of empirical studies, we choose p points in the input space to sample and construct the matrix.

$$\mathbf{C}_j = \begin{bmatrix} w_{1,j}^T & b_{1,j} \\ w_{2,j}^T & b_{2,j} \\ \vdots & \\ w_{p,j}^T & b_{p,j} \end{bmatrix} \in \mathbb{R}^{p \times (d+1)}. \tag{7}$$

For simple, two-dimensional input problems, we choose the p points by sampling from a uniform grid. We also consider the MNIST dataset (LeCun et al., 1998), where the p points we sample from are the 60,000 training or 10,000 testing input samples from that network. We construct the $\mathbf{C}_j$ matrices using the training samples, and we additionally construct $\tilde{\mathbf{C}}_j$ using the testing samples for evaluation of how various modifications impact accuracy on the testing set.

2.4.1 Clustering regions

Even for potentially large numbers of sampled affine maps, it is likely that many samples will have a unique $\mathbf{W}_i$ due to the large number of total linear regions. For example, even simple networks on the MNIST dataset only have overlap on <1% of the training inputs. This is not necessarily surprising, simply due to the sheer number of possible linear regions the network can construct.

However, although these weights are not necessarily equivalent, there is potentially a great deal of redundancy or similarity among them. As discussed in Section 2.3 and shown in the behavior of the dual points of increasingly complex networks in Figure 4, patterns appear in the linear weights that can indicate redundant behavior. We can cluster these linear weights and determine how well those clusters are able to replicate the behavior of the network as a measure of that redundancy.

1. Calculate the $\mathbf{C}_j$ and $\tilde{\mathbf{C}}_j$ matrices.
2. Train k-means clustering models using the rows of each of the $\mathbf{C}_j$ matrices.
3. For each row of each of the $\tilde{\mathbf{C}}_j$ determine for which cluster center it is closest.
4. Use that cluster center as a linear mapping from input space to determine the value for that output.
5. Classify the input based on which of the newly calculated outputs is highest.

By varying k and comparing the resulting accuracy against the original accuracy of the network, we can investigate the degree to which networks can be simplified. If applying this method with $k = 1$ results in near-original accuracy, that suggests the network is behaving holistically as a linear mapping, whereas if it results in near-random accuracy, that suggests the network's behavior can not be well described by a linear transformation from the input space to the output space. Determining at which value of k accuracy approaches the original provides a way to understand how significantly the network can be simplified.

2.4.2 Affine maps between linear functions

Another area where representing the weights of these linear regions as points in space can be useful is in finding similarities between two networks. Given $\mathbf{C}_{j,network1}$ and $\mathbf{C}_{j,network2}$, we can train least-squares regression models for each output to find matrices $\mathbf{M}_j \in \mathbb{R}^{d+1 \times d+1}$ for each $1 \leq j \leq o$ that minimize

$$||\mathbf{C}_{j,network1}\mathbf{M}_j - \mathbf{C}_{j,network2}||_2. \tag{8}$$

This method finds a mapping between the linear region weights, or equivalently, between the gradients of the outputs with respect to the input. Due to this, as with the k-means clustering method, this method requires running inputs through each original network, calculating the Jacobians, then applying the transformation.

This is similar to the study by McNeely-White et al. (2019) where the authors demonstrated that the outputs of the final layer before the linear classifier of networks trained on ImageNet are affine-equivalent. Unlike their study, our study investigates the connection between the affine mappings of the locally linear functions of networks, rather than the feature vectors of networks.

Results of this process for XOR networks using the $\mathbf{C}$ matrices constructed by sampling points on the 101×101 uniform grid are shown in Figure 6, which shows the results of mapping between two networks trained on XOR. Although the two networks have similar behavior, their decision boundaries are somewhat different and their dual representations are close to rotations of each other. The resulting points of $\mathbf{C}_{j,network1}\mathbf{M}_j$ are very similar to $\mathbf{C}_{j,network2}$ and vice versa, meaning that the mapping is successful. The function resulting from this is no longer continuous—because the bias is part of what is being mapped, the result is able to vary based on

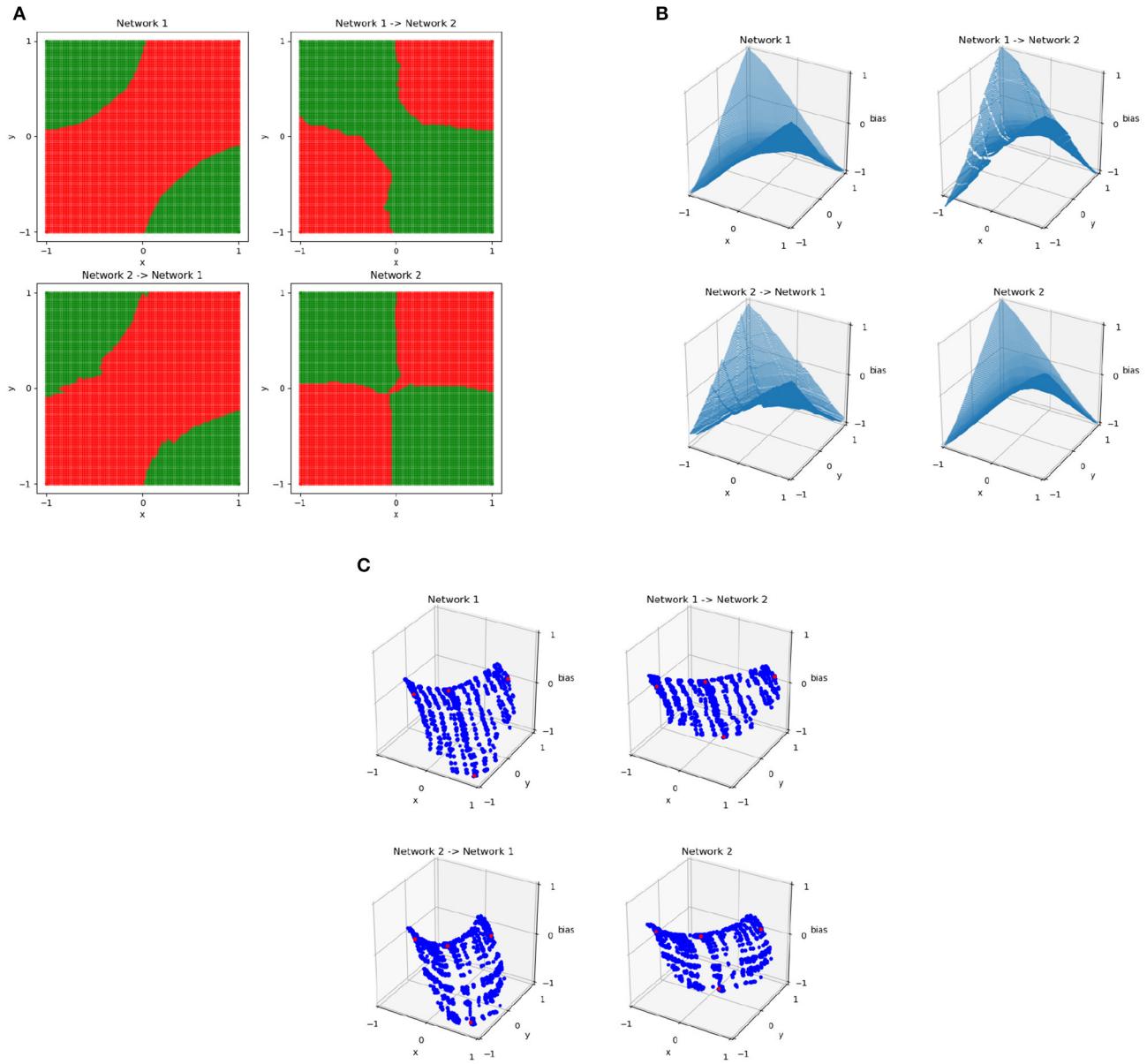


FIGURE 6
The decision boundaries **(A)**, wireframes **(B)**, and weights in the dual space **(C)** for two different XOR networks and the result of training an affine mapping from the linear regions of one network to the other. The original networks are the upper left and lower right in each subplot, with the resulting mapped networks are the top right and bottom left. In the dual space, the four points of XOR are in red and all other points sampled uniformly from the grid in the original input space are in blue.

the position in the plane and regions may no longer join at their boundaries. By applying this method to more complex networks where similarity is not as clear, we can determine potential overlap in network behavior.

2.5 Extension to image data

The idea of investigating and visualizing linear regions can be extended to higher dimensions, and specifically to image data, although visualizations are no longer as simple. We consider the MNIST dataset of handwritten digits which contains 60,000 training samples and 10,000 test samples (LeCun et al., 1998). MNIST was chosen as an image classification dataset due to its relative simplicity. We used PyTorch (Paszke et al., 2019) to train four networks on the MNIST dataset. These networks are

- A fully connected feedforward network with a single hidden layer consisting of 128 ReLU nodes. This network achieves an accuracy of 96.03%. The training process used cross-entropy loss and PyTorch's SGD function with parameters of 0.01

update rate, 0.5 momentum, 0.01 weight decay, and a batch size of 64 over 30 epochs.

- Two simple convolutional networks with equivalent architectures but different initializations. A convolutional layer with 10 filters and a kernel size of 5 is applied to the input, followed by a max pool. The results of that have a convolutional layer with 20 filters and a kernel size of 5 applied, again followed by a max pool. The 320 resulting outputs are used as inputs to a fully connected layer with 50 outputs, which is followed by a linear layer from those 50 nodes to the 10 outputs. The network achieves an accuracy of 98.07% (labeled Conv1) and 98.04% (labeled Conv2). The training process for both used cross-entropy loss and PyTorch's SGD function with parameters of 0.01 update rate, 0.5 momentum, 0.01 weight decay, and a batch size of 64 over 30 epochs.
- A network with the Inception-v3 architecture as implemented in Torchvision's models subpackage trained from scratch (Marcel and Rodriguez, 2010; Szegedy et al., 2016). This network achieves an accuracy of 99.08%. The first layer of the network was modified to expect images with only one channel and images were upsampled, using bilinear interpolation, to the expected size of 224 × 224 pixels. The training process used cross-entropy loss and PyTorch's SGD function with parameters of 0.1 update rate, 0.9 momentum, 1e-4 weight decay, and a batch size of 50 over 22 epochs.
- A network with the ResNet-152 architecture as implemented in Torchvision's models subpackage trained from scratch (Marcel and Rodriguez, 2010; Lin and Jegelka, 2018). This network achieves an accuracy of 98.92%. The first layer of the network was modified to expect images with only one channel. The training process used cross-entropy loss and PyTorch's SGD function with parameters of 0.1 update rate, 0.9 momentum, 1e-4 weight decay, and a batch size of 50 with 60 epochs.

For a given input image and output, each network determines a polytope within the input space which contains the image. By considering a single output, the gradient at the input image can be displayed in the same format as the input image. The collection of 50 different gradient "images," computed by considering each of the five neural networks and each of the 10 output nodes for an example "4" image, is visualized in Figure 7. Based on the appearance of the images, the dense network appears to have relatively little complexity, so it is classifying based on its "ideal" shape of each output, corresponding to what a linear classifier may do. This suggests a possibility for a reduction of number of linear regions used, as presented in Section 2.4.1 and discussed in Secton 3.2. The other networks have more complexity, tend to focus more sharply on the relevant information being passed in, and classify based on that input. ResNet has behavior that is not human-interpretable and appears somewhat noisy. Based on these visuals, the only networks that appear visually similar are the two simple convolutional networks, suggesting possible difficulty in the mappings presented in Section 2.4.2 and discussed in Section 3.3. The visualization of these linear regions is similar to the idea of saliency mappings, although many modern forms of saliency mapping are more sophisticated than simply visualizing the gradients at an input image, as this is doing (Simonyan et al., 2013).

3 Results

Constructing animations and visualizations of how network structure changes throughout the training process for different problems and architectures on two-dimensional problems can aid in understanding how the training process creates some of the properties investigated and provide inspiration for behavior to investigate in more detail. Although two-dimensional problems are useful for building intuition of network behavior, they do not necessarily include all of the behavior that most modern neural networks include. As such, using intuitions gained on those networks, even on a simple dataset such as MNIST, is necessary for confirming that networks can be reduced in complexity or exhibit quantitatively similar behavior despite differences in architecture.

3.1 Polytope evolution through training

The polytope structures discussed in Section 2.3 and their associated linear mappings change as the network trains, as has been studied previously (Hanin and Sellke, 2017; Hanin and Rolnick, 2019a; Zhang and Wu, 2019). However, these studies focus on MNIST and the usage of summary statistics to analyze behavior beyond the visuals in high-dimensional space. We focus on the visualization for two-dimensional input problems here, so that we can fully visualize the polytope structure and identify patterns of behavior across the entirety of the input. We continue with the problem of classifying a circle versus a surrounding annulus and additionally consider a more complex problem that is a combination of the XOR problem and the circle versus annulus problem, both illustrated in Figure 8.

There exist many simple solutions to the single circle versus single annulus problem, but neural networks do not intrinsically take advantage of the rotational symmetry of this problem to express these solutions. As has been demonstrated previously (Hanin and Sellke, 2017; Raghu et al., 2017), any network that solves this problem requires a minimum of three hidden nodes in at least one of its hidden layers. A node in any layer creates a non-linearity along a line in the input space of that layer, but when mapped back to the original input space that line becomes a trajectory that "breaks" by changing direction whenever it encounters a line created by the activation boundary of a node in a previous layer. A network with a maximum width of two is unable to solve this problem as it is unable to create a closed region in the input space. To see this, note that each layer with at most two nodes can only partition space into four regions (both on, one on, the other on, and both off), one of which (both off) will be constant. Due to this, any such network cannot form a closed region in space and will instead have each of its polytopes extend to infinity.

To illustrate how these polytopes and decision boundaries change as the neural network trains, we have two examples to compare as networks increase in complexity. One is the simplest possible network with three nodes in a single hidden layer, and the

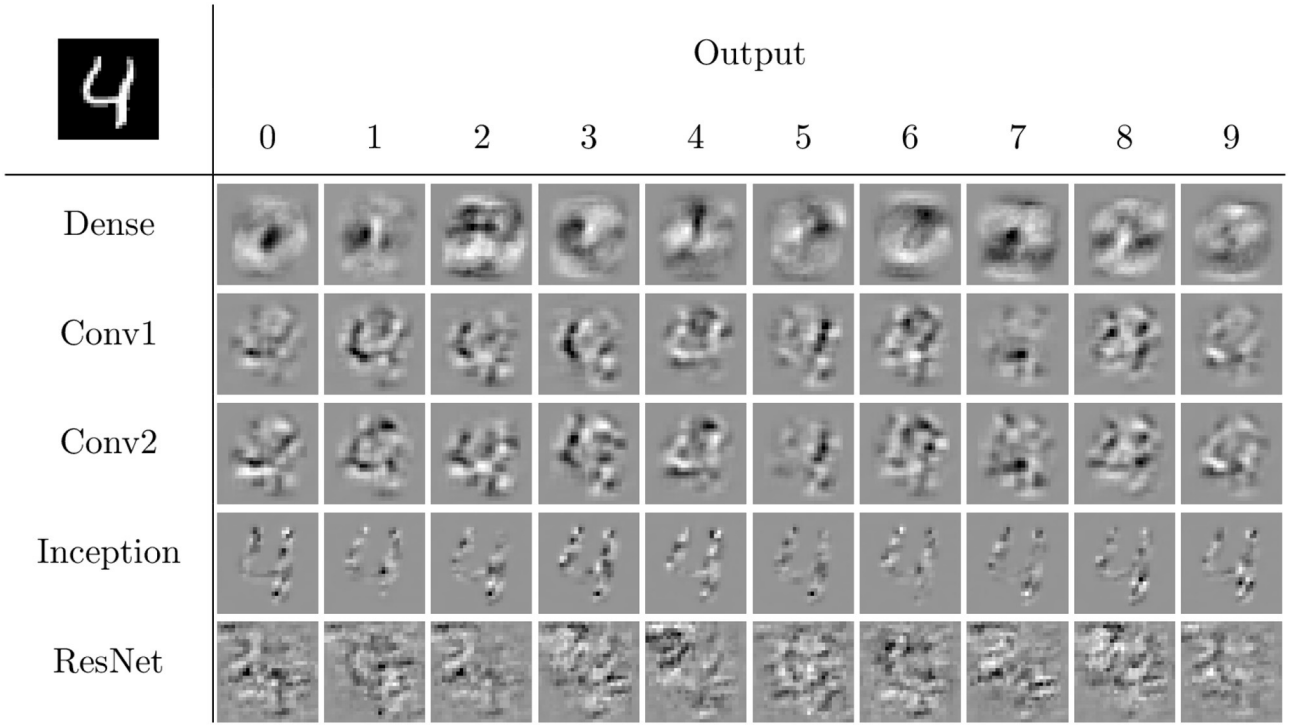


FIGURE 7
The linear regions for each output of the four networks for the input in the top left corner. These visualizations are similar to simple forms of saliency mapping (Simonyan et al., 2013). White pixels correspond to positive values, black pixels to negative values, and gray pixels to zero.

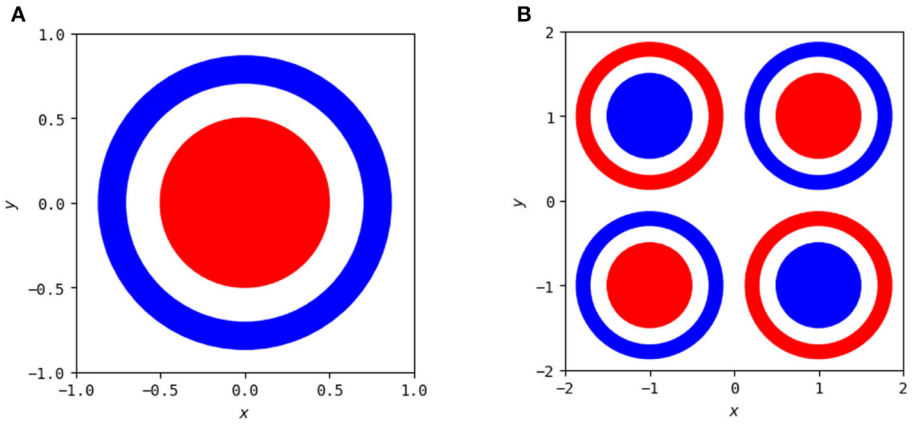


FIGURE 8
Two classification problems used to show animations of polytope structures during the training process. **(A)** The goal is to classify points in the red annulus as being a separate class as those in the blue annulus. **(B)** A combination of the left problem with the XOR problem to demonstrate more sophisticated network behavior.

other is a far more complex network with three hidden layers each containing eight nodes. Still images of the polytope development throughout the training process for the simple network are shown in Figure 9 and the end result of the more complex network is shown in Figure 10. Full videos of the evolution of their polytope structure throughout the training process are available at https://www.youtube.com/watch?v=lpXQI-UJIZM and https://www.youtube.com/watch?v=rANyD9t-X-c, respectively.

As shown in the training animations, the increased complexity of the network in Figure 10 allows it to manipulate its non-linearities to create a closed region in fewer epochs than that of the simple network shown in Figure 9. This comes from the fact that the structure of the simple network forms a subnetwork of the more complex network. The more complex networks initialization is more likely to have activation boundaries in beneficial places for finding good solutions, as believed to occur with the lottery

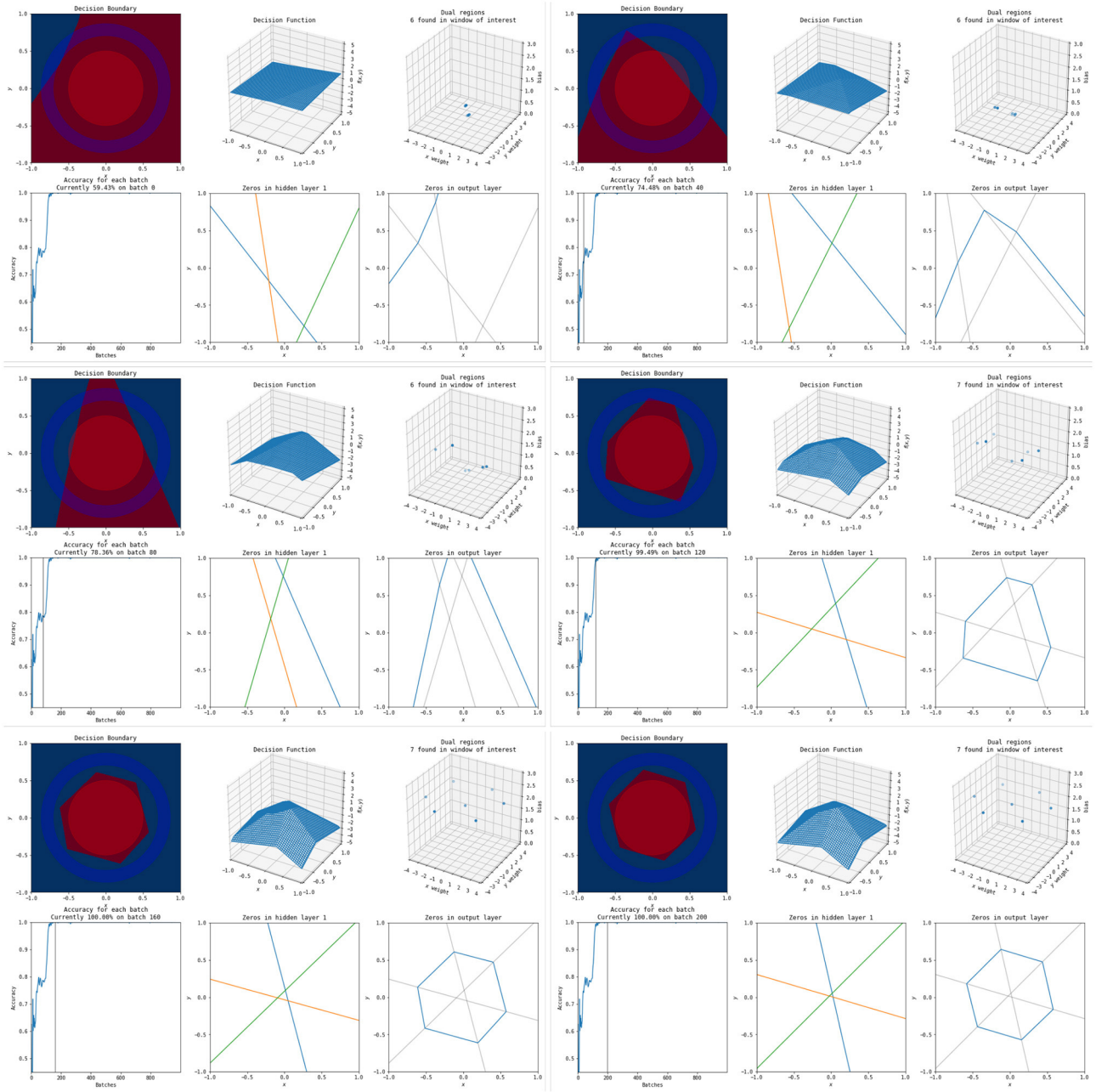


FIGURE 9
A selection of images throughout the training process of the simplest possible network (three hidden nodes in a single layer) on this problem. The **top left** set of six image shows the network at initialization, and each set of six images following that increases by 40 training epochs. Within each set of six images, the **top left** shows the decision surface, the **top center** shows the network output, the **top right** shows the weights in the dual, the **bottom left** shows the training accuracies, the **bottom middle** shows the activation boundaries of the hidden nodes, and the **bottom right** shows the decision boundary overlaid with the previous layers activation boundaries. An animation of this process is available at https://www.youtube.com/watch?v=lpXQI-UJIZM.

hypothesis (Frankle and Carbin, 2018). Additionally, the simple network does not find a solution every time using its training method, frequently finding locally optimal solutions that do not form closed regions and achieving only 50% accuracy.

An example of the polytopes constructed by a more complex network on combination of the XOR and circle vs. annulus problem is in Figure 11. A video of the training process is shown at https://www.youtube.com/watch?v=T_uoGBUOgUY. This network demonstrates a situation where the network requires a more complex structure to successfully classify. Additionally, the points in the dual do not cluster as they do in the original circle versus annulus problem.

It has previously been shown by Raghu et al. (2017) that earlier layers are more important than later layers for the quality of a network and certain visualizations of this were included in their study. These animations provide additional intuitive examples of

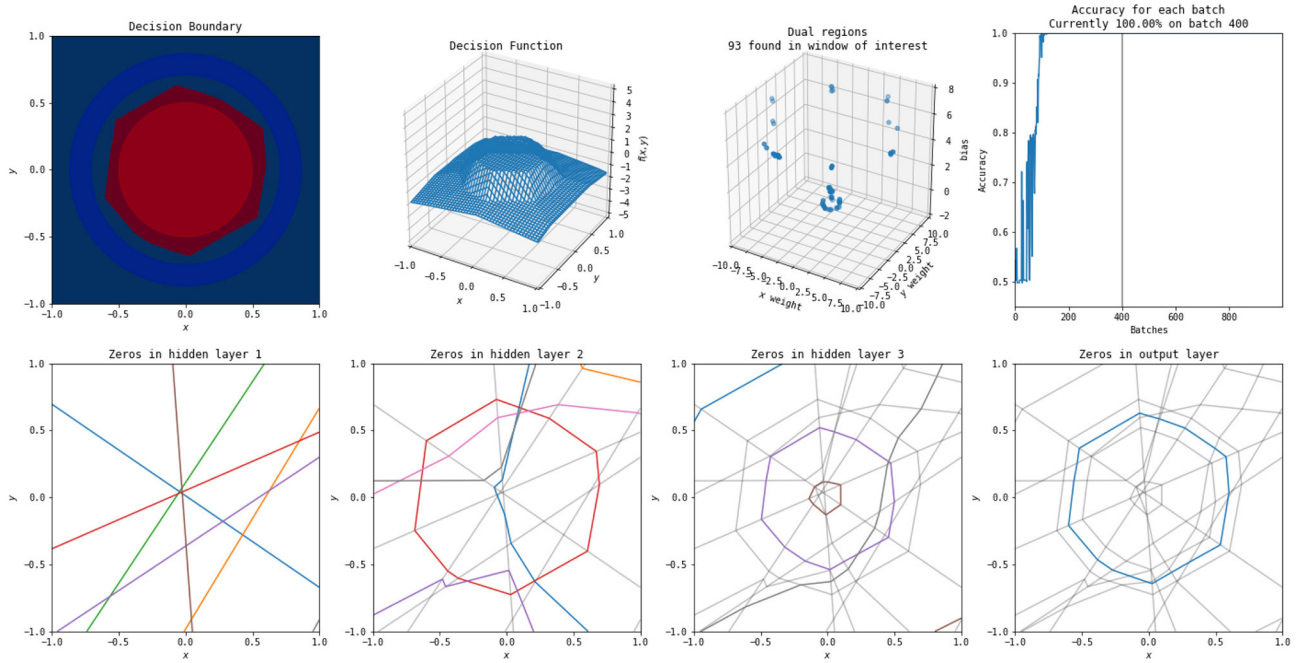


FIGURE 10
A solution to this problem found by a more complex network (three hidden layers, each with eight nodes). An animation of the training process of this network is available at https://www.youtube.com/watch?v=rANyD9t-X-c.

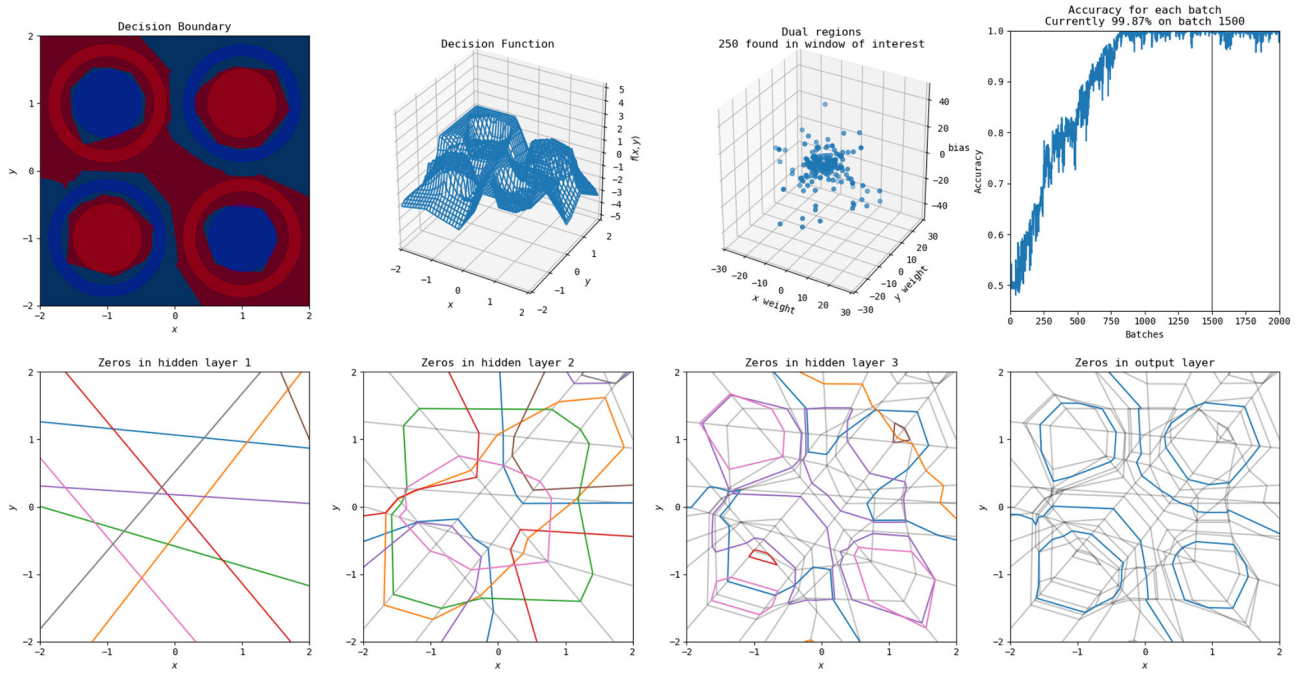


FIGURE 11
A solution to this problem found by a more complex network (three hidden layers, each with eight nodes). An animation of the training process of this network is available at https://www.youtube.com/watch?v=T_uoGBUOgUY.

this—the structures constructed by the early layers are passed on, and many of the deeper layers provide only slight modifications to the structures apparent in the first layers. Even in the more complex problem combining XOR and the circle versus annulus problem, the second layer forms the bulk of the structure for classification, with the third and final layers only refining it.

Using simple visualizations of this sort provide a holistic view of certain network behaviors. Intuitive understanding of existing hypotheses and theorems and inspiration for further investigation can be gained. In this case, the visualizations and animations demonstrate results such as the lottery hypothesis (Frankle and Carbin, 2018) and the influence of earlier layers (Raghu et al., 2017) visually to potentially enhance understanding of the phenomena. It also allows for identification of further areas of interest—in this case, the idea that networks trained on the same problem can exhibit significant similarity, and that there is possible pruning that can be done on networks that are more complex than necessary for representing a solution despite their increased complexity allowing a good solution to be found more effectively. Additionally, they show that the more complex network on the circle problem can have hidden layers removed, despite that method being relatively rare in studies of network pruning (Blalock et al., 2020).

3.2 Clustering sampled local linear functions

Using the methods discussed in Secton 2.4.1 and the networks described in Section 2.5, we can investigate the effect of clustering the linear weights for MNIST trained neural networks. Accuracies for this process with different numbers of cluster centers are shown in Table 1.

Networks with less complex architectures capture the near-linear behavior of the MNIST dataset well. The dense, single-hidden-layer network, in particular, is able to recover a solution close to linear classifiers in the single cluster case, suggesting that the network replicates linear behavior well. This matches previous study that shows that single-hidden-layer wide networks tend to behave in highly linear ways.

Inception and ResNet, however, have near random performance in the single cluster case. This suggests that the networks are transforming the data in such a way that the transformation cannot be approximated linearly, which matches the complexity of architectural structure that those networks have. The basic convolutional networks perform poorly, but significantly better than random, suggesting that their transformation is non-linear but has some linear properties.

As the number of clusters increases, inception quickly recovers a high degree of accuracy, achieving better accuracy than the original dense network with as few as 10 clusters. This suggests that inception identifies a piecewise linear mapping from the input space that can be reasonably well approximated by as few as 10 regions. This means that inception has a high degree of redundancy when trained on MNIST, and could likely be pruned significantly while maintaining original accuracy. This also could be a sign that inception generalizes well on this dataset; it does not maintain a high level of complexity and uses relatively simple methods to classify the data.

This behavior is not matched by the basic convolutional networks or ResNet, with their performance remaining poor. The basic convolutional networks are able to recover for better accuracy than the original dense network with 10,000 clusters (1/6 of the number of training samples), but ResNet maintains poor accuracy throughout. This means that the polytope structure of ResNet cannot be simplified easily in this manner—either a more sophisticated method is necessary to identify ways of simplifying the polytope structure, or the network has a high degree of complexity that can not be reduced.

3.3 Affine maps between sampled local linear functions

Using the methods discussed in Secton 2.4.2 and the networks described in Section 2.5, we can investigate the effect of transforming between the linear weights for different MNIST trained neural networks. Examples of this are illustrated for five input samples for $\mathbf{W}_{0,dense}$ and $\mathbf{W}_{0,conv}$ in Figure 12. Qualitatively, the mapped linear regions are similar, but not equivalent, to the target. One point to note is that all of the linear regions for the dense network appear qualitatively similar, with their shape matching that of a zero. The convolutional network does not follow this pattern, having different patterns for each of the input samples, and the transformed dense regions are able to replicate those patterns despite their visual similarities.

Table 2 shows the results of the affine mapping trained on the training set and evaluated on the testing set for the five networks trained on MNIST. No mapped network performs better than its original accuracy or the original accuracy of the network it is transformed to match. Despite this decrease in accuracy from the original networks, a high level of accuracy is maintained for many of the mappings. This is not necessarily unexpected, as all five networks are attempting to approximate the same function because they are trained to solve the same problem using the same loss function. However, accuracy does not necessarily tell the whole story. A high level of performance is preserved, but it may be the case that slight variations in accuracy represent significant, qualitatively meaningful differences in what the networks do. Even so, these results demonstrate that there is ostensibly an interesting relationship between these different networks and their similar behaviors.

The dense network and the basic convolutional networks are able to transform to each other reasonably well. Transforming to and from the dense network achieves accuracy near its original, and the basic convolutional networks with equivalent architectures and training methods are able to nearly replicate their original accuracies with transforming between each other. This suggests overlap in how the networks transform the data to achieve their networks, with the convolutional networks achieving additional complexity that allows them to increase their accuracy.

Interestingly, the inception network replicates the other networks poorly, with a significant drop in accuracy mapping to the dense network, accuracy near the dense networks transformation to the basic convolutional networks for them, and accuracy below the original dense network when transforming to ResNet. This matches the clustering results, where the network did not have a good single linear representation, explaining its lack of success mapping to the dense network, but recovers accuracy quickly with 10 clusters, suggesting a

TABLE 1 The accuracies of networks on the MNIST dataset after applying *k*-means clustering to their collection of local linear maps.

# Clusters	Dense	Conv1	Conv2	Inception	ResNet
Original	9603	9807	9804	9908	9892
1	8679	6766	6366	974	1432
10	9231	8639	8672	9660	8166
100	9434	9382	9421	9689	8458
1000	9508	9586	9603	9695	8982
10000	9554	9696	9673	9752	9381

Values reported are the number of correctly labeled test set samples out of 10,000. Note that the number of clusters for a given network is technically 10 times larger than stated in the table—each of linear mappings corresponding to a digit is clustered separately.

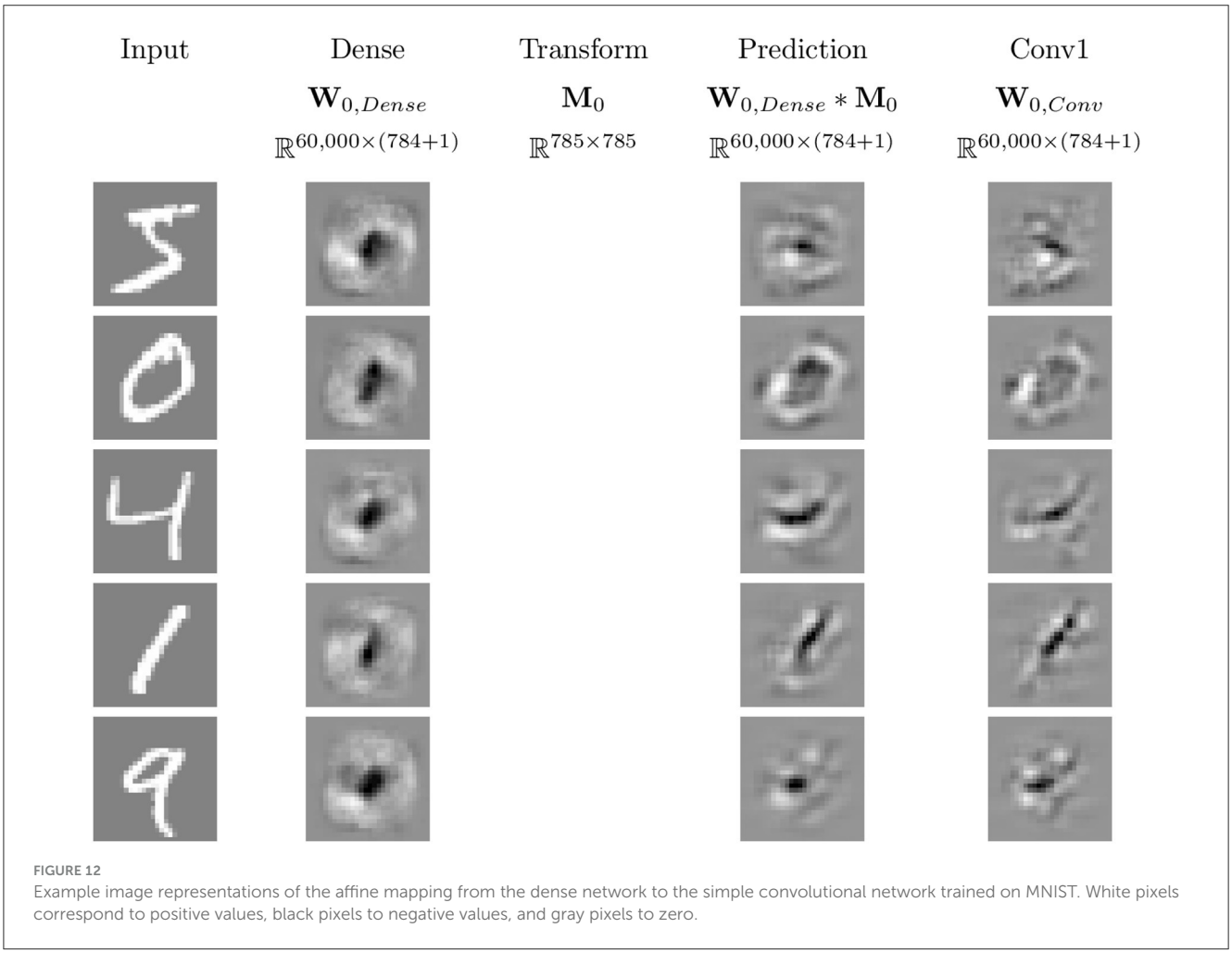


FIGURE 12
Example image representations of the affine mapping from the dense network to the simple convolutional network trained on MNIST. White pixels correspond to positive values, black pixels to negative values, and gray pixels to zero.

possible simple transformation that does not easily replicate the success of the other networks despite its success by itself.

ResNet, however, is able to replicate everything except the dense network well. It comes close to the original accuracies of the basic convolutional networks when mapping to them, as well as both its and inception's accuracies when mapping to inception. This suggests that the linear regions it uses contain the information that the other networks use for classification. Together with the results with clustering, this suggests that ResNet maintains a complex transformation from the input space.

The lack of symmetry between inception and ResNet is interesting. ResNet is able to approximate inception well, but inception is not able to approximate ResNet well. This means that there is a qualitative difference between the methods these networks are using for classification, despite their near equivalent original accuracies. This suggests, due to both of them achieving success, that ResNet identifies information that may not generalize, as inception is able to perform equivalently using a representation that appears simpler. Identifying both the overlap and differences in the methods these networks use for classification provides a way for identifying possible improvements for both networks.

TABLE 2 Number of correct labels on the test set (out of 10,000) after applying affine mappings to the linear mappings of MNIST trained neural networks.

		To				
		Dense	Conv1	Conv2	Inception	ResNet
From	Dense	9603	9536	9519	9290	9068
	Conv1	9567	9807	9776	9662	9588
	Conv2	9562	9786	9804	9644	9579
	Inception	8868	9488	9511	9908	9536
	ResNet	9320	9738	9739	9838	9892

Diagonal elements are the original accuracies of the networks.

By identifying the similarity between the linear regions of networks trained on the same or similar datasets, it is possible to gain a deeper understanding of the networks' behavior. Dissimilar representations suggest the exploitation of different information across the networks, meaning that the network behaviors can potentially be combined to improve effectiveness. The ability for one network to effectively recreate the representations of another suggests that the first network exploits the information the other contains meaning that differences in accuracy can come down to more effective exploitation, rather than identifying qualitative differences.

4 Discussion

Identifying patterns in simple neural networks trained on low-dimensional toy problems can provide meaningful insight and intuition for patterns that are replicated in modern neural networks trained on high-dimensional complex problems. These patterns can assist in gaining deeper insight into the behavior and in suggesting methodologies that can be applied to those complex networks. We have extended the work of Raghu et al. (2017) in visualizing the polytope structure of neural networks with two inputs by constructing animations of the evolution of the polytope structure. These animations demonstrate how early layers have significant influence over the structure of subsequent layers and how the polytope structures form through training.

Additionally, we have shown experimentally that even complex neural networks such as inception can have the complexity of their underlying polytope partitioning of the input space highly reduced. The linear regions of all networks considered, except ResNet, can be clustered to as few as 10 cluster centers for networks trained on MNIST while preserving much of their accuracy.

We have also shown experimentally that the linear regions of different networks are similar under an affine mapping. Applying such an affine mapping preserves a high level of accuracy in the resulting classifier, suggesting that many of the considered networks are solving problems in qualitatively similar ways. By comparing accuracies of mapped networks, we are able to determine where networks may have qualitatively dissimilar behavior in a way that suggests poor generalization or information that can be exploited to improve network behavior.

We provide support for the tantalizing idea that different networks converge to similar solutions that have a great deal more simplicity than would be suggested by their complex architectures. Further investigations of this area could allow for identification of patterns across disparate networks that allow for a more refined understanding of both training networks and modifying them to be effective in full usage. We would like to continue to explore the extent to which that idea is correct for modern neural networks through extensions to more complex datasets and network pruning methodologies.

Although MNIST provides high-dimensional image data, the dataset itself is relatively simple. Extending the similarity and clustering methods to more complex datasets would provide deeper insight into how complexity of dataset can influence the similarity and complexity of neural networks trained on them. Each of the networks trained here, despite their distinct architectures, used similar training methodologies. Extending the study by Zhang and Wu (2019) to investigate how different training methodologies and regularization techniques impact the similarity of network behavior would allow for an understanding of how those methods impact polytope structures.

Additionally, network pruning research provides a natural field where the ability to compare the similarity of two networks, linear regions would be useful. Identifying the degree to which a network can be simplified without impacting its ability to successfully classify can be difficult using only accuracy as a metric (Blalock et al., 2020). Comparing the behavior of those networks directly while being able to treat the interior of the network as a black box provides a promising technique for identifying success.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found in the article/Supplementary material. Code accessible at https://github.com/bsattelb/local-linearity-of-relu-neural-networks/tree/master.

Author contributions

BS: Conceptualization, Methodology, Software, Writing—original draft, Writing—review & editing. RC: Conceptualization, Writing—review & editing. MK: Conceptualization, Writing—review & editing. CP: Conceptualization, Writing—review & editing. RB: Conceptualization, Writing—original draft, Writing—review & editing.

Funding

This study is partially supported by the DARPA Geometries of Learning Program under Award No. HR00112290074.

Acknowledgments

A preprint version of this article is available on arXiv (Sattelberg et al., 2020).

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.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2023.1255192/full#supplementary-material

VIDEO 1.AVI
An animation showing the training process of a simple neural network training to classify a points on a circle vs. points on a surrounding annulus. Polytope structure, local linear weights, decision boundary, and the output of the network are shown through the training process.

VIDEO 2.AVI
An animation showing the training process of a relatively complex neural network training to classify a points on a circle vs. points on a surrounding annulus. Polytope structure, local linear weights, decision boundary, and the output of the network are shown through the training process.

VIDEO 3.AVI
An animation showing the training process of a neural network training to classify points on four circles with classes matching the XOR problem and annuli surrounding each circle with opposite classification. Polytope structure, local linear weights, decision boundary, and the output of the network are shown through the training process.

References

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. (2020). "What is the state of neural network pruning?" in *Proceedings of Machine Learning and Systems 2 (MLSys 2020)* 129–146.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathem. Control Sign. Syst.* 2, 303–314. doi: 10.1007/BF02551274

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). "ArcFace: additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 4690–4699. doi: 10.1109/CVPR.2019.00482

Frankle, J., and Carbin, M. (2018). "The lottery ticket hypothesis: finding sparse, trainable neural networks," in *International Conference on Learning Representations*.

Hanin, B., and Rolnick, D. (2019a). Complexity of linear regions in deep networks. *arXiv preprint arXiv:1901.09021*.

Hanin, B., and Rolnick, D. (2019b). "Deep ReLU networks have surprisingly few activation patterns," in *Advances in Neural Information Processing Systems* 361–370.

Hanin, B., and Sellke, M. (2017). Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*.

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* 770–778. doi: 10.1109/CVPR.2016.90

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Netw.* 4, 251–257. doi: 10.1016/0893-6080(91)90009-T

LeCun, Y., Cortes, C., and Burges, C. J. (1998). *The MNIST database of handwritten digits*. Available online at: http://yann.lecun.com/exdb/mnist/ (accessed October 15, 2019).

Lin, H., and Jegelka, S. (2018). "ResNet with one-neuron hidden layers is a universal approximator," in *Advances in Neural Information Processing Systems* 6169–6178.

Liu, Y., Cole, C., Peterson, C., and Kirby, M. (2023). "Relu neural networks, polyhedral decompositions, and persistent homology," in *the ICML 2023 Workshop on Topology, Algebra, and Geometry in Machine Learning*.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). "The expressive power of neural networks: A view from the width," in *Advances in Neural Information Processing Systems* 6231–6239.

Marcel, S., and Rodriguez, Y. (2010). "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM international conference on Multimedia* 1485–1488. doi: 10.1145/1873951.1874254

McNeely-White, D., Beveridge, J., and Draper, B. (2019). Inception and ResNet features are (almost) equivalent. *Cogn. Syst. Res.* 59, 312–318. doi: 10.1016/j.cogsys.2019.10.004

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems* 2924–2932.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*.

Pascanu, R., Montufar, G., and Bengio, Y. (2013). On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (Red Hook, NY: Curran Associates, Inc.), 8024–8035.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). "On the expressive power of deep neural networks," in *international Conference on Machine Learning, pages* 2847–2854.

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

Sattelberg, B., Cavalieri, R., Kirby, M., Peterson, C., and Beveridge, R. (2020). Locally linear attributes of relu neural networks. *arXiv preprint arXiv:2012.01940*.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* 2818–2826. doi: 10.1109/CVPR.2016.308

Zhang, L., Naitzat, G., and Lim, L.-H. (2018). Tropical geometry of deep neural networks. *arXiv preprint arXiv:1805.07091*.

Zhang, X., and Wu, D. (2019). "Empirical studies on the properties of linear regions in deep neural networks," in *International Conference on Learning Representations*.

frontiers | Frontiers in Artificial Intelligence

TYPE Original Research
PUBLISHED 24 November 2023
DOI 10.3389/frai.2023.1274830

OPEN ACCESS

EDITED BY
Pavan Turaga,
Arizona State University, United States

REVIEWED BY
Ankita Shukla,
Arizona State University, United States
Suhas Lohit,
Mitsubishi Electric Research Laboratories (MERL), United States
Gautam Dasarathy,
Arizona State University, United States

*CORRESPONDENCE
Michael Kirby
✉ michael.kirby@colostate.edu

RECEIVED 08 August 2023
ACCEPTED 03 November 2023
PUBLISHED 24 November 2023

CITATION
Karimov K, Kirby M and Peterson C (2023) An algorithm for computing Schubert varieties of best fit with applications.
Front. Artif. Intell. 6:1274830.
doi: 10.3389/frai.2023.1274830

An algorithm for computing Schubert varieties of best fit with applications

Karim Karimov, Michael Kirby* and Chris Peterson

Department of Mathematics, College of Natural Sciences, Colorado State University, Fort Collins, CO, United States

We propose the geometric framework of the Schubert variety as a tool for representing a collection of subspaces of a fixed vector space. Specifically, given a collection of l-dimensional subspaces $V_1, \ldots, V_r$ of $\mathbb{R}^n$, represented as the column spaces of matrices $X_1, \ldots, X_r$, we seek to determine a *representative* matrix $K \in \mathbb{R}^{n \times k}$ such that each subspace V_i intersects (or comes close to intersecting) the span of the columns of K in at least c dimensions. We formulate a non-convex optimization problem to determine such a K along with associated sets of vectors $\{a_i\}$ and $\{b_i\}$ used to express linear combinations of the columns of the X_i that are close to linear combinations of the columns of K. Further, we present a mechanism for integrating this representation into an artificial neural network architecture as a computational unit (which we refer to as an abstract node). The representative matrix K can be learned *in situ*, or sequentially, as part of a learning problem. Additionally, the matrix K can be employed as a change of coordinates in the learning problem. The set of all l-dimensional subspaces of $\mathbb{R}^n$ that intersects the span of the columns of K in at least c dimensions is an example of a Schubert subvariety of the Grassmannian $GR(l, n)$. When it is not possible to find a Schubert variety passing through a collection of points on $GR(l, n)$, the goal of the non-convex optimization problem is to find the Schubert variety of best fit, i.e., the Schubert variety that comes as close as possible to the points. This may be viewed as an analog of finding a subspace of best fit to data in a vector space. The approach we take is well-suited to the modeling of collections of sets of data either as a stand-alone Schubert variety of best fit (SVBF), or in the processing workflow of a deep neural network. We present applications to some classification problems on sets of data to illustrate the behavior of the method.

KEYWORDS

Schubert variety of best fit, manifold approximation, subspace classification, geometry of learning, neural network, abstract node, GPU parallel computing

1 Introduction

A variety of powerful tools have been developed in Machine Learning and Artificial Intelligence and these have led to remarkable applications. A damper on the success of these tools is the fact that the resulting models are frequently difficult to explain and the predictions may not be trustworthy in high stakes scenarios, e.g., those related to medical diagnoses, battlefield scenarios, or intelligence gathering. One reason the success of ML/AI tools is challenging to explain, or trust, is that these models were designed, first and foremost, to make accurate predictions; attempts to interpret or explain the effectiveness of the models being only an afterthought.

In this paper, we propose a methodology based on an interpretable mathematical framework, i.e., the geometric setting of the Schubert Variety, as the starting point, and explore variations on the theme to determine optimal architectures for predictive modeling. The goal of this research is to begin with a mathematically motivated explainable approach, and then optimize its performance through numerical algorithms. Optimistically, this approach will lead to results of comparable accuracy of the traditional ML/AI toolkit, however, with the advantage of explainability and trustworthiness. To this end, we propose an approach for characterizing sets of linear spaces, i.e., fitting/approximating subspaces with one, or more representative spaces. Additionally, we demonstrate how the mathematical framework and resulting algorithms can be integrated into current tools including deep feed forward neural networks.

The initial development of ML/AI, focused more on thinking machines than interpretability. Human intelligence, and the associated architecture of the human brain, have been a driving force in biomimetic approaches. For example, the McCullough-Pitts node (McCulloch and Pitts, 1943) and its associated weights were proposed as a mathematical model of the cell and its associated neurons, respectively. The first transfer function at a computational node was a simple step function replicating the firing or quiescence of a neuron. Impressively, arrays of such networks were shown to be able to serve as models of associative memory, and to even recall patterns which were partially occluded (Hertz et al., 1991). Hebbian learning (Hebb, 1949) was proposed as a model for memory and convincingly analyzed as a dynamical system where the patterns were stored as fixed points (Hertz et al., 1991), an early appearance of the application of mathematical analysis for explainability of artificial neural networks.

Geometric ideas emerged with Rosenblatt's simple perceptron (Rosenblatt et al., 2002), still loosely based on neurons firing, where the dot product operation between a pattern and a weight vector gave rise to classification via the interpretation of the models as splitting a space into two half-spaces. The multilayer perceptron extended these ideas in natural ways, however, at the expense of geometric interpretability. Nonlinear data reduction was made possible by autoencoder neural networks (Kramer, 1991; Oja, 1992). The encoder-decoder architecture has been widely exploited by Variational Autoencoders (Kingma and Welling, 2019), Centroid-Encoders (Ghosh and Kirby, 2022), and Transformers (Vaswani et al., 2017). These developments, while providing powerful tools, widely lack mathematical underpinnings that provide insight into their utility.

In this paper, we illustrate how one can use mathematical theory and geometric frameworks as a design philosophy in the construction of novel neural network architectures. This general idea can be found in previous work, e.g., geometric, or topological nodes such as circular (Kirby and Miranda, 1996), or spherical computational units (Hundley et al., 1995). Whitney's theorem has also been invoked to provide a basis to understand the power of autoencoders from a geometric perspective (Broomhead and Kirby, 2000) and to provide insights into novel architectures (Broomhead and Kirby, 2001) and dimension estimation (Anderle et al., 2002; Kvinge et al., 2018a,b). These ideas are central to the computation of homeomorphisms between *data sets* residing in spaces of differing dimensions. Using these ideas as motivation we can envision extending the concept of an abstract node more generally to algebraic varieties related to Generalized Principal Component Analysis (Vidal et al., 2005), Klein bottles, Grassmannians, and Schubert Varieties (see Figure 1).

In this paper, we focus our attention on the mathematical framework of the Schubert Variety described in what follows. We are motivated by the idea that a Schubert variety is to a Flag or Grassmann manifold what a subspace is to a vector space. Flag manifolds, and their special case the Grassmann manifolds, are examples of homogeneous manifolds particularly relevant to and amenable to subspace methods in Data Science. They have been observed to be particularly robust to data collected under variations in pattern state (and indeed exploit structure in such data sets). For example, digital images of an object, collected under variations in illumination, are known to sweep out a convex cone. If the object is Lambertian then this cone has been shown to lie close to a low dimensional linear space which can in turn be represented by a point on a Grassmannian (Beveridge et al., 2008). The flag manifold comes equipped with geometric features capable of representing sets of data where the number of points is larger than the number of dimensions in the ambient space (Ma et al., 2021). We note that the flag mean proposed in Marrinan et al. (2014, 2015) and Mankovich et al. (2022), and its various extensions, are a special case of the work proposed here.

Briefly, in this paper we propose several optimization problems which are used to produce a geometric object, e.g., a Schubert variety of best fit (SVBF), that optimally represents a set of linear subspaces of a fixed vector space $\mathbb{R}^n$. In several applications, the set of linear spaces are obtained from sets of sets of data. The optimization problem is determined by a real valued objective function on a manifold (typically a Grassmann or Flag manifold) whose points parameterize a family of potential Schubert varieties of best fit. Further, we show how this framework can be viewed as a component of a machine learning architecture integrated, e.g., into the broader framework of feed forward neural networks.

This paper is organized as follows: Section 2 presents a brief overview of a class of manifolds built from matrix group actions. In Section 3, we describe Schubert varieties and how to define a Schubert variety of best fit to a collection of subspaces. Section 4 describes a specific implemented optimization problem for finding a Schubert variety of best fit and applies it to an illustrative example. Section 5 provides three algorithms and shows how to implement the ideas as an abstract node. Lastly, in Section 6, we provide concluding remarks summarizing the overall findings and contributions of our research.

2 Background

Consider the set, S, of $n \times n$ invertible matrices whose inverse is equal to its transpose, i.e., $S = \{A \in \mathbb{R}^{n\times n} \mid A^T A = I_n\}$. S contains the identity matrix, I_n, and is closed under the operations of matrix inversion and matrix multiplication. In other words, if A and B are in S then both A^{-1} and AB are in S. The set S together with the binary operation of matrix multiplication is known as the *orthogonal group* $O(n)$. Alternatively, $O(n)$ is the group of distance-preserving transformations of an n-dimensional Euclidean space that preserve a fixed point. A distinguished subgroup of $O(n)$ is

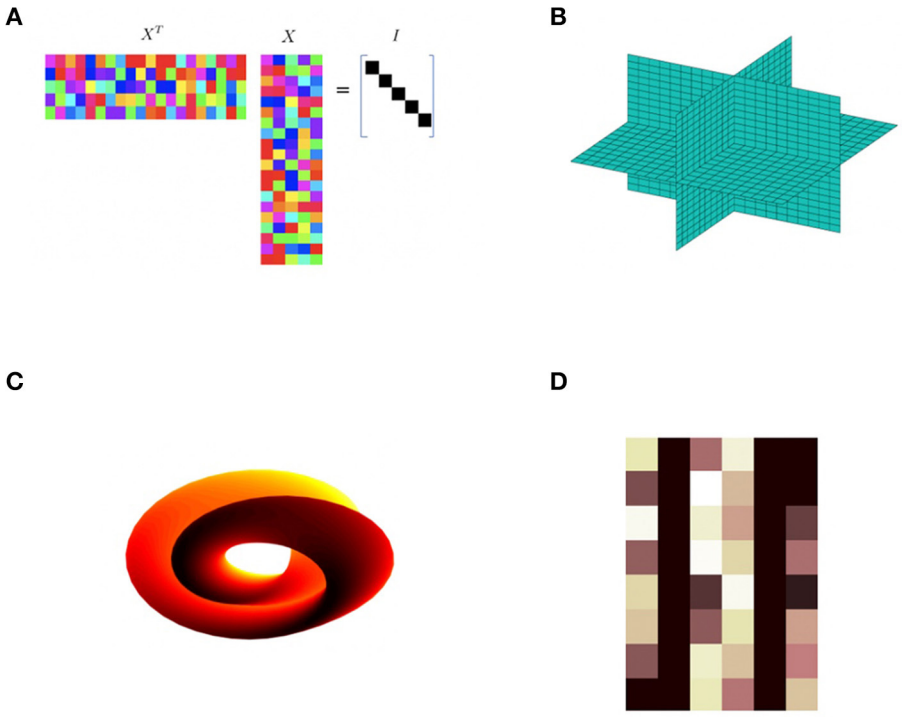


FIGURE 1
Various examples of abstract nodes. These *computational units* can be integrated into data fitting problems, e.g., multilayer neural networks, to extract topological or geometric structure. This paper focuses on the particular case of the Schubert variety constraint **(D)**. **(A)** This computational unit with matrix values X is required to satisfy $X^TX = 1$. **(B)** This unit requires the data to reside in a union of sub-spaces such as the *xy*, *xz*, and *yz* planes shown. **(C)** The Klein bottle is an example of a unit manifold constraint. **(D)** A Schubert variety constraint.

the special orthogonal group $SO(n)$ consisting of elements of S with determinant equal to one (orientation preserving transformations). From a geometric perspective, the orthogonal group $O(n)$ can be considered as a manifold whose points parameterize ordered orthonormal bases of $\mathbb{R}^n$. As a manifold, it consists of two connected components corresponding to the square orthogonal matrices with determinant $+1$ and those with determinant -1. Its dimension as a real manifold is $\binom{n}{2}$. Several interesting manifolds can be built, through a "quotient" operation, by considering the action of subgroups of $O(n)$ [resp. $SO(n)$] on $O(n)$ [resp. $SO(n)$] through multiplication. Some particularly relevant examples are the following:

- Grassmann manifolds $GR(l, n)$
- Oriented Grassmann manifolds $\widetilde{GR}(l, n)$
- Flag manifolds $FL(l_1, l_2, ..., l_m; n)$
- (Partially) oriented flag manifolds
- Steifel manifolds $ST(l, n)$

Grassmann manifold—The Grassmannian of l-dimensional subspaces of R^n is denoted by $GR(l, n)$. Points on $GR(l, n)$ correspond to l-dimensional subspaces of $\mathbb{R}^n$. It can be built as a coset space $O(n)/O(l) \times O(n-l)$ where $O(l) \times O(n-l)$ denotes $n \times n$ matrices consist of an $l \times l$ orthogonal block and an $n-l \times n-l$ orthogonal block. Through this identification, points on $GR(l, n)$ correspond to equivalences classes of $n \times n$ orthogonal matrices where two such matrices are identified if the span of their first l columns agree. We can also think of points on $GR(l, n)$ as corresponding to equivalences classes of $n \times l$ orthogonal matrices where two such matrices are identified if they have the same column space. $GR(l, n)$ can be considered as a homogeneous space and as a differentiable manifold. As a real manifold, the dimension of $GR(l, n)$ is $l(n-l) = dim(O(n)) - dim(O(l)) - dim(O(n-l))$.

Oriented Grassmann manifold—The oriented Grassmannian of all *oriented* l-dimensional subspaces of $\mathbb{R}^n$ is denoted $\widetilde{GR}(l, n)$. It can be built as a coset space $SO(n)/SO(l) \times SO(n-l)$. There is a natural 2:1 covering map from $\widetilde{GR}(l, n)$ to $GR(l, n)$. A special case is $\widetilde{GR}(1, n)$ whose cosets correspond to points on the $n-1$ dimensional sphere S^{n-1} in R^n. $GR(1, n)$ corresponds to the real projective space RP^{n-1}. RP^{n-1} can also be built from S^{n-1} by identifying antipodal points on the sphere. In the map from $\widetilde{GR}(1, n)$ to $GR(1, n)$, a pair of antipodal points on the sphere get mapped to a single point in projective space. As a real manifold, the dimension of $\widetilde{GR}(l, n)$ is the same as the dimension of $GR(l, n)$.

Flag manifold—$FL(l_1, l_2, ..., l_m; n)$ = collection of all flags of the form $V_1 \subset V_2 \subset \cdots \subset V_m \subset \mathbb{R}^n$ such that $\dim V_i = l_i$. The flag manifolds can be built by considering quotients of $O(n)$ by a direct product of smaller orthogonal groups. More precisely by the quotient of $O(n)$ by $O(l_1) \times O(l_2 - l_1) \times O(l_m - l_{m-1}) \times O(n - l_m)$. As a real manifold, the dimension of $FL(l_1, l_2, ..., l_m; n)$ is $dim(O(n)) - dim(O(l_1)) - dim(O(l_2 - l_1)) - dim(O(l_3 - l_2)) - \cdots - dim(O(l_m - l_{m-1})) - dim(O(n - l_m))$.

Oriented flag manifold—$FL^{\circ}(l_1, l_2, ..., l_m; n)$ = collection of all oriented flags of the form $V_1 \subset V_2 \subset \cdots \subset V_m \subset \mathbb{R}^n$ such that $\dim V_i = l_i$. The oriented flag manifolds can be built by considering quotients of $SO(n)$ by a direct product of smaller special orthogonal groups. There is a natural $2^m : 1$ covering map from $FL^{\circ}(l_1, l_2, ..., l_m; n)$ to $FL(l_1, l_2, ..., l_m; n)$. As a real manifold, the dimension of $FL^{\circ}(l_1, l_2, ..., l_m; n)$ is the same as the dimension of $FL(l_1, l_2, ..., l_m; n)$.

Partially oriented flag manifold—The partially oriented flag manifolds can be built by considering quotients of $SO(n)$ by a direct product of a mixture of smaller special orthogonal groups and smaller orthogonal groups [with the additional constraint that the direct product is a subgroup of $SO(n)$]. There are many different types of partially oriented flag manifolds.

Steifel manifold—Points on the Steifel manifold $ST(l, n)$ correspond to ordered orthonormal sets of l vectors in $\mathbb{R}^n$. Alternatively, points on $ST(l, n)$ correspond to elements in the set $\{A \in R^{n\times l} | A^T A = I_l\}$. The Steifel manifold $ST(l, n)$ can also be considered as the oriented flag manifold $FL^{\circ}(1, 2, 3, \ldots, l; n)$. The dimension of $ST(l, n)$ is $(n-1) + (n-2) + \cdots + (n-l)$. There is a natural $2^l : 1$ covering map from $ST(l, n)$ to $FL(1, 2, \ldots, l; n)$.

The homogeneous spaces listed above are all compact differentiable manifolds whose points parameterize flags of (oriented) subspaces of R^n with a common signature $l_1, \ldots, l_m$. Many problems of interest can be formulated in terms of optimizing some function on one or several of these or related parameter spaces. The formulation and solution is often driven by geometric considerations. Problems which have an efficient numerical solution are particularly appealing. Some additional parameter spaces of interest include Affine space, Euclidean space, Hyperbolic space, Anti-di Sitter space, and product spaces built out of any combination of these spaces, their oriented versions, or any the previously described homogeneous spaces from above.

3 Schubert varieties

A Schubert variety in a Grassmann or flag manifold is a certain kind of subvariety (typically with singularities) that can be defined by a collection of linear algebraic incidence constraints with respect to a fixed flag, F, drawn from some flag variety $FL(k_1, k_2, \ldots, k_m; n)$. If you vary the flag then you vary the Schubert variety. The Schubert variety can be viewed as a kind of moduli space while points on the flag variety can be seen as parameterizing a family of Schubert varieties of a particular type.

3.1 Definition of Schubert variety

We first consider an example of a kind of Schubert variety that will be referred to several times in this paper. If $W \in GR(k, n)$ then W is a rank k subspace of $\mathbb{R}^n$. Given a pair of non-negative integers (c, l), we can define a collection of points in $GR(l, n)$ by

$$\Omega_{c,k,l}(W) = \{V \in GR(l, n) \mid \dim(V \cap W) \geq c\}$$

In order for this set of points to be nonempty we will need that $c \leq l$ and that $c \leq k$. For each $W \in GR(k, n)$, $\Omega_{c,k,l}(W)$ is a subvariety of $GR(l, n)$. As a consequence, $GR(k, n)$ can be seen as parameterizing a family of such subvarieties of $GR(l, n)$. The subvariety $\Omega_{c,k,l}(W)$ is an example of a particular kind of Schubert variety on $GR(l, n)$. As mentioned in the previous paragraph, Schubert varieties are typically singular.

The example in the previous paragraph can be extended in several directions. The following is an example where W is drawn from a more general Flag manifold. Recall that $FL(k_1, k_2, ..., k_m; n)$ is the collection of all flags of the form $W_1 \subset W_2 \subset \cdots \subset W_m \subset \mathbb{R}^n$ such that $\dim W_i = k_i$. To emphasize that W is a flag of vector spaces, we will write W as $\mathbf{W}$. We call $FL(k_1, k_2, ..., k_m; n)$ an m step flag manifold. Points on this manifold are m step flags of signature $(k_1, k_2, \ldots, k_m)$. A Grassmann manifold is a one step flag manifold. For instance, $GR(k, n) = FL(k; n)$. A large collection of Schubert subvarieties of $GR(l, n)$ can be built as follows: Pick a point $\mathbf{W}$ on a flag manifold $FL(k_1, k_2, ..., k_m; n)$ and an m-tuple $\vec{c} = (c_1, \ldots, c_m)$ (thus $\mathbf{W}$ corresponds to a specific flag $W_1 \subset W_2 \subset \cdots \subset W_m$ where $\dim W_i = k_i$). Let $\vec{k} = (k_1, k_2, ..., k_m)$. An associated subvariety of $GR(l, n)$ is given by

$$\Omega_{\vec{c},\vec{k},l}(\mathbf{W}) = \{V \in GR(l, n) \mid \dim(V \cap W_i) \geq c_i \text{ for } 1 \leq i \leq m\}$$

Points on $FL(k_1, k_2, ..., k_m; n)$ parameterize a family of such subvarieties.

In a similar manner, one can build subvarieties of a more general flag manifold $FL(l_1, \ldots, l_s; n)$. The subvarieties will be written $\Omega_{C,\vec{k},\vec{l}}(\mathbf{W})$. The data that is determining the subvariety, $\Omega_{C,\vec{k},\vec{l}}(\mathbf{W})$, is the space from which we draw our fixed flags [e.g., $\mathbf{W} \in FL(k_1, \ldots, k_m; n)$], a space on which the subvariety lives [e.g., $FL(l_1, \ldots, l_s; n)$], and incidence constraints, $c_{i,j}$, stored in an $m \times s$ array C. We have

$$\Omega_{C,\vec{k},\vec{l}}(\mathbf{W}) = \{\mathbf{V} \in FL(\vec{l}; n) \mid \dim(V_j \cap W_i) \geq C_{i,j} \text{ for } 1 \leq i \leq m,\ 1 \leq j \leq s\}.$$

3.2 Schubert varieties of best fit

Suppose we are given a collection of l-dimensional subspaces $D = \{V_1, \ldots, V_r\}$ of $\mathbb{R}^n$. Each element in D can be thought of as a point in the Grassmannian $GR(l, n)$ thus we have r points on $GR(l, n)$. We seek to determine a Schubert variety of best fit to the r points. In order for this problem to make sense, we need to answer two questions: the first is "What class of Schubert varieties are you going to use to best fit the data?" and the second question is "What is the objective function you are trying to optimize when searching for a Schubert variety of best fit?." Intuitively, we are searching for a Schubert variety that comes as "close as possible" to the set of points determined by D. This should remind you of finding a "linear space of best fit" to a set of points in $\mathbb{R}^n$. For our purposes, given a point

or collection of points on $Gr(l,n)$ and a Schubert variety S, we further seek to have a measurement of closeness that is orthogonally invariant, i.e., is invariant to the action of the orthogonal group $O(n)$. Points on a Grassmannian correspond to subspaces and Schubert varieties are defined in terms of incidence conditions with respect to a fixed flag of subspaces. With this in mind, in order to achieve measurements that are orthogonally invariant, it is natural to write the measurement of closeness in terms of principle angles between the subspaces involved. There are many different ways in which this can be carried out and this leads to many different answers to the problem.

In what follows, let $D = \{V_1, V_2, \ldots, V_r\}$ be a collection of l dimensional subspaces considered as points on $GR(l,n)$. Given positive integers k and c, our goal is to find a point $W \in GR(k,n)$ such that the Schubert variety

$$\Omega_{c,k,l}(W) = \{V \in GR(l,n) \mid \dim(V \cap W) \geq c\}$$

comes as close as possible to the points in D. We will break this up into three parts. The first part will be to define a measurement of closeness, $d[V_i, \Omega_{c,k,l}(W)]$, between a single point $V_i \in GR(l,n)$ and the Schubert variety $\Omega_{c,k,l}(W)$ [with $W \in GR(k,n)$]. The second part will be to combine these single point measurements into a measurement of closeness between a set of points $D = \{V_1, V_2, \ldots, V_r\} \subset GR(l,n)$ and the Schubert variety $\Omega_{c,k,l}(W)$. The third part will be to find a $W^* \in GR(k,n)$ that optimizes this measure of closeness.

With respect to the third part, suppose we have fixed a real valued measurement of closeness between a set of points, $D \subset GR(l,n)$ and a Schubert variety $\Omega_{c,k,l}(W)$. For each point $W \in GR(k,n)$ we have effectively assigned a real number (the closeness measure) thus we have a function $F : GR(k,n) \to \mathbb{R}$. Since $GR(k,n)$ is a compact manifold, this real valued function will attain both its maximum and minimum values. Our goal is to describe an algorithm, implemented in a neural network, to find a point in $GR(k,n)$ that achieves either a maximum or a minimum of this function.

3.3 Examples of distance/closeness measures

Let V_1, V_2 be subspaces of $\mathbb{R}^n$ of dimension l. Recall that the principal angles between V_1 and V_2 satisfy $0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_l \leq \pi/2$. If $\Theta(V_1, V_2) = [\theta_1, \theta_2, \ldots, \theta_l]$ then $d_g(V_1, V_2) = \|\Theta(V_1, V_2)\|_2$ is known as the geodesic norm between V_1 and V_2. If $\sin\Theta(V_1, V_2) = [\sin\theta_1, \sin\theta_2, \ldots, \sin\theta_l]$ then $d_c(V_1, V_2) = \|\sin\Theta(V_1, V_2)\|_2$ is known as the chordal norm between V_1 and V_2.

We can generalize to the setting where V_1 and V_2 have potentially differing dimensions and we can modify the measure of the length of the principal angle vector between these subspaces. We will rename V_1 as V and V_2 as W to emphasize this flexibility in dimensional differences. Let V, W be subspaces of $\mathbb{R}^n$ of dimensions l and k and let $m = \min(l,k)$. Let c be a positive integer less than or equal to m and define $\Theta_c(V,W) = [\theta_1, \theta_2, \ldots, \theta_c]$ and $\sin\Theta_c(V,W) = [\sin\theta_1, \sin\theta_2, \ldots, \sin\theta_c]$. Given a vector norm, $\|\cdot\|_\alpha$ on $\mathbb{R}^c$, we can measure the "size" of the vector $\Theta_c(V,W)$ or of $\sin\Theta_c(V,W)$. We claim that in either case, this norm gives a measure of closeness between a point $V \in GR(l,n)$ and the Schubert variety $\Omega_{c,k,l}(W)$ by defining $d(V, \Omega_{c,k,l}(W)) = \|\Theta_c(V,W)\|_\alpha$ or by defining $d(V, \Omega_{c,k,l}(W)) = \|\sin\Theta_c(V,W)\|_\alpha$. To see that this is a measure of closeness, note that $\theta_1, \ldots, \theta_c$ are the c smallest principal angles between the subspaces. They correspond to the c smallest possible principal angles between c-dimensional subspaces of V and c-dimensional subspaces of W.

If we now pick a norm $\|\cdot\|_\beta$ on $\mathbb{R}^r$, once we have chosen a norm for measuring the size of $\Theta_c(V,W)$ or of $\sin\Theta_c(V,W)$, we can measure a Schubert variety's fit to a collection of l-dimensional subspaces $D = \{V_1, \ldots, V_r\}$ of R^n by

$$FIT(D, \Omega_{c,k,l}(W)) = \|d(V_1, \Omega_{c,k,l}(W)), d(V_2, \Omega_{c,k,l}(W)), \ldots, d(V_r, \Omega_{c,k,l}(W))\|_\beta$$

and we can define the Schubert variety of best fit to$D$ as $BEST(D, \Omega_{c,k,l}) = \Omega_{c,k,l}(W^*)$ where

$$W^* = arg\ min_{W \in GR(k,n)} FIT(D, \Omega_{c,k,l}(W))$$

For programming advantages, in the next section an optimization problem is described in terms of maximizing the norm of $\cos\Theta_c(V,W) = [\cos\theta_1, \cos\theta_2, \ldots, \cos\theta_c]$ instead of minimizing the norm of $\sin\Theta_c(V,W)$. The goal of the optimization problem is to find $BEST(D, \Omega_{c,k,l})$.

4 Optimization problem for SVBF

Consider a set of matrices $\{X_i\}_{i=1}^r$ with each $X_i \in \mathbb{R}^{n\times l}$ having orthonormal columns. Let $K \in \mathbb{R}^{n\times k}$ be an unknown matrix with orthonormal columns. Let $\mathcal{R}(X_i)$ denote the column space of X_i. Define θ_{ij} to be the jth smallest principal angle between $\mathcal{R}(X_i)$ and $\mathcal{R}(K)$.

Problem Statement: Given the set $\{X_i\}_{i=1}^r$ and an integer c with $1 \leq c \leq min(l,k)$, find K that comes as close as possible to satisfying $dim(\mathcal{R}(X_i) \cap \mathcal{R}(K)) \geq c$ for each of the subspaces $\mathcal{R}(X_i)$.

One approach is to find a matrix $K^* \in \mathbb{R}^{n\times k}$ such that

$$K^* = \arg\max_{K\in\mathbb{R}^{n\times l}} \frac{1}{rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \cos^2(\theta_{ij}) \tag{1}$$

subject to $K^TK = I$ and $1 \leq c \leq \min\{l,k\}$. We have normalized by the factor $1/rc$ so that the optimal value of the solution will be one. Note that for $c = \min\{l,k\}$ this is the flag mean (Marrinan et al., 2015).

Alternatively, we can solve

$$K^* = \arg\max_{K\in\mathbb{R}^{n\times k}} \frac{1}{rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \cos(\theta_{ij}) \tag{2}$$

subject to $K^TK = I$ and $1 \leq c \leq \min\{l,k\}$. Note that for $c = \min\{l,k\}$ this is the flag median (Mankovich et al., 2022).

FIGURE 2
Images from the Cat and Dog dataset. The **(top)** row corresponds to the data in a single cat sample with three elements. Similarly, the **(bottom)** row corresponds to a dog sample consisting of three elements.

4.1 SVBF optimization problem formulation

For simplicity, in this paper we will focus on the case $c = 1$, i.e., we are looking for the column space of each X_i to be close to intersecting the column space of K in at least one dimension.

Suppose we are given a set of matrices $\{X_i\}_{i=1}^r$, each $X_i \in \mathbb{R}^{n\times l}$ satisfying $X_i^T X_i = I_l$. We would like to find a matrix K whose column space intersects the column space of each X_i in at least a one dimensional space. This is achieved if and only if the first principle angle between the column space of K and the column space of X_i is equal to zero for each i. Let $\theta_1(K, X_i)$ denote the first principle angle between the column space of K and the column space of X_i. In the optimization below we seek to find a matrix K that maximizes the function $\sum_{i=1}^r \cos^2(\theta_1(K, X_i))$. This amounts to finding a a matrix $K^* \in \mathbb{R}^{n\times k}$ such that

$$K^* = \arg\max_{b_i, K} \sum_{i=1}^{r} b_i^T K^T X_i X_i^T K b_i \tag{3}$$

subject to $b_i \in \mathbb{R}^{k\times 1}$, $\|b_i\| = 1$, $K \in \mathbb{R}^{n\times k}$, and $K^T K = I_k$.

In the second optimization problem, given below, we seek to find a matrix K that maximizes the function $\sum_{i=1}^r \cos(\theta_1(K, X_i))$. This amounts to finding a matrix $K^* \in \mathbb{R}^{n\times k}$ such that

$$K^* = \arg\max_{a_i, b_i, K} \sum_{i=1}^{r} a_i^T X_i^T K b_i \tag{4}$$

subject to $a_i \in \mathbb{R}^{l\times 1}$, $\|a\| = 1$, $b_i \in \mathbb{R}^{k\times 1}$, $\|b_i\| = 1$, $K \in \mathbb{R}^{n\times k}$, and $K^T K = I_k$.

4.2 SVBF optimization problem implementation with PyTorch

All of the experiments in this paper are conducted for the special case where $c = 1$. This case, along with the relative simplicity of the dataset that we used, allows us to directly initialize the problem with the PyTorch class. The unknown matrix K and vectors $\{b_i\}$ in this case are initialized in the PyTorch class as two sets of parameters: $\{K_{ij}\}$, $1 \le i \le n, 1 \le j \le k$, and $\{b_{ij}\}$, $1 \le i \le r$, $1 \le j \le k$. The Lagrangian, associated with the problem given by Equation (3), is considered via the Adam optimizer and provides a path to the approximate solution:

$$Loss_K\left(K, b\right) = \tag{5}$$
$$-\sum_{i=1}^{r} b_i^T K^T X_i X_i^T K b_i + \lambda_0\left(K^T K - I_k\right) + \sum_{i=1}^{r} \lambda_i\left(\|b_i\| - 1\right)$$

Let's focus on the first part of this expression as the part containing a majority of the complexity. In our design, for each sample it is calculated in three steps:

1. $L_i^1 = K^T X_i$
2. $L_i^2 = L_i^1 L_i^{1T}$
3. $L_i^3 = b_i^T L_i^2 b_i$

Based on this chain of matrix multiplications and assuming that $n \gg k$ we can calculate the time complexity of the forward pass as $\mathcal{O}\left(2k(nl + kl + 2k)\right) = \mathcal{O}\left(nkl\right)$. The backward pass complexity, calculated based on the chain of multiplications of respective Jacobian matrices, is also equal to $\mathcal{O}\left(nkl\right)$. Hence, along with complexity of the Adam optimizer, equal to $\mathcal{O}\left(nk\right)$, the overall complexity of one iteration is equal to $\mathcal{O}\left(nkl\right)$ and the complexity of the entire optimization process is equal to $\mathcal{O}\left(tnklr\right)$, where t is

the number of iterations and r is a number of l-dimensional inputs. Note that it is effectively equal to the complexity of the training of one fully connected layer with k nodes at the output given that it is trained on $l \times r$ n-dimensional vectors with the same number of iterations t.

The given approximations of the complexity are derived following the traditional approach that was common before the era of GPU's. The GPU's benefit from parallelizing an immense number of simple operations and are capable of accelerating matrix multiplications by orders of magnitude. We run all of our experiment on V-100 Tesla GPU's, hence to provide a better understanding of performance in actual experiments we present a computational profiling in Table 1. The profiling was conducted for 100 randomly generated inputs. The dimensionalities of inputs vary as $n \in \{8^2, 8^3, 8^4, 8^5\}$, $l \in \{1, 4^1, 4^2, 4^3\}$, while the dimensionality of K varies as $k \in \{8 \times 1, 8 \times 4^1, 8 \times 4^2, 8 \times 4^3\}$. Given that SVBF optimization can also be run as a training with batches for larger number of samples, this range of parameters supposedly covers several of the possible cases that one may encounter in real experiments.

In fact, instead of including the unit length condition for each of the b_i's in the cost function, we use the normalizing transformations of b_i's suggested in Kirby and Miranda (1996), which doesn't introduce much complexity and, at this point, has already been implemented in PyTorch as a built-in routine. The resulting loss function is as follows:

$$Loss_K\left(K, b\right) = -\sum_{i=1}^{r} b_i^T K^T X_i X_i^T K b_i + \lambda_0 \left(K^T K - I_k\right) \qquad (6)$$

where $\|b_i\| = 1 \, \forall i \in \{1, r\}$ by construction of the PyTorch class. As it was mentioned before this function is minimized with Adam optimizer and the number of iterations is equal to 40,000. The following is a list of some other details important for reproduction of results:

- λ_0 = 10,000
- Initialization of parameters K and $\{b_i\}$:
  element-wise random numbers from a uniform distribution on the interval [0,1]
- Adam optimizer settings:
  lr = 0.001, betas = (0.9,0.999), eps = 1e−08, weight_decay = 0, amsgrad = False
- Python version: 3.6.8
- torch version: 1.10.1
- cuda version: 11.0

All these settings are preserved exactly the same across all experiments including SVBF problem solving. For the faster processing in benchmarking experiments we also leverage Ray 2.0.0 python package to run several experiments in parallel.

4.3 Illustrative example

In this paper we now present results from the given implementations of the SVBF algorithm. Given that our objective is a comparative analysis of these algorithms, we focus on a modestly sized Cat and Dog dataset consisting of 99 images of cats and 99 images of dogs. All the images are 64 × 64 greyscale images; you can see some of the representatives of each class in Figure 2. We preprocess the data by flattening each image into a vector of length $n = 4,096$ and scaling the entries into the range from −1 to 1. The SVBF algorithm operates over sets of subspaces, therefore we split the data into equally sized collections of l vectors and extract an orthonormal basis for each set. The resulting orthonormal bases are used as the inputs. In other words, the inputs, or samples, consist of tall orthonormal matrices of dimension $\mathbb{R}^{n \times q}$ where $q = l$ for training data and $q = m$ for test data. We chose to allow for l and m to be distinct so that we may consider the cases when dimensionality of samples in training and test data might differ. Importantly, no cat or dog vector is used in more than one sample. Further, we never mix classes within one sample, samples consist of sets of only dog vectors or only cat vectors. For example, the notations $\{X_i^{train}\}$, $l = 3$ describes the set of three-dimensional bases of mono-class sets of scaled image-vectors sampled from the training data.

In Figure 3, we show sample solutions of Schubert Varieties of Best fit. In each case the matrix K to be determined is chosen to have one column and these solutions are displayed for cats and dogs individually. It is interesting to compare the solutions to Equation (3) and Equation (4). We see that for $cos(\theta)$ the Schubert variety matrices K show higher resolution detail while

TABLE 1 Tables of processing times for one iteration of optimization loop in milliseconds. The number of samples is fixed to 100, while the dimensionality of inputs l, dimensionalilty k of K, and dimensionality n of ambient space vary.

$l \setminus n$	8^2	8^3	8^4	8^5
1	1.18	1.23	1.58	2.77
4^1	1.27	1.70	5.00	49.31
4^2	1.26	1.69	5.00	45.27
4^3	1.30	1.82	5.20	48.80
$l \setminus n$	8^2	8^3	8^4	8^5
1	1.30	1.49	2.45	1
4^1	1.47	1.97	5.09	28.18
4^2	1.47	1.98	5.18	30.53
4^3	1.54	2.13	5.87	52.20
$l \setminus n$	8^2	8^3	8^4	8^5
1	1.54	2.13	8.62	160.36
4^1	1.86	2.72	10.43	191.06
4^2	1.87	2.76	10.91	192.08
4^3	2.01	3.01	11.99	204.16
$l \setminus n$	8^2	8^3	8^4	8^5
1	4.09	6.76	73.90	491.70
4^1	5.17	7.72	77.40	506.52
4^2	5.49	7.97	80.87	518.80
4^3	6.58	9.96	89.52	616.49

$k = 8 \times 1$. $k = 8 \times 4^1$. $k = 8 \times 4^2$. $k = 8 \times 4^3$.

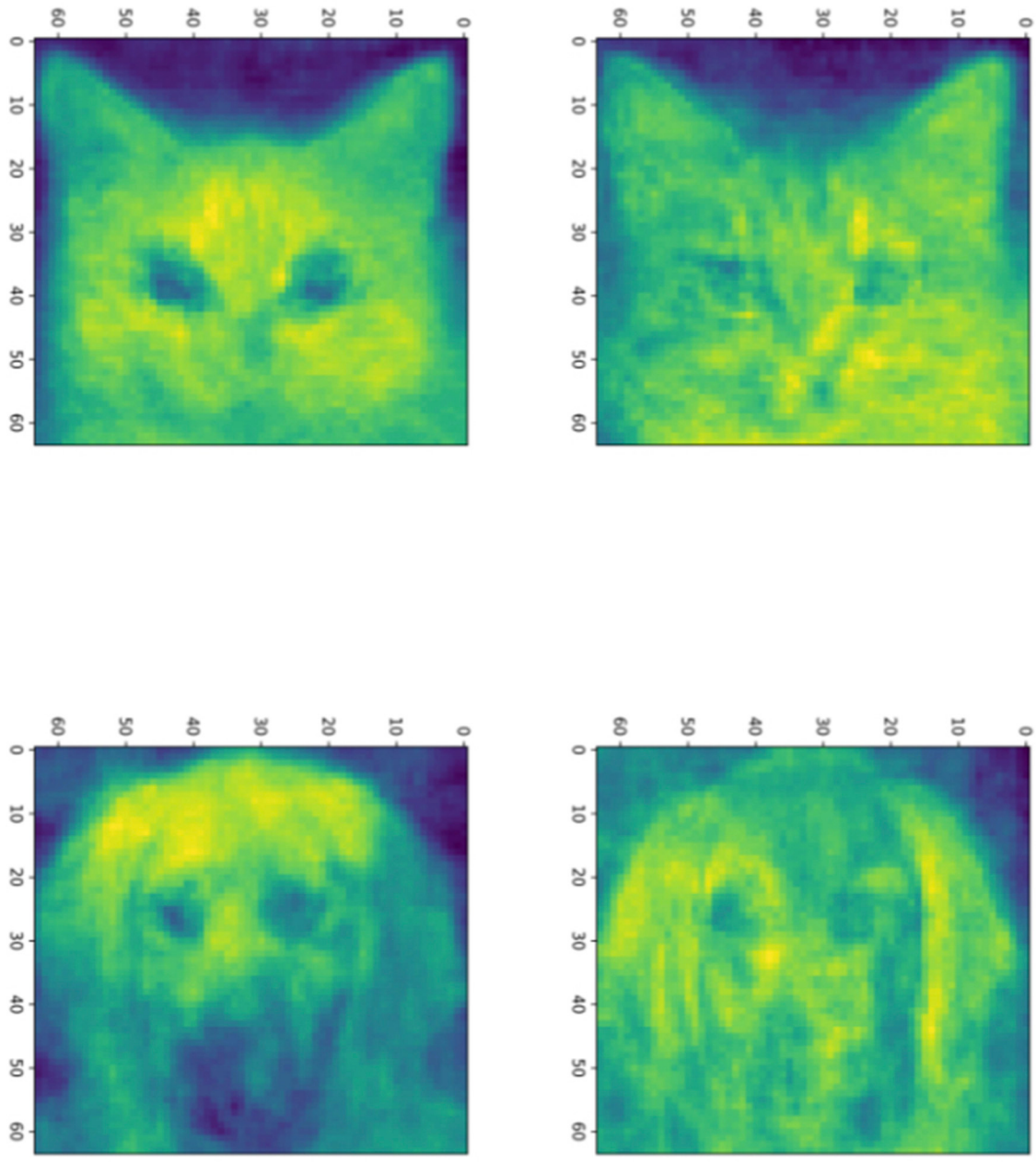


FIGURE 3
Solutions to the Schubert Variety of best fit problem for $a = 1$. The **(left)** column consists of solutions to Equation (3) while the **(right)** column consists of solutions to Equation (4). In each case there are three images in each sample. The **top** and **bottom** rows use cat and dog samples, respectively.

for $\cos^2(\theta)$ the features are larger scale. This example underscores the potential significance in the selection of distance measure in computing SVBFs. For all these examples we select three images in each sample.

4.4 Optimal dimension of K

We illustrate the relative rates of conve rgence of the SVBF optimization problems in Figure 4. Figure 4A shows the value of the objective function for the optimization problem given by Equation (3), i.e., the average sum of the squared cosines of the 1st principal angles for 10 two-dimensional subsets of cats sampled from both training and test datasets versus the dimension k of the solution subspace K. We can see that the objective function for test samples effectively flattens out after k=2. Similar behavior can be observed for the dog dataset, presented in Figure 4B, except the flattening is smoother and starts approximately at k =4. The results for the combined datasets, presented in Figure 4C, show that the flattening also starts at around k = 4. These plots lead us to several observations. Firstly, the flattening of the curves itself suggests that more columns in K do not lead to improvements in the solution. Thus, the set of points $\{X_i\}$ has an intrinsic subspace dimension as captured by K. We see that the dimension of this

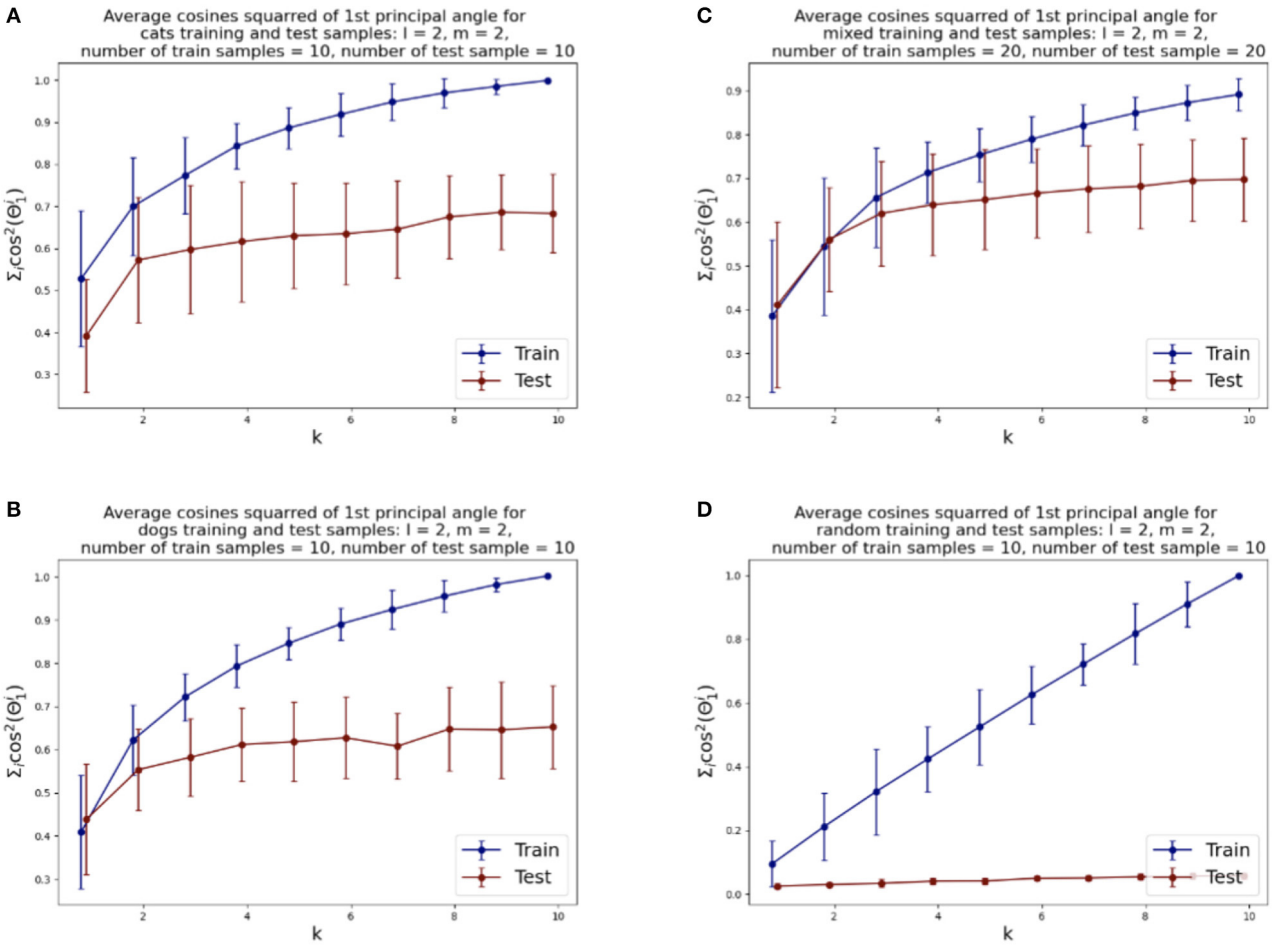


FIGURE 4
Average squared-cosines of the 1st principle angle for training and test data sampled from different classes, from both classes and sampled from the uniform distribution on the Grassmann manifold: **(A)** cat class, **(B)** dog class, **(C)** cats and dogs classes combined, **(D)** random samples.

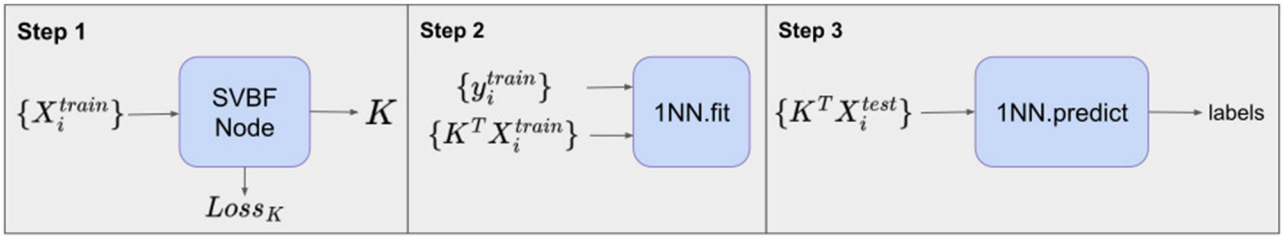


FIGURE 5
Workflow of Cat and Dog classification experiment using Algorithm I.

subspace is smaller than the direct sum of the dimensions of the class-specific subspaces. This does not come as too big a surprise when one considers the correlation between the images. Finally, the Figure 4D illustrates how samples, randomly selected from the uniform distribution (with respect to Haar measure) on a Grassmann manifold, do not possess the same structure and the number of columns required in K does not converge. Experiments with other dimensions of samples have also been conducted, and they align with the results reported here. Other approaches to dimension optimization are also possible with the SVBF method, e.g., task-specific optimizations that will be considered in Section 5.

5 SVBF as an abstract node

Here, we explore the application of the SVBF as a computational unit in a feed-forward neural

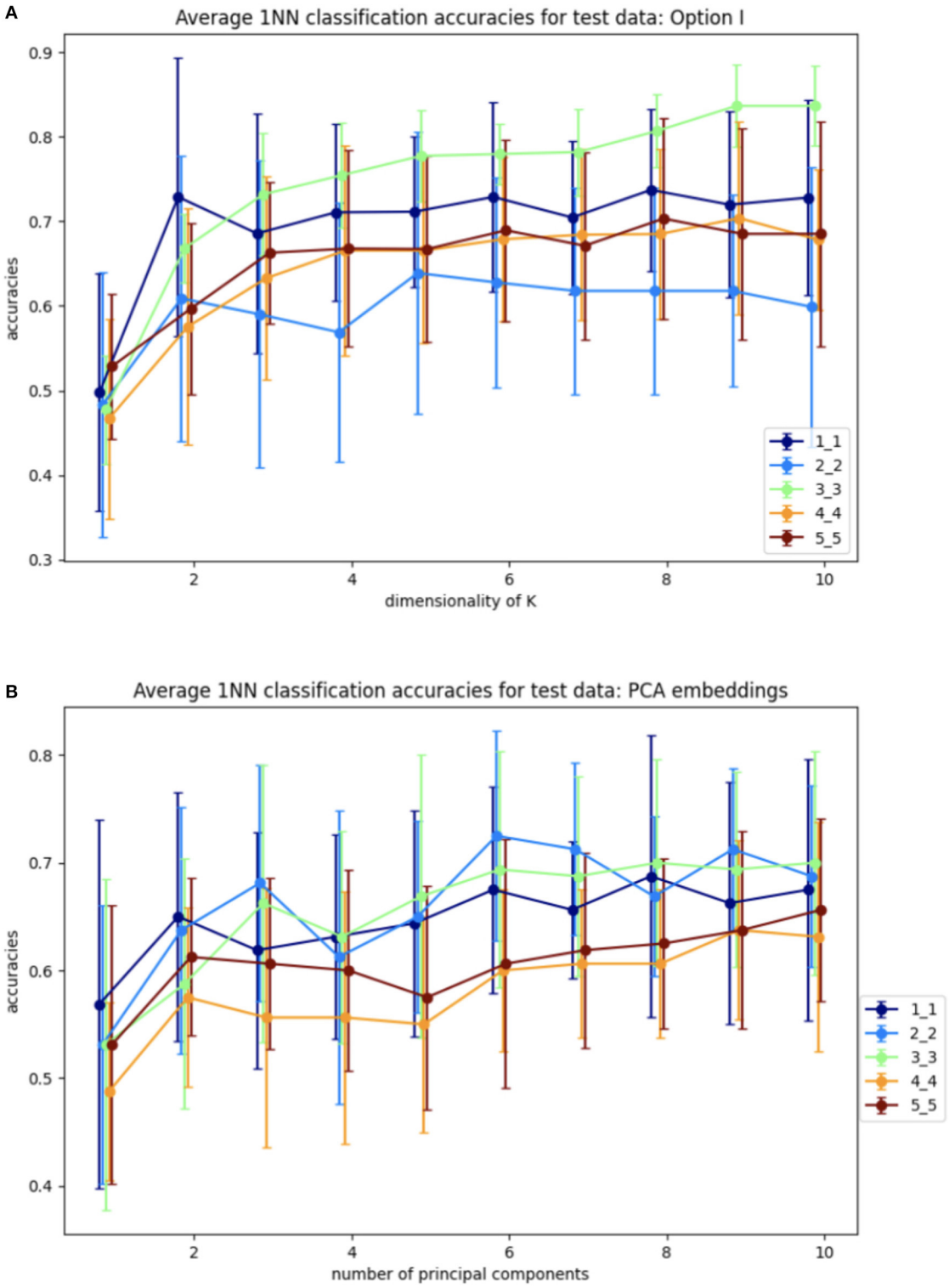


FIGURE 6
Average accuracies with error bars for 1NN-classification of test data for Y embeddings from Algorithm I and for PCA embeddings. **(A)** Y embeddings, plots are labeled as *l_m* depending on the dimensions of training samples *l* and dimensions of test samples *m*. **(B)** PCA embeddings, plots are labeled as *m_m* depending on dimensions of test samples *m*.

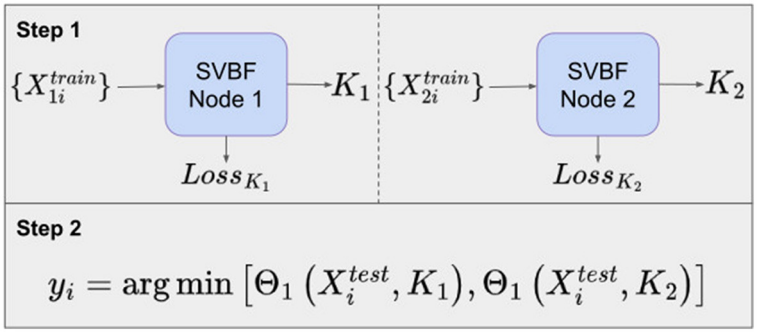


FIGURE 7
Workflow of Cat and Dog classification experiment using Algorithm II.

network. We shall see that Schubert varieties of best fit provide a natural transformation of the data into the coordinates of the learned Schubert variety of best fit K.

5.1 Algorithm I

Given a solution K and a data sample X_i it is natural to consider the change of coordinates of the sample that can be implemented in the neural network, i.e.,

$$Y_i = K^T X_i$$

The design of the classification experiment for this case is presented in Figure 5.

Step 1 involves training the SVBF Node, the outcome of which is the representative K of the SVBF. This K is then used in Step 2 to compute the change of coordinates to produce Y. This is done using training data. Further, in Step 2 the 1-nearest neighbor (1NN) classification is implemented using the scikit-learn Python package. Step 3 involves the application of 1NN to the test data and the results are shown in Figure 6. Figure 6A illustrates the average for 10 different samplings of a 1NN-classifier applied to the $\{Y_i\}$ embeddings of Cat and Dog data. In each sampling, the data is split randomly into train and test datasets with proportions 0.78 and 0.22, respectively. Benchmarking is performed for different dimensions $l = m = 1, \ldots, 5$ of samples in training and tests subsets, and different dimensions $k = 1, \ldots, 10$ for K. The accuracy grows with the dimension of the samples, but interestingly starts decreasing for $l \geq 3$. At the same time, increasing the dimension of K also significantly increases the accuracy up to dimension 3 and only slightly effects the accuracy above that level. This low accuracy is a result of using the Frobenius norm to compute distances between matrices. Later we will see that using subspace distances in general produces superior results. For a better understanding of the benefits of the SVBF method, we provide side by side the results for classification based on PCA-embeddings as well (Figure 6B). In this first experiment, the performance of the classification method based on learning K is slightly better than the one based on learning a representative subspace based on PCA-analysis.

5.2 Algorithm II

Another possible approach is to learn a representative K for each class of data. In our example, in Step 1, we propose to learn K_1 as a solution to Equation (3) for the cat class, and K_2 as a solution to Equation (3) for the dog class. Now these matrices K_1, K_2 can be used to classify the data. To assign a class to an unknown sample X_i in Step 2, we compute the smallest principle angle between X_i and each of K_1 and K_2. The smallest of these two angles provides the classification. The diagram for such an experiment with the Cat and Dog dataset is presented in Figure 7. Labeling of the test data is performed by finding the nearest, in terms of smallest principle angle to K, for each sample.

Figure 8A shows the average classification accuracy across 10 different samplings of the test data. Again, this procedure is implemented for a range of l, m, and k. The data was split into training and test sets with the same ratio as in the Algorithm I experiments.

The benchmarking plots indicate a significantly more accurate classification versus Algorithm I for the comparable cases. Due to the higher overall accuracy of this design, we also investigated it for different dimensions of training and test samples, which can be useful for a wide variety of datasets, specifically when test samples cannot be grouped by labels. As before, increasing the dimension k of K leads to higher accuracies for all cases. However, in contrast to Algorithm I, now the accuracy increases monotonically with the dimensions l, m.

For further comparative analysis, we also calculate PCA embeddings for each class and label test samples, based on the nearest, in terms of the smallest principle angle subspace captured by PCA-analysis. We repeat this experiment 10 times for different samplings similar to the Algorithm I experiment. The results of this experiment are shown in Figure 8B. The lower accuracies for the class-specific PCA experiment indicate the SVBF optimization problem is capturing additional useful information that is helpful for classification.

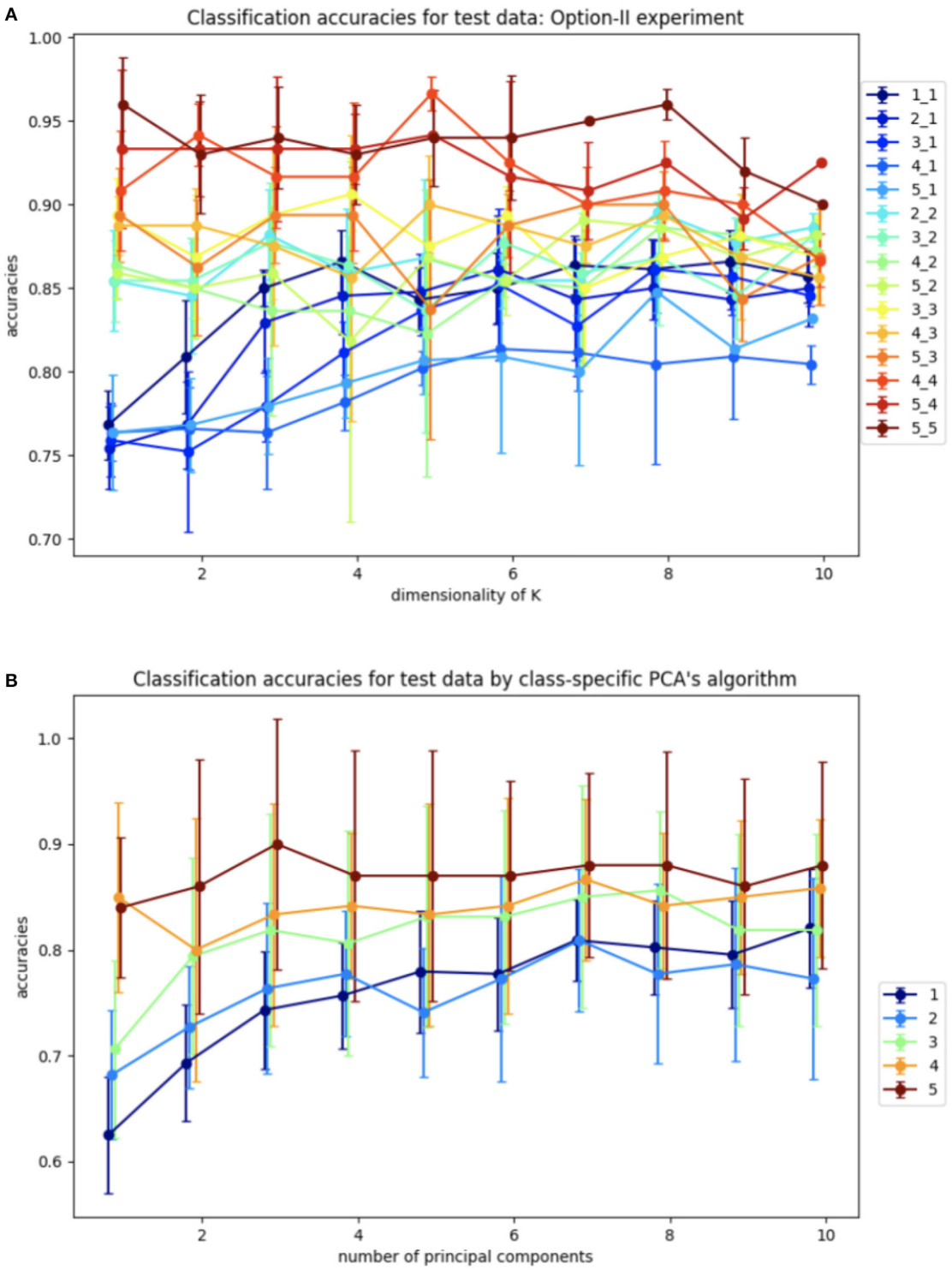


FIGURE 8
Average accuracies of classification of test set by Algorithm II design and design based on the closest class-specific PCA subspaces: **(A)** Algorithm II design, plots are labeled as *l_m* depending on the dimensions of training samples *l* and dimensions of test samples *m*. **(B)** Closest PCA design, plots are labeled as *m* depending on dimensions of test samples *m*.

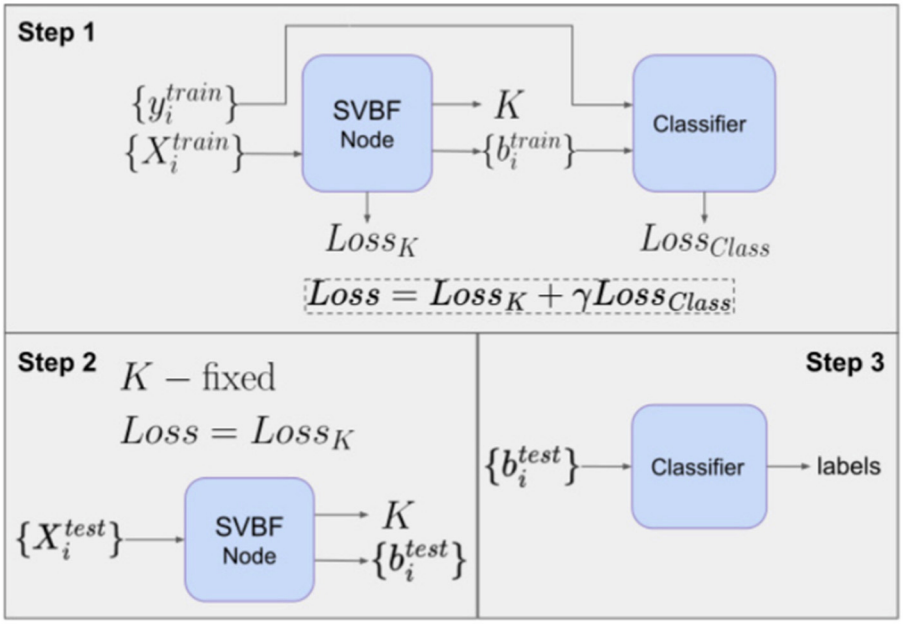


FIGURE 9
Workflow of Cat and Dog classification experiment using Algorithm III.

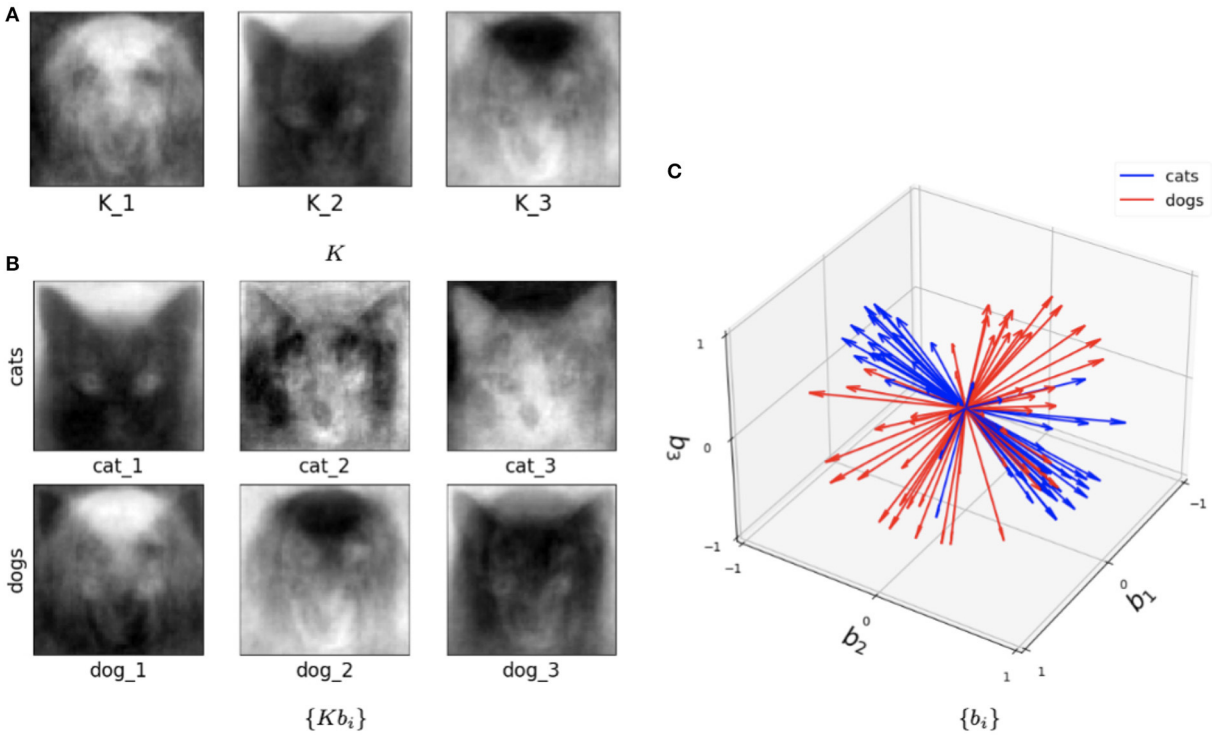


FIGURE 10
Visualization of the outputs of the trained SVBF node and generated embeddings for the entire Cat and Dog dataset, k=3, and l=2. **(A)** Learned basis K, **(B)** some samples reconstructed in ambient space from embeddings, **(C)** learned embeddings.

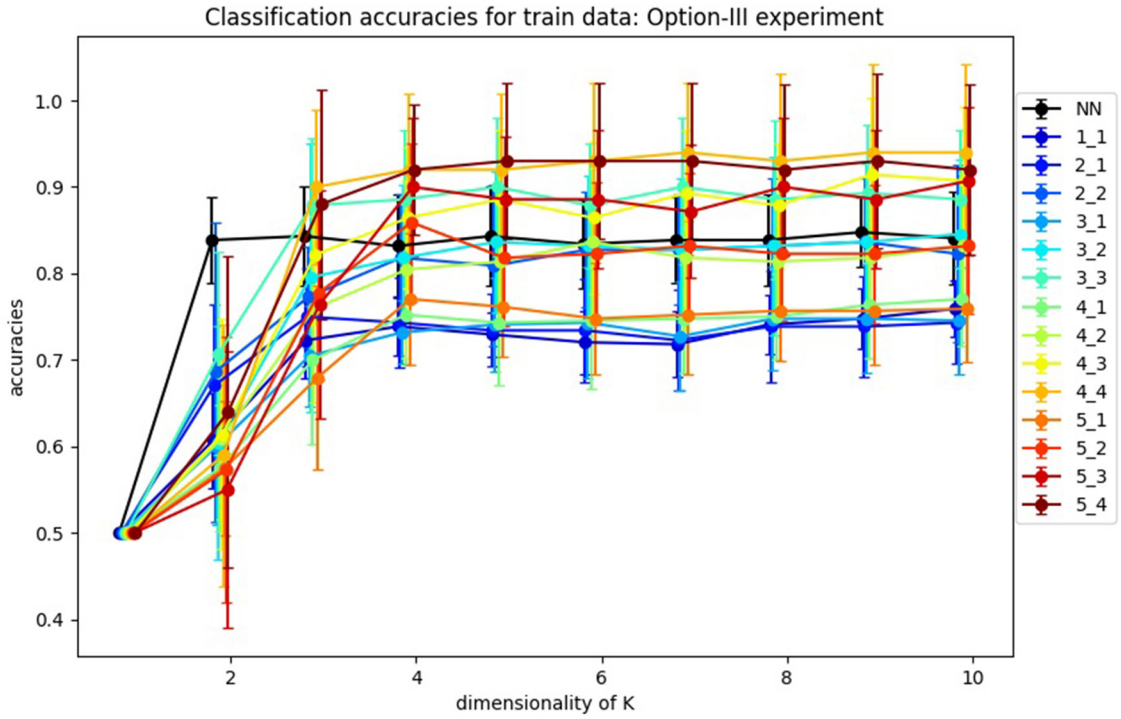


FIGURE 11
Average accuracies of classification of test data by Algorithm III design. Accuracies for one-layer NN design are labeled as "NN," and accuracies for SVBF-Node design as "*l*_*m*," where *l* is dimensionality of train samples and *m* is dimensionality of test samples.

5.3 Algorithm III

Note that the element $\hat{k} \in \mathcal{R}(K)$ given by

$$\hat{k}_i = Kb_i$$

is the vector in $\mathcal{R}(K)$ that is closest to the span of X_i. There is a natural association between X_i and b_i and this can be exploited for classification. Hence, a consequence of Equation (3) is that once we approximate K we can use b_i, the coordinates of the closest to X_i vector in K, as a proxy for X_i in the classification. One of the possible experiments exploiting this option is classification with neural networks outlined in Figure 9.

This experiment also illustrates how SVBF nodes can be stacked into a larger network and trained simultaneously with other parts of the network. In this case, we attach a classifier that takes outputs b_i of SVBF nodes as inputs, learns centroids in a training process and generates soft labels at the inference step. In Step 1, we optimize K and b_i independently at each iteration. In other words, K is not adjusted by any contribution to the gradient from the classification error. Generally, the outline given in Figure 9 does not require detaching the classifier's error back-propagation from that of an SVBF node. However, detaching the training of K and b_i, as we have done here, seems to improve performance without loss of the benefits of parallelization. The loss function for Step 1 is defined as a weighted sum of three loss functions:

$$\begin{aligned} Loss_\theta &= -\sum_{i=1}^{M} b_i^T K^T X_i X_i^T K b_i \\ Loss_{orth} &= K^T K - I_{k\times k} \\ Loss_{Class} &= -\sum_{j=1}^{k} y_{ij} \log\left(\text{logit}_{ij}\right) \\ Loss &= \alpha Loss_\theta + \beta Loss_{orth} + \gamma Loss_{Class} \end{aligned} \tag{7}$$

where $\text{logit}_i = Softmax((b_i^T W)^2, T)$ with W learnable centroids, and the hyperparameter T. The output of Step 1 is the matrix K, set $\{b_i^{train}\}$ and the classifier W. Now, given K, W, the prediction of the labels is accomplished in Steps 2 and 3.

In Step 2, we initialize the SVBF node with the K computed in Step 1 to determine the coordinates b_i^{test}, associated with the test samples X_i^{test}, using the loss function $Loss_K$ similar to Step 1, but without the classification component and with K fixed. In Step 3, we use the classifier W trained in Step 1 to map the b_i^{test} to their class label.

This hybrid network was also tested with the Cat and Dog dataset following exactly the same preprocessing pipeline as in the Algorithm I and Algorithm II experiments. Figure 10 illustrates trained parameters for such a network for the entire Cat and Dog dataset with $l = 2$ and k=3. As we now describe, this picture captures many interesting features of the method. Figure 10A depicts the three columns of K. Interestingly one column is a dog, and one is

clearly a cat! But all columns seem to capture salient features in the data. In Figure 10B, we show the directions Kb_i for three cat samples and three dog samples, where we recall that each sample is a $4,096 \times 2$ matrix. So each of these six figures is a direction in the span of K that is closest to the data training sample. In Figure 10C, we show the set of all features in terms of b_i for cats (blue) and dogs (red). We note that their clear separation is a reflection of the fact that this method captures information capable of discriminating between cats and dogs.

In Figure 11, you can find the classification accuracy for test data for different dimensions of training and tests samples as well as different dimensions for K. It is very interesting that the accuracy of the classification problem levels off at 4 dimensions, which is consistent with the optimal size $k = 4$ of K as suggested in Figure 4. Hence we view $k = 4$ as the apparent *working* dimension for K. We see the accuracy increases monotonically with the dimension of the samples. Importantly, the results in which the samples have dimensions >5, which we are not reporting here, support this observation. However, given our limited data we were not able to pursue this behavior further.

In the same Figure 11, we also report the average accuracies for the classification of l_2-normalized vectors used as inputs. In this design, instead of an SVBF-Node, we use one fully connected layer with k nodes at the output and with the hyperbolic tangent as the activation function (all other settings are kept the same). In Section 4.2, it was shown that this network has the same computational complexity as the SVFB-Node. Along with the higher resulting average accuracies for some cases this method also has an advantage of ∼10 times faster convergence (in terms of the required number of iterations). On the other hand, we can see that for dimensionalities of test samples ≥ 3 the SVBF method is more accurate. It's important to note here that benchmarking against the regular networks processing vector inputs wasn't within the focus of this paper. Our prime goal was to show that in the suggested framework a big part of the progress made with the regular neural networks during the last decades can be directly leveraged in the case of sets of datasets.

6 Conclusions and future work

In this paper we proposed a geometric approach for the analysis of sets of datasets using the idea of a Schubert Variety of Best Fit. We formulated two optimization problems and compared them on a two class data set comprised of sets of cats and dogs. We proposed three distinct algorithms for data classification and explored and benchmarked these using the same dataset and preprocessing pipeline. Algorithm I uses a single solution K to characterize the data and a Frobenius norm to measure distances. Overall, this algorithm performed poorly when compare to Algorithms II and III which generated promising results. Algorithm II, based on learning two SVBFs, provides the most accurate classifications. This algorithm explicitly builds a model for the cats and dogs through their respective SVBFs and appears to be more capable of addressing complex structures of data, including classes with higher within-class variance. Algorithm III introduces the idea of using auxiliary features to perform data classification, and is based on a single model K.

All three algorithms suggest that there is a best dimension for the representative subspace and that exceeding this leads to no improvement in classification accuracy. Hence the SVBF approach appears to provide a measure of complexity of a set of data sets through this *working* dimension. Interestingly, Algorithm III provides a very clear signal for this optimal dimension through a classification criterion. Algorithm III illustrates how the SVBF node can be used as an abstract unit of computation in a neural network. This enables researchers to process sets of datasets in the same spirit as sets of vectors are processed in a variety of ML libraries based on neural networks.

It is worth noting that all the experimental results provided in this paper are based on approximations of one or multiple K's intersecting all training samples or special label groups only in one direction. The preliminary results indicate the promise of the SVBF approach for finding a representative subspace for the set of datasets in classification tasks. The natural path forward is to increase the number of intersections in the SVBF optimization problem, i.e., increase the number of angles being used in the optimization problem. We also plan to explore this approach on more realistic data sets and explore how this impacts our ability to determine optimal K. We anticipate the main challenges will be in parallelizing the code efficiently to make larger problems computationally feasible.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

KK: Writing—original draft, Writing—review & editing. MK: Writing—original draft, Writing—review & editing. CP: Writing—original draft, Writing—review & editing.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. This work was partially supported by the DARPA Geometries of Learning Program under Award No. HR00112290074.

Acknowledgments

We would like to thank DARPA for support under the Geometries of Learning program grant number HR00112290074.

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.

Publisher's note

References

Anderle, M., Hundley, D., and Kirby, M. (2002). The bilipschitz criterion for mapping design in data analysis. *Intell. Data Anal.* 6, 85–104. doi: 10.3233/IDA-2002-6106

Beveridge, J. R., Draper, B. A., Chang, J.-M., Kirby, M., Kley, H., and Peterson, C. (2008). Principal angles separate subject illumination spaces in YDB and CMU-pie. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 351–363. doi: 10.1109/TPAMI.2008.200

Broomhead, D., and Kirby, M. (2000). A new approach for dimensionality reduction: theory and algorithms. *SIAM J. Appl. Math.* 60, 2114–2142. doi: 10.1137/S0036139998338583

Broomhead, D., and Kirby, M. (2001). The Whitney reduction network: a method for computing autoassociative graphs. *Neural Comput.* 13, 2595–2616. doi: 10.1162/089976601753196049

Ghosh, T., and Kirby, M. (2022). Supervised dimensionality reduction and visualization using centroid-encoder. *J. Mach. Learn. Res.* 23, 20–21.

Hebb, D. O. (1949). The first stage of perception: growth of the assembly. *Organ. Behav.* 4, 78–60.

Hertz, J., Krogh, A., Palmer, R. G., and Horner, H. (1991). Introduction to the theory of neural computation. *Am. Instit. Phys.* 44:70. doi: 10.1063/1.2810360

Hundley, D., Kirby, M., and Miranda, R. (1995). "Spherical nodes in neural networks with applications," in *Intelligent Engineering Through Artificial Neural Networks, Vol. 5*, eds S. Dagli, B. Fernandez, J. Ghosh, and R. S. Kumara (New York, NY: The American Society of Mechanical Engineers), 27–32.

Kingma, D. P., and Welling, M. (2019). An introduction to variational autoencoders. *Found. Trends Mach. Learn.* 12, 307–392. doi: 10.1561/2200000056

Kirby, M., and Miranda, R. (1996). Circular nodes in neural networks. *Neural Comput.* 8, 390–402. doi: 10.1162/neco.1996.8.2.390

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.* 37, 233–243. doi: 10.1002/aic.690370209

Kvinge, H., Farnell, E., Kirby, M., and Peterson, C. (2018a). "A GPU-oriented algorithm design for secant-based dimensionality reduction," in *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)* (Geneva), 69–76. doi: 10.1109/ISPDC2018.2018.00019

Kvinge, H., Farnell, E., Kirby, M., and Peterson, C. (2018b). "Too many secants: a hierarchical approach to secant-based dimensionality reduction on large data sets," in *2018 IEEE High Performance Extreme Computing Conference (HPEC)* (Waltham, MA), 1–7. doi: 10.1109/HPEC.2018.8547515

Ma, X., Kirby, M., and Peterson, C. (2021). "The flag manifold as a tool for analyzing and comparing sets of data sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops* (Montreal, BC), 4185–4194. doi: 10.1109/ICCVW54120.2021.00465

Mankovich, N., King, E. J., Peterson, C., and Kirby, M. (2022). "The flag median and flagirls," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (New Orleans, LA), 10339–10347. doi: 10.1109/CVPR52688.2022.01009

Marrinan, T., Beveridge, J. R., Draper, B., Kirby, M., and Peterson, C. (2015). "Flag manifolds for the characterization of geometric structure in large data sets," in *Numerical Mathematics and Advanced Applications-ENUMATH 2013*, eds T. J. Barth, Mi. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick (Berlin; Heidelberg: Springer), 457–465. doi: 10.1007/978-3-319-10705-9_45

Marrinan, T., Draper, B., Beveridge, J. R., Kirby, M., and Peterson, C. (2014). "Finding the subspace mean or median to fit your need," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Columbus, OH), 1082–1089. doi: 10.1109/CVPR.2014.142

McCulloch, W. S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* 5, 115–133. doi: 10.1007/BF02478259

Oja, E. (1992). Principal components, minor components and linear neural networks. *Neural Netw.* 5, 927–935. doi: 10.1016/S0893-6080(05)80089-9

Rosenblatt, D., Lelu, A., and Georgel, A. (2002). "Learning in a single pass: a neural model for principal component analysis and linear regression," in *Proceedings of the IEE International Conference on Artificial Neural Networks* (London), 252–256.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Advances in Neural Information Processing Systems, Vol. 30* (Long Beach, CA).

Vidal, R., Ma, Y., and Sastry, S. (2005). Generalized principal component analysis (GPCA). *IEEE Trans. Pattern Anal. Mach. Intell.* 27, 1945–1959. doi: 10.1109/TPAMI.2005.244

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 19 December 2023
DOI 10.3389/fcomp.2023.1275026

OPEN ACCESS

EDITED BY
Yunye Gong,
SRI International, United States

REVIEWED BY
Jesús Malo,
University of Valencia, Spain
Peter C. Doerschuk,
Cornell University, United States

*CORRESPONDENCE
Nathaniel Blanchard
nathaniel.blanchard@colostate.edu

RECEIVED 09 August 2023
ACCEPTED 20 November 2023
PUBLISHED 19 December 2023

CITATION
Pickard W, Sikes K, Jamil H, Chaffee N, Blanchard N, Kirby M and Peterson C (2023) Exploring fMRI RDMs: enhancing model robustness through neurobiological data. *Front. Comput. Sci.* 5:1275026. doi: 10.3389/fcomp.2023.1275026

Exploring fMRI RDMs: enhancing model robustness through neurobiological data

William Pickard[1], Kelsey Sikes[1], Huma Jamil[1], Nicholas Chaffee[1], Nathaniel Blanchard[1]*, Michael Kirby[1,2] and Chris Peterson[2]

[1]Department of Computer Science, Colorado State University, Fort Collins, CO, United States, [2]Department of Mathematics, Colorado State University, Fort Collins, CO, United States

Artificial neural networks (ANNs) are sensitive to perturbations and adversarial attacks. One hypothesized solution to adversarial robustness is to align manifolds in the embedded space of neural networks with biologically grounded manifolds. Recent state-of-the-art works that emphasize learning robust neural representations, rather than optimizing for a specific target task like classification, support the idea that researchers should investigate this hypothesis. While works have shown that fine-tuning ANNs to coincide with biological vision does increase robustness to both perturbations and adversarial attacks, these works have relied on proprietary datasets—the lack of publicly available biological benchmarks makes it difficult to evaluate the efficacy of these claims. Here, we deliver a curated dataset consisting of biological representations of images taken from two commonly used computer vision datasets, ImageNet and COCO, that can be easily integrated into model training and evaluation. Specifically, we take a large functional magnetic resonance imaging (fMRI) dataset (BOLD5000), preprocess it into representational dissimilarity matrices (RDMs), and establish an infrastructure that anyone can use to train models with biologically grounded representations. Using this infrastructure, we investigate the representations of several popular neural networks and find that as networks have been optimized for tasks, their correspondence with biological fidelity has decreased. Additionally, we use a previously unexplored graph-based technique, Fiedler partitioning, to showcase the viability of the biological data, and the potential to extend these analyses by extending RDMs into Laplacian matrices. Overall, our findings demonstrate the potential of utilizing our new biological benchmark to effectively enhance the robustness of models.

KEYWORDS

brain-inspired neural networks, computational neuroscience, deep learning, geometric analysis, object recognition, functional MRI, Fiedler partitioning, human visual system

1 Introduction

Over the last decade, the landscape of state-of-the-art neural networks has shifted at a near continuous rate. But, within the last few years, the discourse around how to achieve the state-of-the-art has shifted from an emphasis on architecture to an emphasis on robust learned representations. In this work, we note that what constitutes a "good" representation to optimize for is still a matter of debate—we posit that one promising representation to strive for is the one employed by the biological brain. Our contributions include the curation of a new dataset to facilitate measuring the similarity of neural network representations with biological representations, an investigation of the biological fidelity of several state-of-the-art models' representations, and the establishment of a new evaluative benchmark to facilitate further research into aligning artificial and biological neural representations.

Investigations into how similar a trained neural network is to a biological brain have been unfolding since the early days of the neural network boom (Yamins et al., 2013, 2014; Hong et al., 2016; Kheradpisheh et al., 2016; Yamins and DiCarlo, 2016). Prior work has shown that representations closer to the biological brain are more robust to adversarial attacks (Li et al., 2019), are adaptable to new tasks in a zero-shot context (Schrimpf et al., 2018; Blanchard et al., 2019), and have gains in task performance that emerge quicker than when learning representations without biological similarity (Blanchard, 2019; Blanchard et al., 2019). Given this pedigree, one would be remiss not to wonder why research into these comparisons is so rare. Unfortunately, most biological datasets are proprietary or too small (Chang et al., 2019) and without this resource, neither researchers nor practitioners can further investigate this phenomenon. Further, state-of-the-art models have traditionally been assessed by their accuracy on key datasets while evaluations of how well-embedded representations generalize to new tasks is a relatively recent phenomenon (Radford et al., 2021).

Additionally, many of these works simply focus on *post-hoc* evaluations. There are relatively few works investigating how to optimize networks to achieve biological representations (Elsken et al., 2018; Hsu et al., 2018; Liu et al., 2018; Pham et al., 2018; Bashivan et al., 2019). Even recent efforts to learn strong representations focus on unsupervised methods that allow massive amounts of data to be used for training, with the hope that stronger representations will emerge (Radford et al., 2021). We hypothesize that a large biological dataset would facilitate a deeper investigation into the viability of biological representations for artificial neural networks. Of particular interest to this community is the potential for deeper investigations into how to optimize for biologically grounded manifolds—current methodologies utilize representational similarity analysis (RSA) (Kriegeskorte et al., 2008), but recent work has suggested methods to adopt and expand these methods (Jamil et al., 2023) by redefining the core data structures of RSA, representational dissimilarity matrices (RDMs), into weighted graphs.

Following the methodology pioneered by Jamil et al. (2023), we demonstrate that network representations drift further away from biological representations when networks are optimized for task performance. This is in agreement with Kumar et al. (2022), who found an inverse-U relationship exists between ImageNet classification accuracy of a network and its perceptual similarity score (Zhang et al., 2018). We posit that these mirror critiques of prominent research groups like Google's DeepMind, where Goh et al. (2021) identified the viability of an adversarial typographic attack where simply writing the incorrect word on an object sufficed for causing misclassifications. In a blog post discussing the attack, Goh et al. (2021) suggested,

> "this attack exploits the way image classification tasks are constructed. While images may contain several items, only one target label is considered true, and thus the network must learn to detect the most 'salient' item in the frame. The adversarial patch exploits this feature by producing inputs much more salient than objects in the real world. Thus, when attacking object detection or image segmentation models, we expect a targeted toaster patch to be classified as a toaster, and not to affect other portions of the image."

It is true that assessing state-of-the-art models has always been important for both practitioners adapting those models to their own tasks and researchers seeking to understand and push toward better models (Kingma and Welling, 2013); however, the advent of works like CLIP, from Radford et al. (2021), have ushered in a new era driven by evaluating neural networks on how adaptable their learned representations are to new tasks in a zero-shot context. This work provides the tools for researchers to take this idea further providing biologically viable target representations that can be factored into the optimization of networks. As illustrated in Figure 1, our contributions include:

- The presentation and re-release of the BOLD5000 dataset (Chang et al., 2019), which has been fully processed to facilitate evaluating and optimizing neural networks on biologically grounded representations of data. Our efforts culminate in one of the largest biological datasets for vision ever released, which will facilitate widespread investigations into optimal representations.
- The application of a previously unexplored graph-based technique, the Fiedler algorithm, to this preprocessed dataset, demonstrating its versatility as an evaluation metric.
- The introduction of a framework which allows researchers to better fine-tune, evaluate, and select models for robustness. Ultimately, the products of this work will facilitate future research into how robust representations manifest, and methods for optimizing networks to achieve trustworthy and adversarially robust results.

2 Related work

Here, we detail prior works that investigate biological representation benchmarks. In particular, we focus on methods that investigate "neuro-similarity," i.e., the similarity of an artificial neural network's (ANN's) learned representation to a benchmark of the biological brain. First, we examine metrics of neuro-similarity, then, efforts to increase neuro-similarity, and conclude with an investigation of works that link biologically consistent ANNs and robustness.

2.1 Metrics of neuro-similarity

Representational similarity analysis (RSA) is a popular tool measuring neuro-similarity where similarity metrics are derived from representational dissimilarity matrices (RDMs) (Kriegeskorte et al., 2008). ANN activations and neural data can be abstracted into RDMs for a set of stimuli. If two RDMs are created using the same stimuli set, they can be directly compared to one another by measuring the similarity of the consistency across that stimuli set. Two established metrics that capitalize on RSA for measuring

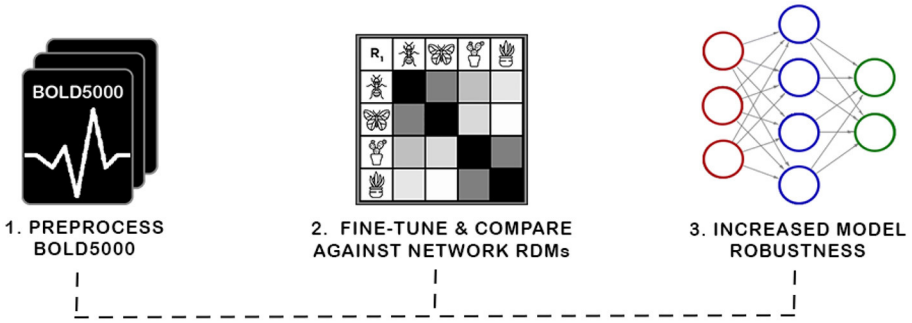


FIGURE 1
In this work, we present a new biologically grounded representation for evaluation and optimization of neural representations. Prior work has shown such representations correspond with robustness to adversarial attacks and task generalization. The curation of this new benchmark required preprocessing the BOLD5000 data into representational dissimilarity matrices (RDMs) and establishing a framework for investigating biological representations. We then investigate the viability of our discovered representation with a novel application of Fiedler partitioning on the data to demonstrate the potential of the biological representation for adversarial robustness.

the neuro-similarity of ANNs are human-model similarity (HMS) (Blanchard et al., 2019) and Brain-Score (Schrimpf et al., 2018).

HMS (Blanchard et al., 2019) evaluates the neuro-similarity between fMRI data and ANNs as the Spearman correlation between the averaged fMRI RDM and an ANN's RDM. The metric was validated on self-supervised predictive coding networks—a form of ANN composed of convolutional long short-term memory (LSTM) units designed to mimic predictive coding employed by biological visual systems. They found that models with higher HMS exhibited higher performance on next-frame prediction (the self-supervised task the networks were trained on) and were more robust to other tasks that networks were not trained for, such as object matching. They also found that HMS could be accurately measured early in the training process, and they proposed that it could be utilized for "early stopping", i.e., training could be abandoned before the weights fully converged.

Similar to HMS, Brain-Score (Schrimpf et al., 2018) is a composite neural and behavioral benchmark set, which uses multiple evaluation metrics to score and rank ANNs according to how brain-like their visual object-recognition mechanisms are. To accomplish this, the internal representations of ANNs trained on ImageNet were compared for similarity against neural recordings taken from the V4 and IT cortical areas of macaque monkeys. From this, Dense-Net169, COREnet-S, and ResNet-101 were found to be the most brain-like, though Brain-Score was unable to reveal why.

HMS is the most similar to our methodology because we too use publicly available fMRI data, but a major limitation of HMS is that it only utilizes 92 stimuli, making it unsuitable to train with since networks quickly overfit to the small sample. These metrics are a great starting point for measuring neural similarity—however, to improve model robustness, more specific metrics need to be created. To effectively achieve this, datasets similar to this one must have as little noise in them as possible, something we address with BOLD5000.

In addition to RSA, there has also been research into psychophysical comparison metrics between ANNs and the human vision system. Jacob et al. (2021) found that ANNs trained for object recognition tasks were susceptible to some of the same visual illusions as the human visual system, such as mirror confusion, while other effects were absent. Gomez-Villa et al. (2020) found that ANNs trained for low-level visual tasks such as denoising and deblurring demonstrate human-like contrast similarity, but noted that deeper, more flexible network architectures did not demonstrate the same similarity. Human-like contrast similarity was also found in a variety of ANN architectures trained for different tasks (Li et al., 2022; Akbarinia et al., 2023).

2.2 Increasing neuro-similarity

Multiple methods have been investigated to affect an increase in the neuro-similarity of ANNs. One approach is the tailoring of image training datasets to achieve a distribution of input stimuli that more closely matches what may be experienced in nature (Aliko et al., 2020; Mehrer et al., 2021; Roads and Love, 2021; Allen et al., 2022). This approach is based on observations that training datasets designed for machine vision applications are crafted for domain specific applications, or otherwise contain internal biases in their distribution of subject matter that do not match what is in nature (Smith and Slone, 2017). A specific example of such a bias is the fact that ImageNet (Deng et al., 2009), one of the most widely used image classification datasets in the field, contains 120 categories of dog breeds, but lacks any categories for humans. By creating datasets with more natural image distributions, researchers have been able to significantly improve the neuro-similarity of the DNNs trained on these datasets.

While this approach does improve neuro-similarity in the trained models and demonstrates the potential of DNNs to achieve higher levels of neuro-similarity, it may not always be feasible or desirable to augment every dataset with a great enough volume of images, or images of the correct type, to achieve a distribution that matches the natural world. For example, domain specific datasets, such as for medical imaging research, don't have a complementary input set in nature to draw from. Datasets for machine vision research are also growing in size constantly and it may not be cost-effective or efficient to increase their size to a point where a natural distribution is achieved. However, these domain-specific models can potentially still benefit from greater neuro-similarity.

One architectural approach to increasing neuro-similarity is divisive normalization, which seeks to replicate how neighboring neurons normalize their activations non-linearly (Miller et al., 2021; Veerabadran et al., 2021; Hernández-Cámara et al., 2023).

It has been demonstrated that DNN models with greater neuro-similarity perform better at some tasks than models with lesser neuro-similarity. One exciting example of this, and the inspiration for this paper was work done by Li et al. (2019), who improved the robustness of a deep convolutional neural network (DCNN) to image noise via fine-tuning with an additional loss function that favored greater neuro-similarity. These experiments were conducted using a dataset derived from two-photon excitation microscopy (2PEF) of mice brains—they released the code to enable the fine-tuning but did not release the data itself. The fine-tuning was enabled via RDM comparisons—however, unlike Brain-Score and HMS, they approximated complete RDMs during training by only creating an RDM for a subset of stimuli. Constructing an entire RDM during training is computationally expensive because activations for each of the stimuli must be collected and compared.

2.3 Linking neuro-similarity to robustness

Despite initial findings that improving neuro-similarity could increase robustness (Li et al., 2019), none of the known evaluation metrics explicitly measure this improvement. We think this is an area where some could be created. We propose that robustness should be measured via psychophysics (RichardWebster et al., 2018, 2019). This evaluation focuses on evaluating robustness across a range of different noise levels. It also focuses on explainable and trustworthy evaluations of networks—by exploring a multitude of different noise types, the evaluation reveals specific weaknesses that networks are susceptible to, e.g., in the domain of face recognition, RichardWebster et al. (2018) found that FaceNet was surprisingly susceptible to brown noise, while other methods were not.

3 Materials and methods

3.1 BOLD5000

BOLD5000, one of the largest, publicly available fMRI datasets, was created to address three areas of neural dataset design: create a dataset of sufficient size to enable fine-tuning a deep neural network (DNN), have a greater diversity of images and image categories than is normally present in a neural study, and provide an overlap between the stimulus images used in the fMRI trials and the training image datasets of DNNs to allow for a more direct comparison of DNN and human brain activations (Chang et al., 2019).

Similar to most other human fMRI brain scan datasets, BOLD5000 is composed of stimuli images pulled from existing machine vision image datasets (Geirhos et al., 2018; Allen et al., 2022). In total, it consists of fMRI brain scans from four participants (CSI1-4) who were presented with 4,916 real-world images from three commonly used computer vision datasets: 1,916 from ImageNet (Deng et al., 2009), 2,000 from Common Objects in Context (COCO) (Lin et al., 2014), and 1,000 custom images of scenes from categories inspired by Scene UNderstanding (SUN) (Xiao et al., 2010). Collectively, these datasets span a wide variety of categories and consist of images of real-world indoor and outdoor scenes and objects either centered in or interacting with complex real-world scenes.

All selected images were resized, cropped to 375 × 375, and adjusted for even luminance. For each input dataset, exemplar images were hand-selected by the BOLD5000 authors on a per-category basis. Subjects then engaged in 15 functional MRI sessions, where all images were presented on a single trial basis, except for a subset of 113, for which unique neural representation data was collected.

During the original BOLD5000 study, one participant (CSI4) did not complete the entire experiment. As a result, CSI4 is typically discarded from studies using the BOLD5000 (Sexton and Love, 2022). However, because there are already only three complete participants to begin with, and because the majority of the stimuli images are only presented once, this study incorporates CSI4's partial data into a mean subject using the RDMs calculated as part of the RSA analysis (Section 3.4).

A second release of the BOLD5000 dataset occurred in 2021 (Chang et al., 2021). The major difference with the second release was the re-processing of the beta values for the fMRI sessions using the GLMSingle toolbox (Prince et al., 2022). The goal of the second release was to increase the reliability of the beta value estimates.

3.2 Preprocessing

All betas were provided in NIfTI format, divided by subject and session. The image coordinate transforms provided within the file header did not correspond to the transforms used for brainmasks, ROI masks, and T1w anatomical images from the original BOLD5000 release. This transform information is required for several other processing steps, including the re-application of the functional region of interest (ROI) masks from the original release of the BOLD5000 and application of the two new ROI atlases, vcAtlas (Rosenke et al., 2018) and visfAtlas (Rosenke et al., 2021), to the four participant brains. We solved this issue by intuiting that the provided NIfTI files were derived from the same fMRIPrep (Esteban et al., 2019) derivatives as the original BOLD5000, thus allowing us to utilize the same alignments and brainmasks. The affine transforms from the original BOLD5000 brainmasks were applied to the GLMSingle beta files and the results were visually checked against both the original brainmasks and the T1w anatomical scans of the participants to confirm good alignment. The generation of a global brainmask intersection was also required for each of the four subjects across all sessions. RSA analysis calculates distance metrics for each pair of input stimuli and therefore requires that the input vectors for each of the stimuli have the same

number of dimensions (in the case of fMRI, dimension is a voxel). The BOLD5000 is somewhat unique in that it is largely made up of single presentations of each stimulus, and the order of the stimuli is randomized across multiple sessions for each participant. This poses a challenge because even very minor positional changes between sessions can lead to the introduction of invalid voxel values, especially around the very edge or pial surface of the brain.

The fMRIPrep pipeline uses a number of advanced tools to correct for any changes (Esteban et al., 2019), however, it was found that the participant brain masks provided in the original BOLD5000 release still resulted in invalid voxels being included for some trials. To address this issue, a global mask was calculated for each participant using the intersection of the valid voxels for each input across all sessions. These global participant brain masks were applied to every ROI to ensure that no invalid voxel data entered into the RSA calculations.

3.3 FreeSurfer

FreeSurfer is an incredibly powerful suite of tools originally developed with the goal of reconstructing cortical surface models from T1w anatomical scans (Fischl, 2012). A further goal of this original development in reconstructing the cortical surface is finding alignments between subject brains based on cortical folding patterns. It is this alignment functionality that makes the FreeSurfer a vital component of the fMRIPrep (Esteban et al., 2019) pipeline used in the original BOLD5000 release (Chang et al., 2019).

As follow-on researchers, we leverage these FreeSurfer derivatives to extract additional information from the dataset. We use FreeSurfer to parcellate a reconstructed cortical surface based on its folding patterns using specially crafted atlases. We used this functionality to identify and extract additional areas relevant to vision based on structural connectivity or functional response to images using vcAtlas and visfAtlas respectively. Our analysis is concerned with comparing the BOLD activations of voxels in volumetric space. Thus, several steps were required to convert these surface atlases into volumetric ROI masks.

First, the labels from the atlases were resampled from the standard fsaverage surface to each of the subjects' cortical surfaces. This is accomplished using the mri_surf2surf command. With the labels for each atlas and ROI now resampled onto the subjects' cortical surfaces, the labels were used to define a volumetric ROI as the volume of gray matter that makes up the cortex beneath the cortical surface label. This is accomplished with the mri_label2vol command with projection fraction set to include 100% of the volume between the pial and white matter surfaces. The output of this function is a series of volumetric ROI masks in NIfTI format, similar to the ROI masks from the original BOLD5000. All ROI masks generated using FreeSurfer also had the global mask for each participant applied to them to ensure that only valid voxels would be extracted for a given ROI.

3.4 RSA

After preprocessing and utilizing FreeSurfer to identify ROIs, we create RDMs from the neural data. We construct RDMs in accordance with established methodology (Kriegeskorte et al., 2008; Blanchard et al., 2019). Here, we briefly summarize the process:

RDM construction. Given a single feature f and a single stimulus s, $v = f(s)$, where v is the value of feature f in response to s. Likewise, the vector

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}^T = \begin{bmatrix} f_1(s) \\ f_2(s) \\ \vdots \\ f_n(s) \end{bmatrix}^T \tag{1}$$

can represent the feature values of a collection of n features, f_1, f_2, ..., f_n, in response to s. If one expands the representation of s to a set of m stimuli $S = s_1, s_2, ..., s_m$, the natural extension of $\vec{v}$ is the set of feature value collections $V = \vec{v}_1, \vec{v}_2, ..., \vec{v}_m$, in which $s_i \in S$ is paired with $\vec{v}_i \in V$ for each $i = 1, 2, ..., m$. The last step prior to constructing an RDM is to define the dissimilarity score between any two $\vec{v}_i \in V$ and $\vec{v}_j \in V$. We use the symmetric function

$$\psi(\vec{v}_i, \vec{v}_j) := 1 - \frac{(\vec{v}_i - \bar{v}_i) \cdot (\vec{v}_j - \bar{v}_j)}{\|\vec{v}_i - \bar{v}_i\|_2 \|\vec{v}_j - \bar{v}_j\|_2} \tag{2}$$

where $\bar{v}$ is the mean of the features in $\vec{v}$. An RDM R may then be constructed from S, V, and ψ as:

$$R = \begin{bmatrix} \psi(\vec{v}_1, \vec{v}_2) & \psi(\vec{v}_1, \vec{v}_3) & \dots & \psi(\vec{v}_1, \vec{v}_m) \\ & \psi(\vec{v}_2, \vec{v}_3) & \dots & \psi(\vec{v}_2, \vec{v}_m) \\ & & \ddots & \vdots \\ & & & \psi(\vec{v}_{m-1}, \vec{v}_m) \end{bmatrix} \tag{3}$$

3.4.1 Biological similarity metric

The methodology for comparing a network to a biologically constructed RDM is simple: After constructing an RDM R_1 for the network following the procedure outlined in Section 3.4 using the same stimuli set S, one can compute the similarity to the biological RDM R_2 with the function

$$biologicalSimilarity = \rho(\hat{R}_1, \hat{R}_2) \tag{4}$$

where $\hat{R}$ is the flattened RDM and ρ corresponds with a similarity metric, e.g., Pearson's correlation. Note, many works suggest estimating the RDM during training by only considering a subset of the stimuli (Li et al., 2019).

3.4.2 rsatoolbox package

rsatoolbox is a Python package for RSA (Nili et al., 2014). Originally developed for Matlab, rsatoolbox is under

TABLE 1 Five supercategories were created by combining the synset labels from the ImageNet stimuli.

Supercategory	Hypernyms	Num. images
Vertebrate	[animal, person]	646
Invertebrate	[invertebrate]	96
Natural object	[food, plant, fungus, plant_part]	128
Artifact	[artifact]	912
Place	[structure, geological_formation]	134

active development and can be used for the generation and comparison of RDMs, the creation and evaluation of multiple types of models with various statistical tools, and visualization tools. All fMRI RDMs, RDM comparisons, and models were performed with rsatoolbox (Initial RDM generation for the ANNs were generated using functionality built into the Net2Brain tool as detailed in Section 3.7.).

3.5 ImageNet in BOLD5000

The use of images from the ImageNet dataset in the BOLD5000 presents a unique opportunity because the ImageNet Large Scale Visual Recognition Challenge (ILSVC) benchmark remains the standard benchmark and training dataset for image classification models such as those included in this paper. Prior to the BOLD5000 data, representations of neurological data tended to be collected for simple stripped-back stimuli such as a clearly cut-out image against a gray background. While these simple stimuli enabled research comparing biological representations to artificial representations (e.g., Blanchard et al., 2019), they had limited additional uses. For example, these stimuli were too simple and too few for fine-tuning networks to exhibit biologically consistent embeddings. The use of complex images like those within the ImageNet dataset may be non-ideal for traditional fMRI research, but they enable a wealth of experiments examining artificial neural networks (ANNs).

ImageNet classes are based on the WordNet synset hierarchy. In theory, this synset hierarchy can be used to establish the relationships between image classes. However, there are known deficiencies in the WordNet structure and most researchers resort to creating custom "supercategories" that combine multiple synsets. The original BOLD5000 paper used four supercategories for t-SNE analysis: Objects, Food, Living Inanimate, and Living Animate (Chang et al., 2019). For this work, five new supercategories were chosen that attempt to create more logical pairings for comparison. Animate subjects were divided into "vertebrate" and "invertebrate" supercategories, inanimate classes were divided into "artifact" (man-made) and "natural object", and a final "place" category was included to align with the emphasis on scenes in the BOLD5000. Table 1 summarizes the supercategories created for this project and the hypernyms used to define each supercategory. Each of the ImageNet synset labels were sorted into a supercategory by matching the synset's hypernyms to one of the supercategory hypernyms.

3.6 Categorical model analysis

While the end goal of our RSA analysis is to compare the biological data from the BOLD5000 fMRI trials to ANNs, RSA also allows us to leverage other types of dissimilarity models such as the supercategories within ImageNet as described in Section 3.5. First, categorical RDMs are generated for each supercategory as illustrated in Figure 2. These consist of an RDM where all images of the same category are assigned the minimum distance/dissimilarity for a given metric and all images from other categories are assigned the maximum distance/dissimilarity for a given metric.

Using rsatoolbox's weighted model functionality, the individual category RDMs are linearly fit to the Mean Subject RDMs for the vcAtlas ROIs. The model weights were then used to predict the final categorical model shown in Figure 3. This categorical model is a representation of the relative similarities of each of the supercategories as perceived by the human brain. Categorical models such as this can act as a reference point for later RSA analysis because it relies on additional structural information that is embedded into the ImageNet image labels.

3.7 Net2Brain

Here, we link our preprocessed data and subsequent evaluations to Net2Brain, a toolbox for researching the internal geometric representations of artificial deep neural networks, particularly convolution neural networks, using RSA. One of the strengths of Net2Brain is the very extensive set of over 600 models that it is preconfigured to pull down, extract activations from, and calculate RDMs for. Net2Brain is able to pull models not only from the official PyTorch model zoo, but also from timm, the Pytorch Image Models library created by Ross Wightman. All of the aforementioned 600+ models available to Net2Brain come pre-trained and are fully ready for activation extraction. All of the stimuli from the BOLD5000 are made available to Net2Brain and once it pulls down the pre-trained model in question, it presents each of the BOLD5000 images to the model as input and performs a forward pass. The model activations from each of the model's convolutional layers are then extracted and stored to disk. Once all of the activations have been extracted, RDMs for each of the convolutional layers are calculated. As of the time of writing, the toolbox enables creating RDMs using Pearson's correlation, and there are plans to add various other distance metrics.

3.7.1 Model selection

Of the over 600 models available, four were chosen based on a couple of criteria. First, due to the limitations in the architecture of both Net2Brain and rsatoolbox, the calculation of RDMs required substantial amounts of memory given the number of unique stimuli in the BOLD5000. There was, therefore a relative size limit to the number of output activations in a model given the memory limits of available hardware. The

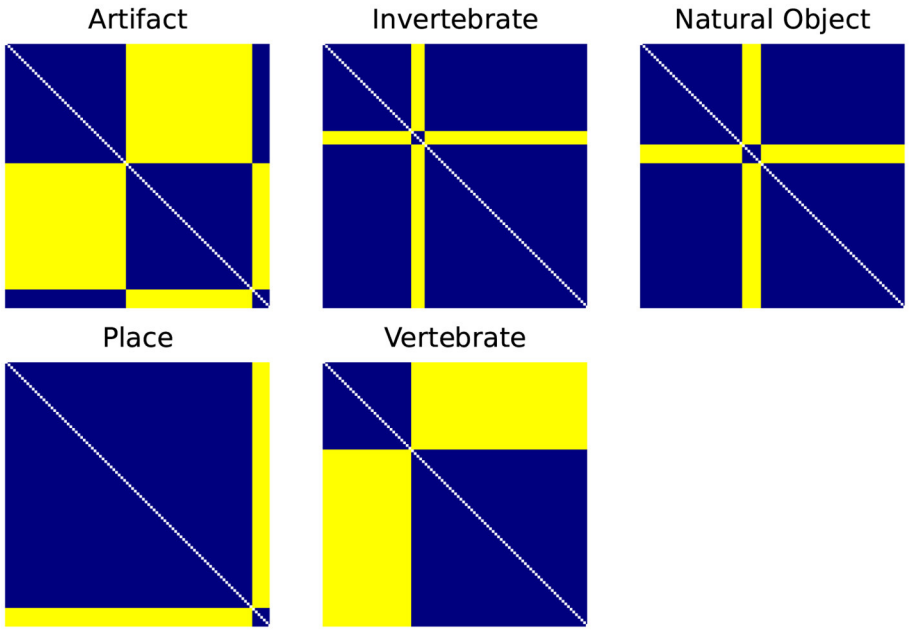


FIGURE 2
Categorical RDMS for each of the ImageNet supercategories. Categorical RDMs consist of an RDM where all images of the same category are assigned the minimum distance/dissimilarity for a given metric (i.e., for the 1-r distance metric, 0) and all images from other categories are assigned the maximum distance/dissimilarity for a given metric (i.e., 1).

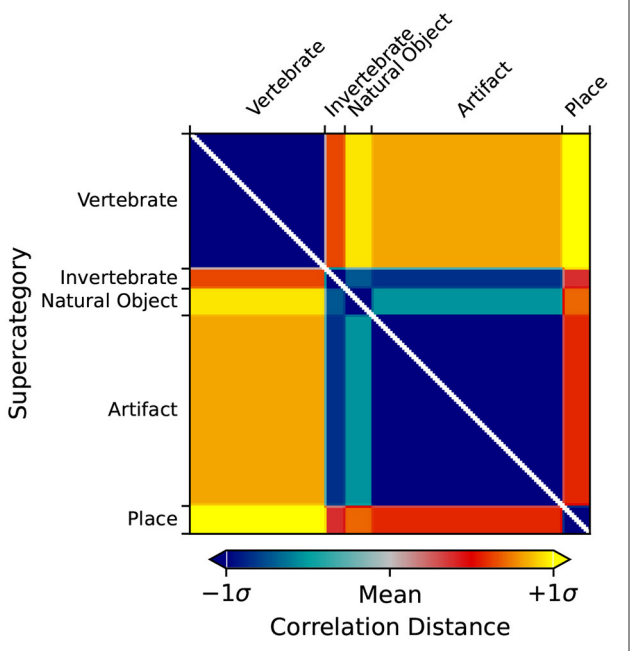


FIGURE 3
Predicted output RDM of the weighted categorical model.

second criterion was to achieve a representative sampling of ANN model architectures that are designed for image classification tasks and pre-trained on the ImageNet dataset over time. The four models chosen were: AlexNet (Krizhevsky et al., 2012), the progenitor of all subsequent deep convolutional neural networks, ResNet50 (He et al., 2016), which introduced skip connections to neural network architectures, MobileNetv2 (Sandler et al., 2018), which was specifically designed to perform well even on restricted hardware such as mobile devices, and finally EfficientNet (Pham et al., 2018), which expands on the same architectural concepts present in MobilNet with efficient network scaling.

3.8 Fiedler vector partitioning

In this section, we detail how we employ Fiedler partitioning, a graph-based technique, on the processed data. Fiedler partitioning aims to partition a graph into two distinct groups by utilizing the Fiedler vector, which corresponds to the second smallest eigenvector of the Laplacian (Fiedler, 1973, 1975).

We analyzed individual RDMs for three BOLD5000 participants (CSI1-3), and a mean RDM (averaged subject data) for fMRI data specific to the Left-Hand Fusiform Gyrus 3 (LHFG3). Each RDM is composed of the supercategories described in Table 1. From these supercategories, we first extracted subsets of each class pairing. We then applied Fiedler partitioning to these RDMs and recorded the classification accuracy for each class in a pair. The pseudo code for finding the Fiedler partitioning accuracy for an RDM is detailed in Algorithm 1. Bias corrected and accelerated (BCa) bootstrap intervals were calculated for each binary classification.

4 Results

4.1 Categorical model analysis

Each of the vcAtlas ROIs from the mean subject was compared back against the predicted categorical model to determine which ROI best represents the supercategorical structure of the data. Figure 4 shows the correlation of each of the ROIs to the categorical model. In the case of the BOLD5000 data, the LHFG3 is the best exemplar of the categorical model.

Figure 5 shows the RDM for LHFG3 from the mean subject with the images sorted by supercategory. Comparing LHFG3 Figure 5 to the categorical model Figure 3, is it clear how the supercategory representations cluster in a similar way. This can be explored further through the comparison of RDM correlations.

Figure 6 shows the RDMs of the layer with the highest correlation to the categorical model for each of the four ANNs investigated. When visually comparing the categorical model, Figure 3, the mean subject fMRI response, Figure 5 and the ANN responses, Figure 6, a correspondence between the representation of the supercategories is evident.

Require: Representational Dissimilarity Matrix R
Ensure: Classification Accuracy
1) Get a subset R_i of R with two categories.
2) Compute Adjacency Matrix $A = 1 - R_i$.
3) Compute Degree matrix from A.
4) Compute Laplacian matrix: $L = D - A$.
5) Get second smallest eigen vector e_2 for L.
6) Compute Fiedler partitioning: $P_1 = \{i \in N : e_2(i) < 0\}$ and $P_2 = \{i \in N : e_2(i) > 0\}$.
7) Compute Accuracy = $(|P_1| + |P_2|)/len(e_2)$

Algorithm 1. Fiedler partitioning classifier.

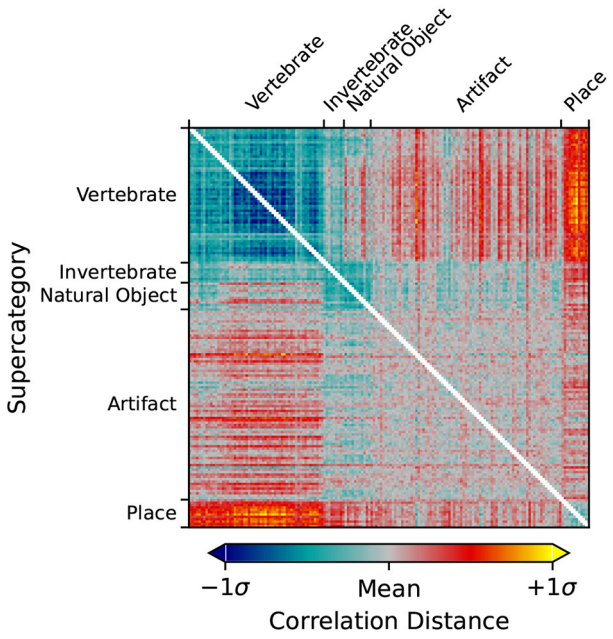


FIGURE 5
RDM of the Left-Hand Fusiform Gyrus 3 (LHFG3) ROI calculated from the mean subject using the correlation distance metric. Image inputs are sorted by their ImageNet supercategory. Clustering of similar images within supercategories is visible, as is dissimilarity between supercategories.

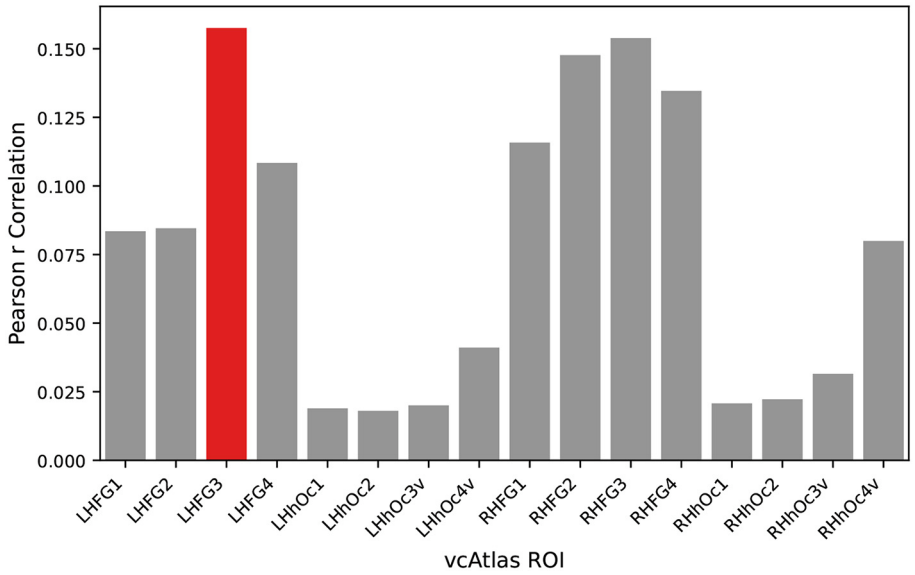


FIGURE 4
An exemplar ROI is chosen from the available vcAtlas ROIs by comparing its Pearson correlation to the categorical model (Figure 3). The Left-Hand Fusiform Gyrus 3 (LHFG3) (highlighted in red), was found to have the highest correlation with the categorical model.

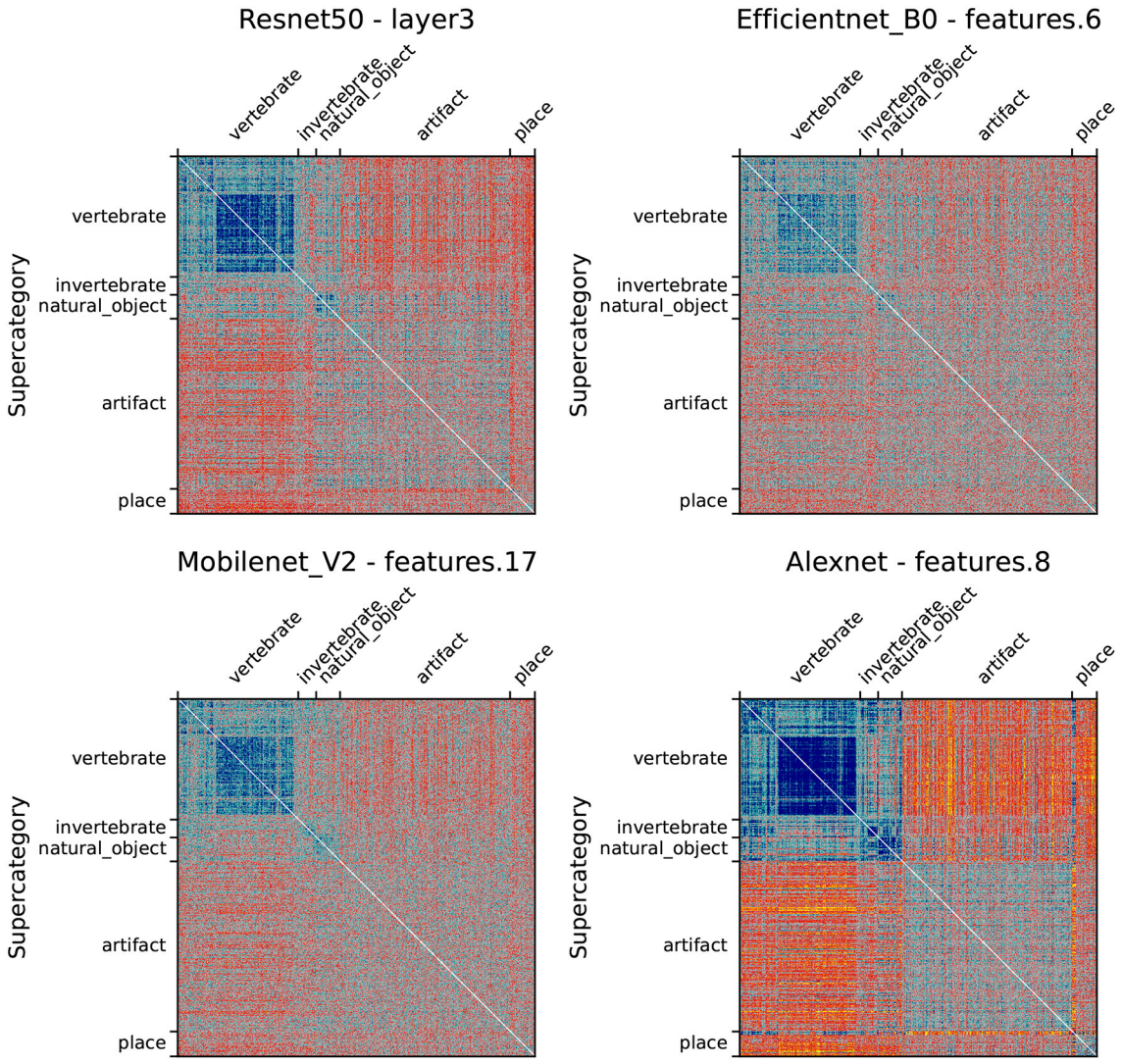


FIGURE 6
RDMs from each of the four ANNs ordered by ImageNet supercategory. Each RDM is taken from the ANN layer with the highest correlation to the categorical model calculated in Section 3.6.

4.1.1 ANN vs. human fMRI RDM comparison

Direct comparison of RDMs can be accomplished through a number of similarity measures. Here, we report Pearson correlation, an established standard for use in RSA (Kriegeskorte et al., 2008). Table 2 presents the Pearson correlation between the categorical model and each of the four ANNs under test. Bootstrap resampling of the input stimuli was performed to assess that all results were statistically significant ($p < 0.001$). All comparisons and statistical evaluations were performed using the rsatoolbox package (Section 3.4.2).

An unexpected result of this analysis is the inverse relationship between model age and its biological similarity. AlexNet (Krizhevsky et al., 2012), the model that kicked off the deep convolutional neural network revolution in machine vision, has the highest biological similarity of the models tested, and EfficientNet (Tan and Le, 2019), the most modern and highest performing classification model, has by far the lowest biological similarity.

4.1.2 Comparing human fMRI ROIs to individual ANN layers

In Figure 7, we break down our evaluation layer by layer in order to provide fine-grained details on which components of the trained network best exhibits biological similarity.

One of the goals in reprocessing the BOLD5000 dataset using the vcAtlas (Rosenke et al., 2018) and visfAtlas (Rosenke et al., 2021) maps was to enable future research into comparing how various components of an ANN, such as individual convolutional layers, can be compared to specialized structures in biological representations. For example, consider the theoretical concept behind the ventral visual stream in the human brain is that visual information flows from the early visual cortex at the back of the brain forward into the Fusiform Gyrus. Along the way, the visual stimuli is decoded in increasingly higher order representations. Our findings give credence to the observation that deep convolutional neural networks mimic some of what occurs with this process.

TABLE 2 Comparison of mean subject LHFG3 ROI RDM to categorical model and ANN RDMs with bootstraped *p*-values.

Model	Pearson correlation	*p* (against 0)
Categorical	0.165	<0.001
AlexNet	0.054	<0.001
MobileNet v2	0.023	<0.001
ResNet50	0.031	<0.001
EfficientNet b0	0.015	<0.001

The human brain also has a number of very specialized areas for certain tasks such as facial recognition in the Fusiform Face Area (FFA) (Kanwisher et al., 1997), one of the ROIs included in the visfAtlas. The goal is to provide the data so that these specialized areas of the brain can be used to analyze and train equivalently specialized components of ANNs.

4.2 Fiedler partitioning

Figure 8 displays the Fiedler partitioning accuracies for each of the four ANNs from our experiments, and Figure 9 shows the partition accuracy for the biological data. All accuracies illustrate the separability of class pairs—the results indicate that the human subjects consistently achieved higher classification accuracy when discriminating between the vertebrate class and the invertebrate, natural object, and place categories. This shows that the feature embeddings in the LHFG3 are well-clustered for those categories.

Bonferroni-corrected BCa confidence intervals are also reported in Figures 8, 9. Because Fiedler partitioning is applied against supercategory pairs, the null hypothesis corresponds to an accuracy of 0.50. An interesting phenomenon that can be seen in the results is that even if the Fiedler partitioning achieves high levels of accuracy, we may not be able to reject the null hypothesis due to the lower bootstrap confidence interval being at or near 0.50. This can be seen both in the fMRI subject results and the ANN results. Analysis of the bootstrap sample distribution shows this is sometimes the result of a bimodal distribution of results, where a small cluster of failed partitions will be present at or near 0.5 accuracy, with the rest scoring much higher. This effect is likely due to a combination of label noise in the BOLD5000 dataset and how the Fiedler vector is calculated.

Label noise arises when the two ImageNet images selected for each synset category contains more than just the intended subject, or when other factors, such as the crop of the image that was used for presentation obscure or make the intended subject unclear. An example of this is the fact that many of the stimuli images in the BOLD5000 which fall under the supercategory "artifact" depict people holding the object with a full human face visible in the image. The presence of human faces in images that are meant to depict inanimate objects causes significant unintended brain activation in areas such as the FFA (Kanwisher et al., 1997).

The second factor contributing to this phenomenon is the fact that Fiedler partitioning is not an operation that is performed on individual stimuli, but on the RDM as a whole. The Fiedler vector is calculated as the second smallest eigenvector of the Laplacian of the RDM. Therefore, if a bootstrap sample is composed of a set of images with sufficient label noise, the Fiedler vector will partition the entire RDM orthogonal to the intended supercategories. An example of this was found with the above "artifact" supercategory example. When applied solely to the stimuli images from the "artifact" supercategory in an unsupervised manner, the Fiedler partitioning algorithm spontaneously partitions the images into a group that contains humans in the image and a group that contains just inanimate objects in the frame.

Overall, our findings indicate that Fiedler partitioning effectively identifies supercategory clusters in the RDMs of human fMRI subjects and ANNs. Similar to our findings with the RDM comparisons, a surprising trend emerges with the ANNs. AlexNet, the oldest of the ANNs, produces a far higher Fiedler partitioning accuracy than the newer models. EfficientNet-b0, in particular, does not produce results significantly above noise for most of the supercategory pairings.

5 Discussion and future work

ANNs have long suffered from decreased performance as a result of their sensitivity to random noise and adversarial attacks. Recent works have shown that fine-tuning a network representation to align with a biological standard fortifies networks against both noise and adversarial corruptions of images (Blanchard et al., 2019; Li et al., 2019). However, exploration of these ideas has been limited by the unavailability of public datasets: prior works have relied on private datasets (Li et al., 2019) or datasets with a limited number of stimuli (Blanchard et al., 2019). The BOLD5000 dataset has always been a promising resource for investigating just this, but the data was not packaged for use by researchers without a strong neuroscience background. Here, we eliminate this barrier—our curation and investigations of the BOLD5000 data will now enable the broader community to explore the viability of biological representation in networks.

An important result of our analysis is that recent, more advanced, neural networks such as EfficientNet (Tan and Le, 2019) have lower neuro-similarity to human fMRI responses than the much older and simpler AlexNet, despite also performing much better on the standard ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

The discovery that ANNs are diverging from their biological inspiration is not, in and of itself, surprising and is supported by other recent research (Gomez-Villa et al., 2020; Kumar et al., 2022). It does emphasize the fundamental question of whether or not neuro-similarity is an asset, a hindrance, or simply a non-factor. Are these newer models performing better on an, admittedly artificial, metric because of their neuro-dissimilarity or in spite of it? Humans are not susceptible to the same adversarial attacks that ANNs have been shown to be susceptible, so it's possible this divergence in the geometry of ANN embedding spaces from their human counterparts is opening up new avenues of attack.

To put a finer point on it, are more advanced models achieving higher accuracy by focusing on minutia instead of the complete composition, e.g., the features being extracted from an image of one of ImageNet's many dog breed classes focused on things like fur texture and color as a way to correctly classify the breed, the source

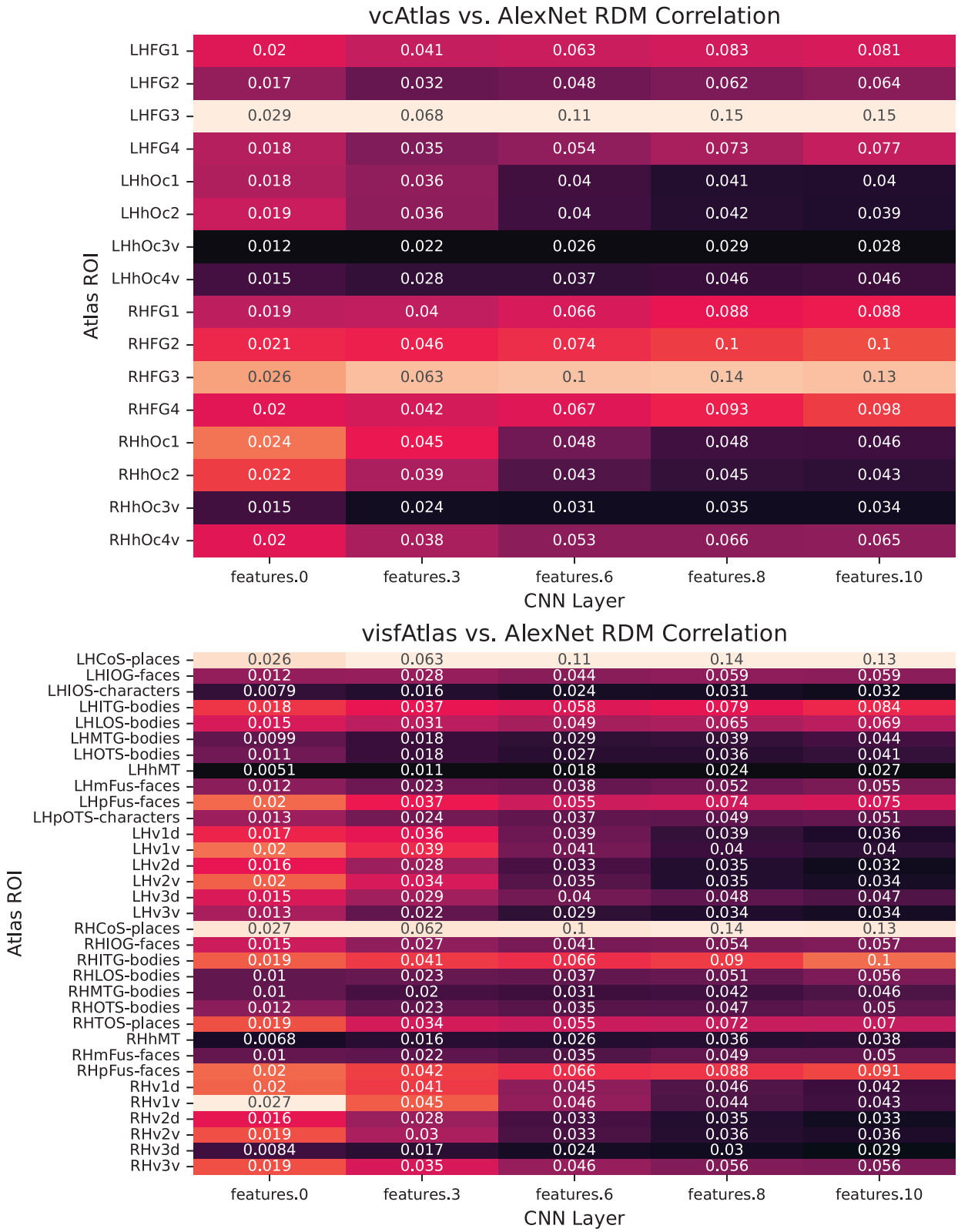


FIGURE 7
Pairwise analysis of each of the layers of AlexNet to ROIs of the new vcAtlas and visfAtlas cortical atlases. In the vcAtlas comparison it can be seen that while the LHFG3 ROI does dominate the comparison, there is a correlation between the first two layers of AlexNet with the early visual cortex in Oc1 and Oc2.

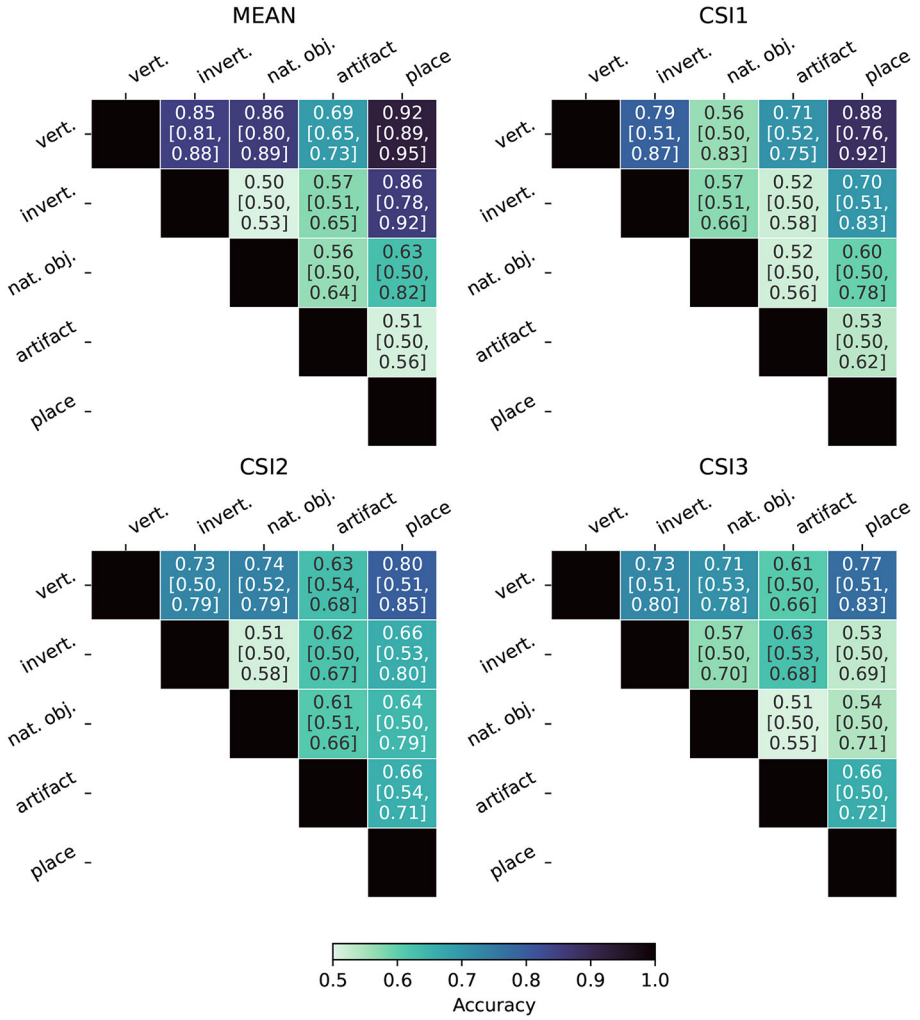


FIGURE 8
Fiedler partitioning accuracy for ANNs. Partitioning is performed on the RDM of the layer with the highest correlation to the categorical model developed in Section 3.6. Accuracy is reported along with Bonferroni-corrected BCa bootstrap intervals (CIs) (accuracy [lower CI bound, upper CI bound]). AlexNet and ResNet demonstrate significantly higher accuracy and significance across all supercategory comparisons.

of the image classification, or on the fact that the image depicts a four-legged creature with two eyes and other mammalian features? Having a fragmented embedding space that emphasizes minutia is likely to make a model more susceptible to adversarial attacks. To use the example above, a model that has been overfit to the point where it only focuses on fur pattern features to identify something as a dog, could be tricked into misidentifying a common artifact such as a box, by covering it with a fake fur texture or image.

We expect images containing similar features to elicit activations that are closer together within the embedding space while dissimilar activations should exist further apart—Goh et al. (2021) investigated the presence of this phenomenon, finding certain neuron groups in CLIP activated or deactivated in response to similar concepts. Fiedler partitioning of an RDM should be able to exploit this clustering of like embeddings to get us in the ballpark of a reasonable classification by selecting an appropriate class category regardless of whether or not there is a strong correlation between the ANN and the biological benchmark. By demonstrating that this works well for a model like AlexNet, but not for a model like EfficientNet, we imply that these more advanced models are not creating the expected clusters within their embedding space. This leads to the question of how these new ANNs are actually structuring their feature space or whether they are extracting a similar set of features at all. Our work shows that ANNs trained for classification performance are evolving internal embedding space geometries more dissimilar from the human vision system and that these embedding spaces lack a geometry that clusters like image subjects together. We can either conclude that state-of-the-art ANNs are creating a novel way to learn and store image feature representations, or we must conclude that embedding spaces are becoming more disjoint because of the singular push to maximize classification accuracy.

Since learned representations like (Goh et al., 2021) do seem to demonstrate this phenomenon with CLIP embeddings, and since CLIP embeddings match or surpass the performance of the models we evaluate (Radford et al., 2021), it seems likely that

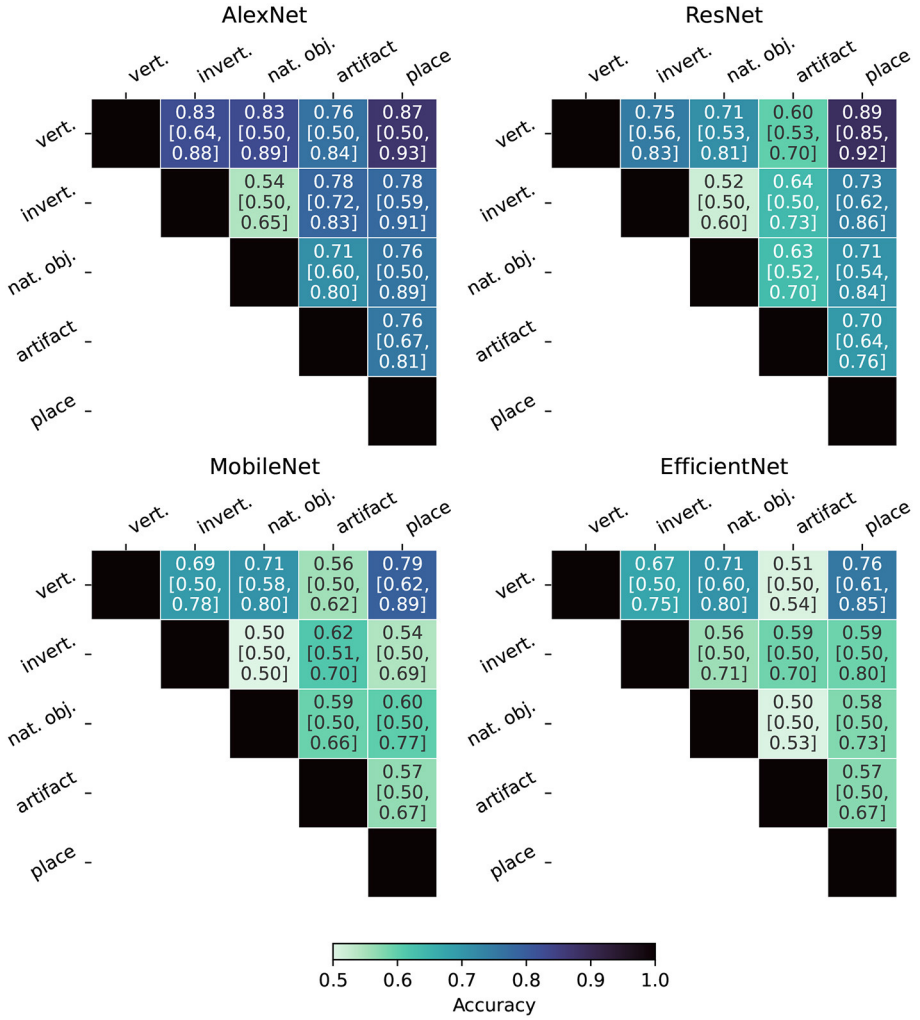


FIGURE 9
Fiedler partitioning accuracy for fMRI subjects. Partitioning is performed on the RDM of the LHFG3 ROI. Accuracy is reported along with Bonferroni-corrected BCa bootstrap intervals (CIs) (accuracy [lower CI bound, upper CI bound]). The mean subject demonstrates both higher accuracy and statistical significance when compared to each of the three complete participants.

the biological ideal does correspond with robustness. However, a full investigation of the viability of the biological benchmark is beyond the scope of this work—and likely beyond the scope of any singular work. Instead, a wealth of future research is needed to tease out the intricacies of what kinds of representations correspond with robustness. The most impactful outcome of this work is the facilitation of these future research projects via a shared, publicly available dataset that allows researchers and practitioners to scrutinize the evidence for a biologically grounded representation, and investigate alternatives.

Finally, the curation of this data also facilitates additional uses of the data: modeling neural processes and creating new biologically consistent architectures. Neural networks are the premier means for modeling neural data. However, it has also been shown that current architectures have largely plateaued (Storrs et al., 2021) and that all networks are equally predictive of the human inferior temporal cortex. This is problematic because these models still fail to predict certain properties of visual processing (Storrs et al., 2021). Our data could facilitate the creation of neural network designs that are biologically grounded. Previously, work has shown that networks deliberately modeled on neural phenomena exhibit higher biological consistency than traditional CNNs (Blanchard, 2019), which corresponds with higher performance. However, even this work would vastly benefit from expanding methods for comparing with biological benchmarks via novel techniques like extending RDMs into Laplacian matrices (Jamil et al., 2023).

6 Conclusion

Here, we establish a new biological benchmark for embedded representations. Our experiments on our benchmark establish the viability of utilizing this data to enhance the robustness of learned representations to inputs like adversarial attacks. Specifically, our experiments with Fiedler partitioning showcase how biologically

grounded representations facilitate interwoven separability and clustering of data. As part of this work, we release our curated data and a framework to facilitate further investigation.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://doi.org/10.5061/dryad.wpzgmsbtr.

Author contributions

WP: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. KS: Investigation, Writing – original draft, Visualization. HJ: Investigation, Methodology, Software, Visualization, Writing – original draft. NC: Software, Visualization, Writing – original draft. NB: Conceptualization, Investigation, Project administration, Software, Supervision, Writing – original draft. MK: Supervision, Writing – review & editing. CP: Supervision, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. The work was supported by Defense Advanced Research Projects Agency (DARPA) HR00112290074.

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.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Akbarinia, A., Morgenstern, Y., and Gegenfurtner, K. R. (2023). Contrast sensitivity function in deep networks. *Neural Netw.* 164, 228–244. doi: 10.1016/j.neunet.2023.04.032

Aliko, S., Huang, J., Gheorghiu, F., Meliss, S., and Skipper, J. I. (2020). A naturalistic neuroimaging database for understanding the brain using ecological stimuli. *Sci. Data* 7, 347. doi: 10.1038/s41597-020-00680-2

Allen, E. J., St-Yves, G., Wu, Y., Breedlove, J. L., Prince, J. S., Dowdle, L. T., et al. (2022). A massive 7T fMRI dataset to bridge cognitive neuroscience and artificial intelligence. *Nat. Neurosci.* 25, 116–126. doi: 10.1038/s41593-021-00962-x

Bashivan, P., Tensen, M., and Dicarlo, J. (2019). "Teacher guided architecture search," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (Seoul), 5319–5328. doi: 10.1109/ICCV.2019.00542

Blanchard, N., Kinnison, J., Richard Webster, B., Bashivan, P., and Scheirer, W. J. (2019). "A neurobiological evaluation metric for neural network model search," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA), 5399–5408. doi: 10.1109/CVPR.2019.00555

Blanchard, N. T. (2019). *Quantifying internal representation for use in model search* (Ph.D. thesis). University of Notre Dame, Notre Dame, IN, United States.

Chang, N., Pyles, J., Prince, J., Tarr, M., and Aminoff, E. (2021). *BOLD5000 Release 2.0*. Carnegie Mellon University. doi: 10.1184/R1/14456124. Available online at: https://kilthub.cmu.edu/articles/dataset/BOLD5000_Release_2_0/14456124/2

Chang, N., Pyles, J. A., Marcus, A., Gupta, A., Tarr, M. J., and Aminoff, E. M. (2019). BOLD5000, a public fMRI dataset while viewing 5000 visual images. *Sci. Data* 6, 49. doi: 10.1038/s41597-019-0052-3

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). "ImageNet: a large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (Miami, FL), 248–255. doi: 10.1109/CVPR.2009.5206848

Elsken, T., Metzen, J. H., and Hutter, F. (2018). Neural architecture search: a survey. *J. Mach. Learn. Res.* 20, 1997–2017. doi: 10.48550/arXiv.1808.05377

Esteban, O., Markiewicz, C. J., Blair, R. W., Moodie, C. A., Isik, A. I., Erramuzpe, A., et al. (2019). fMRIPrep: a robust preprocessing pipeline for functional MRI. *Nat. Methods* 16, 111–116. doi: 10.1038/s41592-018-0235-4

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Math. J.* 23, 298–305.

Fiedler, M. (1975). A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.* 25, 619–633.

Fischl, B. (2012). FreeSurfer. *Neuroimage* 62, 774–781. doi: 10.1016/j.neuroimage.2012.01.021

Geirhos, R., Temme, C. R. M., Rauber, J., Schütt, H. H., Bethge, M., and Wichmann, F. A. (2018). "Generalisation in humans and deep neural networks," in *Advances in Neural Information Processing Systems*, eds S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc.).

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., et al. (2021). Multimodal neurons in artificial neural networks. *Distill.* 6, e30. doi: 10.23915/distill.00030

Gomez-Villa, A., Martín, A., Vazquez-Corral, J., Bertalmío, M., and Malo, J. (2020). Color illusions also deceive CNNs for low-level vision tasks: analysis and implications. *Vision Res.* 176, 156–174. doi: 10.1016/j.visres.2020.07.010

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 770–778.

Hernández-Cámara, P., Vila-Tomás, J., Laparra, V., and Malo, J. (2023). Neural networks with divisive normalization for image segmentation. *Pattern Recogn. Lett.* 173, 64–71. doi: 10.1016/j.patrec.2023.07.017

Hong, H., Yamins, D. L. K., Majaj, N. J., and DiCarlo, J. J. (2016). Explicit information for category-orthogonal object properties increases along the ventral stream. *Nat. Neurosci.* 19, 613–622. doi: 10.1038/nn.4247

Hsu, C.-H., Chang, S.-H., Liang, J.-H., Chou, H.-P., Liu, C.-H., Chang, S.-C., et al. (2018). MONAS: multi-objective neural architecture search using reinforcement learning. *arXiv [Preprint]. arXiv:1806.10332.* doi: 10.48550/arXiv.1806.10332

Jacob, G., Pramod, R. T., Katti, H., and Arun, S. P. (2021). Qualitative similarities and differences in visual object representations between brains and deep networks. *Nat. Commun.* 12, 1872. doi: 10.1038/s41467-021-22078-3

Jamil, H., Liu, Y., Caglar, T., Cole, C., Blanchard, N., Peterson, C., et al. (2023). "Hamming similarity and graph Laplacians for class partitioning and adversarial image detection," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (Vancouver, BC), 590–599. doi: 10.1109/CVPRW59228.2023.00066

Kanwisher, N., McDermott, J., and Chun, M. M. (1997). The fusiform face area: a module in human extrastriate cortex specialized for face perception. *J. Neurosci.* 17, 4302–4311.

Kheradpisheh, S. R., Ghodrati, M., Ganjtabesh, M., and Masquelier, T. (2016). Deep networks can resemble human feed-forward vision

in invariant object recognition. *Sci. Rep.* 6, 32672. doi: 10.1038/srep32672

Kingma, D. P., and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv [Preprint]. arXiv:1312.6114*. doi: 10.48550/arXiv.1312.6114

Kriegeskorte, N., Mur, M., and Bandettini, P. (2008). Representational similarity analysis - connecting the branches of systems neuroscience. *Front. Syst. Neurosci.* 2, 4. doi: 10.3389/neuro.06.004.2008

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc.),

Kumar, M., Houlsby, N., Kalchbrenner, N., and Cubuk, E. D. (2022). Do better ImageNet classifiers assess perceptual similarity better? *arXiv [Preprint]. arXiv:2203.04946*. doi: 10.48550/arXiv.2203.04946

Li, Q., Gomez-Villa, A., Bertalmío, M., and Malo, J. (2022). Contrast sensitivity functions in autoencoders. *J. Vision* 22, 8. doi: 10.1167/jov.22.6.8

Li, Z., Brendel, W., Walker, E., Cobos, E., Muhammad, T., Reimer, J., et al. (2019). "Learning from brains how to regularize machines," in *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, Inc.).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). "Microsoft COCO: common objects in context," in *Computer Vision – ECCV 2014*, eds D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Cham: Springer International Publishing), 740–755.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., et al. (2018). "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34.

Mehrer, J., Spoerer, C. J., Jones, E. C., Kriegeskorte, N., and Kietzmann, T. C. (2021). An ecologically motivated image dataset for deep learning yields better models of human vision. *Proc. Natl. Acad. Sci. U.S.A.* 118, e2011417118. doi: 10.1073/pnas.2011417118

Miller, M., Chung, S., and Miller, K. D. (2021). "Divisive feature normalization improves image recognition performance in AlexNet," in *International Conference on Learning Representations*.

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., and Kriegeskorte, N. (2014). A toolbox for representational similarity analysis. *PLoS Comput. Biol.* 10, e1003553. doi: 10.1371/journal.pcbi.1003553

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). "Efficient neural architecture search via parameters sharing," in *Proceedings of the 35th International Conference on Machine Learning* (PMLR), 4095–4104.

Prince, J. S., Charest, I., Kurzawski, J. W., Pyles, J. A., Tarr, M. J. and Kay, K. N. (2022). GLMsingle: a toolbox for improving single-trial fMRI response estimates. *bioRxiv [Preprint]*. doi: 10.1101/2022.01.31.478431

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., et al. (2021). "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning* (PMLR), 8748–8763.

RichardWebster, B., Anthony, S. E., and Scheirer, W. J. (2019). PsyPhy: a psychophysics driven evaluation framework for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 2280–2286. doi: 10.1109/TPAMI.2018.2849989

RichardWebster, B., Kwon, S. Y., Clarizio, C., Anthony, S. E., and Scheirer, W. J. (2018). "Visual psychophysics for making face recognition algorithms more explainable," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 252–270.

Roads, B. D., and Love, B. C. (2021). "Enriching ImageNet with human similarity judgments and psychological embeddings," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Nashville, TN: IEEE), 3546–3556.

Rosenke, M., van Hoof, R., van den Hurk, J., Grill-Spector, K., and Goebel, R. (2021). A probabilistic functional atlas of human occipito-temporal visual cortex. *Cereb. Cortex* 31, 603–619. doi: 10.1093/cercor/bhaa246

Rosenke, M., Weiner, K. S., Barnett, M. A., Zilles, K., Amunts, K., Goebel, R., et al. (2018). A cross-validated cytoarchitectonic atlas of the human ventral visual stream. *Neuroimage* 170, 257–270. doi: 10.1016/j.neuroimage.2017.02.040

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). "MobileNetV2: inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 4510–4520.

Schrimpf, M., Kubilius, J., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., et al. (2018). Brain-score: which artificial neural network for object recognition is most brain-like? *BioRxiv [Preprint]*. 407007. doi: 10.1101/407007

Sexton, N. J., and Love, B. C. (2022). Reassessing hierarchical correspondences between brain and deep networks through direct interface. *Sci. Adv.* 8, eabm2219. doi: 10.1126/sciadv.abm2219

Smith, L. B., and Slone, L. K. (2017). A developmental approach to machine learning? *Front. Psychol.* 8, 2124. doi: 10.3389/fpsyg.2017.02124

Storrs, K. R., Kietzmann, T. C., Walther, A., Mehrer, J., and Kriegeskorte, N. (2021). Diverse deep neural networks all predict human inferior temporal cortex well, after training and fitting. *J. Cogn. Neurosci.* 33, 2044–2064. doi: 10.1162/jocn_a_01755

Tan, M., and Le, Q. (2019). "EfficientNet: rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research*, eds K. Chaudhuri and R. Salakhutdinov (PMLR), 6105–6114.

Veerabadran, V., Raina, R., and de Sa, V. R. (2021). "Bio-inspired learnable divisive normalization for ANNs," in *SVRHM 2021 Workshop@ NeurIPS*.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). "SUN database: large-scale scene recognition from abbey to zoo," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (San Francisco, CA), 3485–3492. doi: 10.1109/CVPR.2010.5539970

Yamins, D. L., Hong, H., Cadieu, C., and DiCarlo, J. J. (2013). "Hierarchical modular optimization of convolutional networks achieves representations similar to macaque IT and human ventral stream," in *Advances in Neural Information Processing Systems*, eds C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Curran Associates, Inc.).

Yamins, D. L. K., and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nat. Neurosci.* 19, 356–365. doi: 10.1038/nn.4244

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc. Natl. Acad. Sci. U.S.A.* 111, 8619–8624. doi: 10.1073/pnas.1403112111

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). "The unreasonable effectiveness of deep features as a perceptual metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 586–595.

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 11 January 2024
DOI 10.3389/fcomp.2023.1274695

OPEN ACCESS

EDITED BY
Pavan Turaga,
Arizona State University, United States

REVIEWED BY
Jiang Liu,
Johns Hopkins University, United States
Chun Pong Lau,
City University of Hong Kong, Hong Kong SAR, China

*CORRESPONDENCE
Chao Chen
✉ chao.chen.1@stonybrook.edu
Dimitris Metaxas
✉ dnm@cs.rutgers.edu

RECEIVED 08 August 2023
ACCEPTED 19 December 2023
PUBLISHED 11 January 2024

CITATION
Zhang W, Zhang Y, Hu X, Yao Y, Goswami M, Chen C and Metaxas D (2024) Manifold-driven decomposition for adversarial robustness. *Front. Comput. Sci.* 5:1274695. doi: 10.3389/fcomp.2023.1274695

Manifold-driven decomposition for adversarial robustness

Wenjia Zhang[1], Yikai Zhang[2], Xiaoling Hu[3], Yi Yao[4], Mayank Goswami[5], Chao Chen[6]* and Dimitris Metaxas[1]*

[1]Department of Computer Science, Rutgers University, Piscataway, NJ, United States, [2]Morgan Stanley, New York, NY, United States, [3]Department of Computer Science, Stony Brook University, Stony Brook, NY, United States, [4]SRI International, Computer Vision Lab, Princeton, NJ, United States, [5]Department of Computer Science, Queens College of CUNY, New York, NY, United States, [6]Department of Biomedical Informatics, Stony Brook University, Stony Brook, NY, United States

The adversarial risk of a machine learning model has been widely studied. Most previous studies assume that the data lie in the whole ambient space. We propose to take a new angle and take the manifold assumption into consideration. Assuming data lie in a manifold, we investigate two new types of adversarial risk, the normal adversarial risk due to perturbation along normal direction and the in-manifold adversarial risk due to perturbation within the manifold. We prove that the classic adversarial risk can be bounded from both sides using the normal and in-manifold adversarial risks. We also show a surprisingly pessimistic case that the standard adversarial risk can be non-zero even when both normal and in-manifold adversarial risks are zero. We finalize the study with empirical studies supporting our theoretical results. Our results suggest the possibility of improving the robustness of a classifier without sacrificing model accuracy, by only focusing on the normal adversarial risk.

KEYWORDS

robustness, adversarial attack, manifold, topological analysis of network, generalization

1 Introduction

Machine learning (ML) algorithms have achieved astounding success in multiple domains such as computer vision (Krizhevsky et al., 2012; He et al., 2016), natural language processing (Wu et al., 2016; Vaswani et al., 2017), and robotics (Levine and Abbeel, 2014; Nagabandi et al., 2018). These models perform well on massive datasets but are also vulnerable to small perturbations on the input examples. Adding a slight and visually unrecognizable perturbation to an input image can completely change the prediction of the model. Many studies have been published, focusing on such adversarial attacks (Szegedy et al., 2013; Carlini and Wagner, 2017; Madry et al., 2017). To improve the robustness of these models, various defense methods have been proposed (Madry et al., 2017; Shafahi et al., 2019; Zhang et al., 2019). These methods mostly focus on minimizing the *adversarial risk*, i.e., the risk of a classifier when an adversary is allowed to perturb any data with an oracle.

Despite the progress in improving the robustness of models, it has been observed that compared with a standard classifier, a robust classifier often has a lower accuracy on the original data. The accuracy of a model can be compromised when one optimizes its adversarial risk. This phenomenon is called *the trade-off between robustness and accuracy*. Su et al. (2018) observed this trade-off effect on a large number of commonly used model architectures. They concluded that there is a linear negative correlation between the logarithm of accuracy and adversarial risk. Tsipras et al. (2018) proved that adversarial risk is inevitable for any classifier with a non-zero error rate. Zhang et al. (2019) decomposed the adversarial risk into the summation of standard error and boundary error. The

decomposition provides the opportunity to explicitly control the trade-off. They also proposed a regularizer to balance the trade-off by maximizing the boundary margin.

In this study, we investigate the adversarial risk and the robustness-accuracy trade-off through a new angle. We follow the classic manifold assumption, i.e., data are living in a low dimensional manifold embedded in the input space (Cayton, 2005; Niyogi et al., 2008; Narayanan and Mitter, 2010; Rifai et al., 2011). Based on this assumption, we analyze the adversarial risk with regard to adversarial perturbations within the manifold and normal to the manifold. By restricting to in-manifold and normal perturbations, we define the *in-manifold adversarial risk* and *normal adversarial risk*. Using these new risks, together with the standard risk, we prove an upper bound and a lower bound for the adversarial risk. We also show that the bound is tight by constructing a pessimistic case. We validate our theoretical results using synthetic and real-world datasets.

Our study sheds light on a new aspect of the robustness-accuracy trade-off. Through the decomposition into in-manifold and normal adversarial risks, we might find an extra margin to exploit without confronting the trade-off.

A preliminary version of this study, which mainly focuses on the theoretical results, was published in the study mentioned in the reference (Zhang et al., 2022). The major differences between this article and Zhang et al. (2022) include the adding of experimental validation on real-world datasets to verify our theoretical discoveries. To realize this validation process, we employ the Tangent-Normal Adversarial Regularization algorithm (TNAR) by Yu et al. (2019), which obtain the normal and in-manifold directions within real data. This strategic utilization of Tangent-Normal Adversarial Regularization algorithm not only strengthens the empirical foundation of our research but also indicates our commitment to bridging the gap between theoretical insights and practical applicability. By integrating this experimental result, we not only refines the theoretical framework but also provides an empirical verification, enhancing the overall credibility and relevance of our research findings.

1.1 Related works

Robustness-accuracy trade-off: It was believed that a classifier cannot be optimally accurate and robust at the same time. Different articles study the trade-off between robustness and accuracy (Su et al., 2018; Tsipras et al., 2018; Dohmatob, 2019; Zhang et al., 2019). One main question is whether the best trade-off actually exists. Tsipras et al. (2018) first recognized this trade-off phenomenon by empirical results and further proved that the trade-off exists under the infinite data limit. Dohmatob (2019) showed that a high accuracy model can inevitably be fooled by the adversarial attack. Zhang et al. (2019) gave examples showing that the Bayes optimal classifier may not be robust.

However, others have different views on this trade-off or even its existence. In contrast to the idea that the trade-off is unavoidable, according to these studies, the drop of accuracy is not due to the increase in robustness. Instead, it is due to a lack of effective optimization methods (Shaham et al., 2018; Awasthi et al., 2019; Rice et al., 2020) or better network architecture (Fawzi et al., 2018; Guo et al., 2020). Yang et al. (2020) showed the existence of both robust and accurate classifiers and argued that the trade-off is influenced by the training algorithm to optimize the model. They investigated distributionally separated dataset and claimed that the gap between robustness and accuracy arises from the lack of a training method that imposes local Lipschitzness on the classifier. Remarkably, in the study mentioned in the reference (Carmon et al., 2019; Gowal et al., 2020; Raghunathan et al., 2020), it was shown that with certain augmentation of the dataset, one may be able to obtain a model that is both accurate and robust.

Our theoretical results upperbound the adversarial risk using different manifold-derived risks plus the standard Bayes risk (which is essentially the accuracy). This quantitative relationship provides a pathway toward an optimal robustness-accuracy trade-off. In particular, our results suggest that, by adversarial training, the model against perturbations in the normal direction can improve robustness without sacrificing accuracy.

Manifold assumption: One important line of research focuses on the manifold assumption on the data distribution. This assumption suggests that observed data are distributed on a low dimensional manifold (Cayton, 2005; Narayanan and Mitter, 2010; Rifai et al., 2011), and there exists a mapping that embeds the low dimension manifold in some higher dimension space. Traditional manifold learning methods (Tenenbaum et al., 2000; Saul and Roweis, 2003) that try to recover the embedding by assuming the mapping preserves certain properties such as distances or local angles. Following this assumption, on the topic of robustness, Tanay and Griffin (2016) showed the existence of adversarial attack on the flat manifold with linear classification boundary. It was proved later (Gilmer et al., 2018) that in-manifold adversarial examples exist. They stated that high-dimension data are highly sensitive to l_2 perturbations and pointed out that the nature of adversarial is the issue with potential decision boundary. Later, Stutz et al. (2019) showed that with the manifold assumption, regular robustness is correlated with in-manifold adversarial examples, and therefore, accuracy and robustness may not be contradictory goals. Further discussion (Xie et al., 2020) even suggested that adding adversarial examples to the training process can improve the accuracy of the model. Lin et al. (2020) used perturbation within a latent space to approximate in-manifold perturbation. Most existing studies only focused on in-manifold perturbations. To the best of our knowledge, we are the first to discuss normal perturbation and normal adversarial risk. We are also unaware of any theoretical results proving upper/lower bounds for adversarial risk in the manifold setting.

We also note a classic manifold reconstruction problem, i.e., reconstructing a d-dimensional manifold given a set of points sampled from the manifold. A large group of classical algorithms (Edelsbrunner and Shah, 1994; Dey and Goswami, 2006; Niyogi et al., 2008) are probably good, i.e., they give a guarantee of reproducing the manifold topology with a sufficiently large number of sample points.

Under data manifold assumption, Stutz et al. (2019) and Shamir et al. (2021) first reconstruct the data manifold using Generative Networks. Then, with the approximation of manifold,

the authors explored different approaches for computing in-manifold attack examples under manifold assumption. Stutz et al. (2019) approximate the data manifold using VAE models and then directly perturbed the latent space without considering the perturbed distance in the original space, making it difficult to bound their on-manifold examples. On the other hand, Shamir et al. (2021) first perturbed the latent code to generate a set of basis in the tangential space, using these basis vectors to generate on-manifold directions and search for in-manifold attack examples. In the study by Lau et al. (2023), the author employs generative model-based methods to simultaneously perturb the input data in both the original space and the latent space. This dual perturbation process results in in-manifold perturbed data even on high-resolution datasets.

The Tangent-Normal Adversarial Regularization (TNAR) algorithm (Yu et al., 2019) distinguishes itself by finding tangential directions along the data manifold through power iteration and conjugate gradient algorithms. Subsequently, we perform a targeted search along these tangential directions to find valid L_p norm-based adversarial examples while ensuring effective perturbation bounds on the in-manifold examples.

2 Manifold-based risk decomposition

In this section, we state our main theoretical result (Theorem 1), which decomposes the adversarial risk into normal adversarial risk and in-manifold (or tangential) adversarial risk. We first define these quantities and set up basic notations. Next, we state the main theorem in Section 2.3. For the sake of simplicity, we describe our main theorem in the setting of binary labels, $\{-1, 1\}$. Informally, the main theorem states that under mild assumptions, (1) the adversarial risk can be upper-bounded by the sum of the standard risk, normal adversarial risk, in-manifold adversarial risk, and another small risk called nearby-normal-risk; (2) when the normal adversarial risk is zero, the adversarial risk can be upper-bounded by the standard risk and the in-manifold adversarial risk. Finally, we show in Theorem 2 that the bounds are tight by constructing pessimistic cases.

2.1 Data manifold

Let $(\mathbb{R}^D, ||.||)$ denote the D dimensional Euclidean space with ℓ_2-norm, and let p be the data distribution. For $x \in \mathbb{R}^D$, let $B_\epsilon(x)$ be the open ball of radius ϵ in $\mathbb{R}^D$ with center at x. For a set $A \subset \mathbb{R}^D$, define $B_\epsilon(A) = \{y : \exists x \in A, d(x, y) < \epsilon\}$.

Let $\mathcal{M} \subset \mathbb{R}^D$ be a d-dimensional compact smooth manifold embedded in $\mathbb{R}^D$. Thus, for any $x \in M$, there is a corresponding coordinate chart (U, g), where $U \ni x$ is an open set of $\mathcal{M}$ and g is a homeomorphism from U to a subset of $\mathbb{R}^d$. Let $T_x\mathcal{M}$ and $N_x\mathcal{M}$ denote the tangent and normal spaces at x. Intuitively, the tangent space $T_x\mathcal{M}$ is the space of tangent directions or equivalence classes of curves in $\mathcal{M}$ passing through x, with two curves considered equivalent if they are tangent at x. The normal space $N_x\mathcal{M}$ is the set of vectors in $\mathbb{R}^D$ that are orthogonal to any vector in $T_x\mathcal{M}$. Since $\mathcal{M}$ is a smooth d-manifold, $T_x\mathcal{M}$ and $N_x\mathcal{M}$ are d and $(D - d)$ dimensional vector spaces, respectively (see Figure 1 for an illustration). For detailed definitions, we refer the reader to the study mentioned in the reference (Bredon, 2013).

We assume that the data and (binary) label pairs are drawn from $\mathcal{M} \times \{-1, 1\}$, according to some unknown distribution $p(x, y)$. Note that $\mathcal{M}$ is unknown. A score function $f(x)$ is a continuous function from $\mathbb{R}^D$ to $[0, 1]$. We denote by 1_A the indicator function of the event A that is 1 if A occurs and 0 if A does not occur and will use it to represent the 0-1 loss.

2.2 Robustness and risk

Given data from $\mathcal{M} \times \{-1, 1\}$ drawn according to data distribution p and a classifier f on $\mathbb{R}^D$, we define three types of risks. The first, adversarial risk, has been extensively studied in machine learning literature:

Definition 1 (Adversarial risk). Given $\epsilon > 0$, define the adversarial risk of classifier f with budget ϵ to be

$$R_{adv}(f, \epsilon) := \mathbb{E}_{(x,y)\sim p} \mathbb{1}(\exists x' \in B_\epsilon(x) : f(x')y \leq 0)$$

Notice that $B_\epsilon(x)$ is the open ball around x in $\mathbb{R}^D$ (the ambient space).

Next, we define risk that is concerned only with in-manifold perturbations. Previously, Gilmer et al. (2018) and Stutz et al. (2019) showed that there exist in-manifold adversarial examples and empirically demonstrated that in-manifold perturbations are a cause of the standard classification error. Therefore, in the following, we define the in-manifold perturbations and in-manifold adversarial risk.

Definition 2 (In-manifold Adversarial Risk). Given $\epsilon > 0$, the in-manifold adversarial perturbation for classifier f with budget ϵ is the set

$$B^{in}_\epsilon(x) := \{x' \in \mathcal{M} : \|x - x'\| \leq \epsilon\}$$

The in-manifold adversarial risk is

$$R^{in}_{adv}(f, \epsilon) := \mathop{\mathbb{E}}_{(x,y)\sim p} \mathbb{1}(\exists x' \in B^{in}_\epsilon(x) : f(x')y \leq 0)$$

We remark that while the above perturbation is on the manifold, many manifold-based defense algorithms use generative models to estimate the homeomorphism (the manifold chart) $z = g(x)$ for real-world data. Therefore, instead of in-manifold perturbation, one can also use an equivalent η-budget perturbation in the *latent* space. However, for our purposes, the in-manifold definition will be more convenient to use. Finally, we define the *normal* risk:

Definition 3 (Normal adversarial risk). Given $\epsilon > 0$, the normal adversarial perturbation for classifier f with budget ϵ is be the set

$$B^{nor}_\epsilon(x) := \{x' : x' - x \in N_x\mathcal{M}, \|x - x'\| \leq \epsilon\}$$

Define the normal adversarial risk as

$$R^{nor}_{adv}(f, \epsilon) := \mathop{\mathbb{E}}_{(x,y)\sim p} \mathbb{1}(\exists x' \neq x \in B^{nor}_\epsilon(x) : f(x')y \leq 0)$$

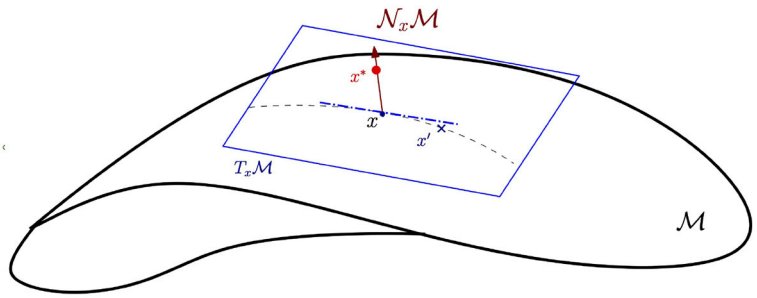


FIGURE 1
Tangent and normal spaces of a manifold. Here, x is the original data point on the data manifold $\mathcal{M}$. $T_x\mathcal{M}$ is the tangent space along the data manifold $\mathcal{M}$ at point x. x' is the in-manifold adversarial example on the data manifold $\mathcal{M}$. $N_x\mathcal{M}$ denotes the normal space perpendicular to $T_x\mathcal{M}$. x^* is an adversarial perturbation along $N_x\mathcal{M}$.

Notice that the normal adversarial risk is non-zero if there is an adversarial perturbation $x' \neq x$ in the normal direction at x. Finally, we have the usual *standard risk*: $R_{std}(f) := \mathbb{E}_{(x,y)\sim p}\, \mathbb{1}(f(x)y \leq 0)$.

2.3 Main result: decomposition of risk

In this section, we state our main result that decomposes the adversarial risk into its tangential and normal components. Our theorem will require a mild assumption on the decision boundary $DB(f)$ of the classifier f, i.e., the set of points x where $f(x) = 0$.

Assumption [A]: For all $x \in DB(f)$ and all neighborhoods $U \ni x$ containing x, there exist points x_0 and x_1 in U such that $f(x_0) < 0$ and $f(x_1) > 0$.

This assumption states that a point that is difficult to classify by f has points of both labels in any given neighborhood around it. In particular, this means that the decision boundary does not contain an open set. We remark that both Assumption A and the continuity requirement for the score function f are implicit in previous decomposition results such as Equation 1 in the study by Zhang et al. (2019). Without Assumption A, the "neighborhood" of the decision boundary in the study by Zhang et al. (2019) will not contain the decision boundary, and it is easy to give a counterexample to Equation 1 in the study by Zhang et al. (2019) if f is not continuous.

Our decomposition result will decompose the adversarial risk into the normal and tangential directions: however, as we will show, an "extra term" appears, which we define next:

Definition 4 (NNR Nearby-Normal-Risk). Fix $\epsilon > 0$. Denote by $A(x, y)$ the event that $\forall x' \in B^{nor}_{\epsilon}(x), f(x')y > 0$, i.e., the normal adversarial risk of x is zero.

Denote by $B(x, y)$ the event that

$$\exists x' \in B^{in}_{2\epsilon}(x) : (\exists z \in B^{nor}_{\epsilon}(x') : f(z)f(x') \leq 0),$$

i.e., x has a point x' near it such that x' has non-zero normal adversarial risk.

Denote by $C(x, y)$ the event $\forall x' \in B^{in}_{2\epsilon}(x), f(x')y > 0$, i.e., x has no adversarial perturbation in the manifold within distance 2ϵ.

The Nearby-Normal-Risk (denoted as NNR) of f with budget ϵ is defined to be

$$\mathbb{E}_{(x,y)\sim p}\, \mathbb{1}(A(x, y) \wedge B(x, y) \wedge C(x, y)),$$

where $\wedge$ denotes "and".

We are now in a position to state our main result.

Theorem 1. [Risk Decomposition] Let $\mathcal{M}$ be a smooth compact manifold in $\mathbb{R}^D$ and let data be drawn from $\mathcal{M} \times \{-1, 1\}$, according to some distribution p. There exists a $\Delta > 0$ depending only on $\mathcal{M}$ such that the following statements hold for any $\epsilon < \Delta$. For any score function f satisfying assumption A,

(I)

$$R_{adv}(f, \epsilon) \leq R_{std}(f) + R^{nor}_{adv}(f, \epsilon) + R^{in}_{adv}(f, 2\epsilon) + \mathrm{NNR}(f, \epsilon). \quad (1)$$

(II) If $R^{nor}_{adv}(f, \epsilon) = 0$, then

$$R_{adv}(f, \epsilon) \leq R_{std}(f) + R^{in}_{adv}(f, 2\epsilon)$$

Remark:

1. The first result decomposes the adversarial risk into the standard risk, the normal adversarial risk, the in-manifold adversarial risk, and an "extra term"—the Nearby-Normal-Risk. The NNR comes into play when a point x does not have normal adversarial risk, and the score function on all points nearby agrees with $y(x)$, yet there is a point near x that has non-zero normal adversarial risk.
2. The second result states that if the normal adversarial risk is zero, the ϵ-adversarial risk is bounded by the sum of the standard risk and the 2ϵ in-manifold adversarial risk.
3. Our bound suggests that there may be "free lunch" in robustness-accuracy trade-off. There is an extra margin one can exploit without confronting the trade-off. Specifically, this corollary suggests that by solely minimizing the normal adversarial risk, we can govern the difference between adversarial risk and standard accuracy by focusing exclusively on in-manifold adversarial risk. This insight provides a pathway

to navigating the trade-off under the condition of zero normal adversarial risk, wherein the key lies in minimizing the in-manifold risk. This strategic approach opens up ways for fine-tuning and optimizing the robustness-accuracy trade-off, shedding light on potential methods for achieving better performance on robust models.

One may wonder if a decomposition of the form $R_{adv}(f,\epsilon) \leq R_{std}(f) + R^{nor}_{adv}(f,\epsilon) + R^{in}_{adv}(f,2\epsilon)$ is possible. We prove that this is not possible. The complete proof of Theorem 1 is technical and is provided in the Supplementary material. Here, we provide a sketch of the proof first.

2.3.1 Proof sketch of theorem 1

We first address the existence of the constant Δ that only depends on $\mathcal{M}$ in the theorem statement. Define a *tubular neighborhood* of $\mathcal{M}$ as a set $\mathcal{N} \subset \mathbb{R}^D$ containing $\mathcal{M}$ such that any point $z \in \mathcal{N}$ has a unique projection $\pi(z)$ onto $\mathcal{M}$ such that $z - \pi(z) \in N_{\pi(z)}\mathcal{M}$. Thus, the normal line segments of length ϵ at any two points $x, x' \in \mathcal{M}$ are disjoint.

By Theorem 11.4 in the study by Bredon (2013), we know that there exists Δ such that $N := \{y \in \mathbb{R}^D : dist(y,\mathcal{M}) < \Delta\}$ is a tubular neighborhood of $\mathcal{M}$. The Δ guaranteed by Theorem 11.4 is the Δ referred to our theorem, and the budget ϵ is constrained to be at most Δ.

For simplicity, we first sketch the proof of the case when y is deterministic (the setting of Corollary 1). Considering a pair $(x,y) \sim p$, x has an adversarial perturbation x' within distance ϵ. We show that one of the four cases must occur:

- $x' = x$ (standard risk).
- $x' \neq x, x' \in N_x\mathcal{M}$, and $f(x)y > 0$ (normal adversarial risk).
- Let $x'' = \pi(x')$ (the unique projection of x' onto $\mathcal{M}$), then $d(x'',x) \leq 2\epsilon$ and either
  - $f(x'')y \leq 0$ and x have an 2ϵ in-manifold adversarial perturbation (in-manifold adversarial risk) or
  - $f(x'')f(x') \leq 0$, which implies that x is within 2ϵ of a point $x'' \in \mathcal{M}$ that has non-zero normal adversarial risk (NNR: nearby-normal-risk).

The second of these sets is $Z^{nor}(f,\epsilon)$ in the setting of Corollary 1. One can observe that the four cases correspond to the four terms in Equation 2.

For the proof of Theorem 1, one has to observe that since y is not deterministic, the set $Z^{nor}(f,\epsilon)$ is random. One then has to average over all possible $Z^{nor}(f,\epsilon)$ and show that the average equals NNR.

For the second part of Theorem 1 and Corollary 1, we observed that if the normal adversarial risk is zero, in the last case, x'' has non-zero normal adversarial risk, with normal adversarial perturbation x'. Unless x'' is on the decision boundary, by continuity of f one can show that there exists an open set around x'' such that all points have non-zero normal adversarial risk. This contradicts the fact that the normal adversarial risk is zero, implying that case 4 happens only on a set of measure zero (recalling that by assumption A, the decision boundary does not contain any open set). This completes the proof sketch.

Theorem 2. [Tightness of decomposition result]

For any $\epsilon < 1/2$, there exists a sequence $\{f_n\}_{n=1}^{\infty}$ of continuous score functions such that

(I) $R_{std}(f) = 0$ for all $n \geq 1$,

(II) $R^{in}_{adv}(f_n, 2\epsilon) = 0$ for all $n \geq 1$, and

(III) $R^{nor}_{adv}(f_n,\epsilon) \to 0$ as n goes to infinity,

but $R_{adv}(f,\epsilon) = 1$ for all $n > \frac{1}{\sqrt{3\epsilon}}$.

Thus, all three terms, except the NNR term, indicate zero, but the adversarial risk (the left side of Equation 2) indicates one.

Here, we provide a sketch of the proof of Theorem 2. Then, we give the complete proof in the Supplementary material.

2.3.2 Proof of theorem 2

Let $\mathcal{M} = [0,1]$ and fix $\epsilon < 1/2$ and $n \geq 1$. We will think of data as lying in the manifold $\mathcal{M}$ and $\mathbb{R}^2$ as the ambient space. The true distribution is simply $\eta(x) = 1$ for all $x \in \mathcal{M}$, hence $y \equiv 1$ (all labels on $\mathcal{M}$ are 1).

Let $\ell_1 = \frac{n-1}{n(n+1)}$ and $\ell_2 = \frac{1}{n^2}$. Note that $(n+1)\ell_1 + n\ell_2 = 1$. Consider the following partition of $\mathcal{M} = A_0 \cup B_1 \cup A_1 \cup B_2 \cup \cdots \cup B_n \cup A_n$, where A_i $(0 \leq i \leq n)$ is of length ℓ_1 and B_i $(1 \leq i \leq n)$ is an interval of length ℓ_2. The interval $A_0, B_1, A_1, \cdots, B_n, A_n$ appears in this order from left to right.

For ease of presentation, we will consider $\{0,1\}$ binary labels and build score functions f_n, taking values in $[0,1]$ that satisfy the conditions of the Theorem.

For an $x \in A_i$ for some $0 \leq i \leq n$, define $g_n(x) = 1$. For $x \in B_i$ for some $1 \leq i \leq n$, define $g_n(x) = \epsilon/2$. Observe that $\epsilon/2 < 1/4$.

We now define the decision boundary of f_n as the set of points in $\mathbb{R}^2$ on the "graph" of g_n and $-g_n$. That is,

$$DB(f_n) = \left\{(x, cg_n(x)) : x \in [0,1], c \in \{-1,1\}\right\}.$$

(see Figure 2 for a picture of the upper decision boundary).

Now, let f_n be any continuous function with decision boundary $DB(f_n)$ as above. That is, $f_n : \mathbb{R}^2 \to [0,1]$ is such that $f_n(x,t) > 1/2$ if $|t| < g_n(x)$, $f_n(x,t) < 1/2$ if $|t| > g_n(x)$ and $f_n(x,y) = 1/2$ if $|t| = g_n(x)$.

In-manifold adversarial risk is zero: Observe that since $\eta(x) = 1$ on $[0,1]$, the in-manifold adversarial risk of f_n is zero, since $f_n(x,0) > 1/2$, and so $\text{sign}(2f_n - 1)$ equals 1, which is the same as the label y at x. This means that there are no in-manifold adversarial perturbations, no matter the budget. Thus, $R^{in}_{adv}(f_n,\epsilon) = 0$ for all $n \geq 1$.

Normal adversarial risk goes to zero: Next, we consider the normal adversarial risk. If $x \in A_i$ for some i, a point in the normal ball with budget ϵ is of the form (x,t) with $|t| < \epsilon < 1/2$ but $f_n(x,t) > 1/2$ for such points and thus sign $(2f_n - 1) = y(x)$. Thus, $x \in A_i$ does not contribute to the normal adversarial risk. If $x \in B_i$ for some i then $f_n(x,\epsilon) < 1/2$ while $f_n(x,0) > 1/2$, and hence such x contributes to the normal adversarial risk. Thus,

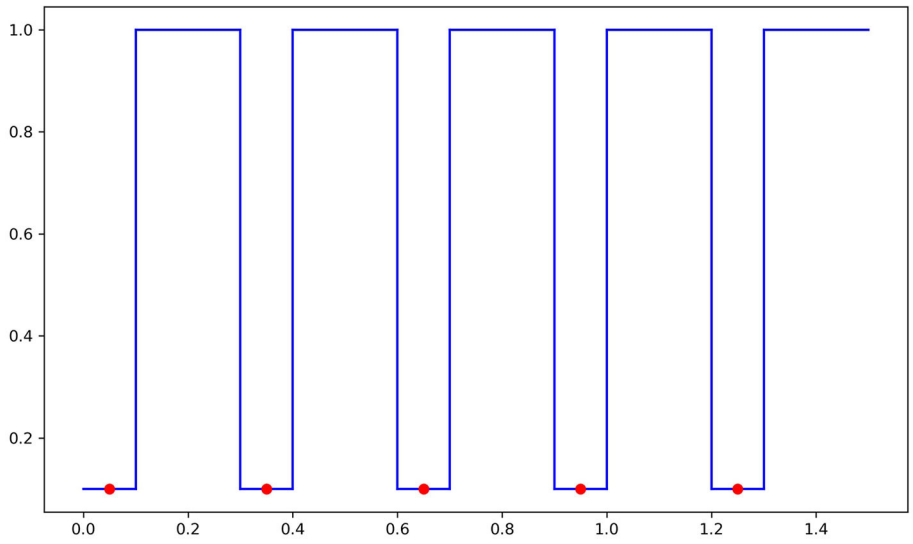


FIGURE 2
Lower bound illustration.

$R^{nor}_{adv}(f_n, \epsilon) = \sum_{i=1}^{n} \mu(B_i) = \sum_{i=1}^{n} \ell_2 = 1/n$, which goes to zero as n goes to infinity.

Adversarial risk goes to one: Now, we show that $R_{adv}(f_n, \epsilon)$ goes to one. In fact, we will show that as long as n is sufficiently large, the adversarial risk is 1. Consider n such that $\ell_1 := \frac{n-1}{n(n+1)} < \sqrt{3}\epsilon$. Note that such an n exists simply because ℓ_1 goes to zero as n goes to infinity and $n > \frac{1}{\sqrt{3}\epsilon}$ works.

Clearly, points in B_i contribute to adversarial risk as they have adversarial perturbations in the normal direction. However, if we consider $x \in A_i$ (which does not have adversarial perturbations in the normal direction or in-manifold), we show that there still exists an adversarial perturbation in the ambient space: that is, there exists a point x' such that a), the distance between $(x', \epsilon/2)$ and $(x, 0)$ is at most ϵ and b) $sign(2f_n(x, \epsilon/2)) \neq sign(2f_n(x, 0))$. Let x' be the closest point in $B := \cup B_i$ to x. Then, $|x' - x| \leq \ell_1/2 < \sqrt{3}\epsilon/2$. Thus, the distance between $(x', \epsilon/2)$ and $(x, 0)$ is at most $\sqrt{(\sqrt{3}\epsilon/2)^2 + (\epsilon/2)^2} = \epsilon$. Since $x' \in B, f_n(x', \epsilon/2) < 1/2$, whereas $f_n(x, 0) < 1/2$, $(x', \epsilon/2)$ is a valid adversarial perturbation around x.

Thus, for all $x \in [0, 1]$, there exists an adversarial perturbation within budget ϵ and therefore $R_{adv}(f_n, \epsilon) = 1$ as long as $n > \frac{1}{\sqrt{3}\epsilon}$. This completes the proof.

2.4 Decomposition when y is deterministic

Let $\eta(x) = Pr(y = 1|x)$. We consider here the simplistic setting when $\eta(x)$ is either 0 or 1, i.e., y is a deterministic function of x. In this case, we can explain our decomposition result in a simpler way.

Let $Z^{nor}(f, \epsilon) := \{x \in \mathcal{M} : f(x)y > 0 \text{ and } \exists x' \neq x \in B^{nor}_{\epsilon}(x), f(x')y(x) \leq 0\}$. That is, $Z^{nor}(f, \epsilon)$ is the set of points with no standard risk but with a non-zero normal adversarial risk under a positive but less than ϵ normal perturbation. Let $\overline{Z^{nor}(f, \epsilon)} = \mathcal{M} \setminus Z^{nor}(f, \epsilon)$ be the complement of $Z^{nor}(f, \epsilon)$. For a set $A \subset \mathcal{M}$, let $\mu(A)$ denote the measure of A.

Corollary 1. Let $\mathcal{M}$ be a smooth compact manifold in $\mathbb{R}^D$, and let $\eta(x) \in \{0, 1\}$ for all $x \in M$. There exists a $\Delta > 0$ depending only on $\mathcal{M}$ such that the following statements hold for any $\epsilon < \Delta$. For any score function f satisfying assumption A,

(I)

$$\begin{aligned} R_{adv}(f, \epsilon) \leq\ & R_{std}(f) + R^{in}_{adv}(f, 2\epsilon) + R^{nor}_{adv}(f, \epsilon) \\ & + \mu(\overline{Z^{nor}(f, \epsilon)}) \cap B_{2\epsilon}(Z^{nor}(f, \epsilon)) \end{aligned} \tag{2}$$

(II) If $R^{nor}_{adv}(f, \epsilon) = 0$, then

$$R_{adv}(f, \epsilon) \leq R_{std}(f) + R^{in}_{adv}(f, 2\epsilon).$$

Therefore, in this setting, the adversarial risk can be decomposed into the in-manifold adversarial risk and the measure of a neighborhood of the points that have non-zero normal adversarial risk.

3 Experiment: synthetic dataset

In this section, we verify the decomposition upper bound in Theorem 1 on synthetic data sets. We train different classifiers and empirically verify the inequalities on these classifiers.

In our experiments, instead of using L_2 norm to evaluate the perturbation, we search the neighborhood under L_∞ norm, which would produce a stronger attack than L_2 norm one. The experimental results indicate that our theoretical analysis may hold for an even stronger attack.

3.1 Toy data set and perturbed data

We generate four different data sets where we study both the *single decision boundary case* and the *double decision boundary case*. The first pair of datasets are in 2D space and the second pair is in 3D. We aim to provide empirical evidence for the claim *i*) in the Theorem 1 using the single and double decision boundary data.

For the 2D case, we sample training data uniformly from a unit circle $C_1 : x_1^2 + x_2^2 = 1$. For the single decision boundary data set, we set

$$y = 2\mathbb{1}(x_1 > 0) - 1 \text{ (Single Decision Boundary)}$$
$$y = 2\mathbb{1}(x_1 x_2 > 0) - 1 \text{ (Double Decision Boundary)}$$

The visualization of the dataset is shown in Figures 3A, B). In particular, we set unit circle C_1 has $\Delta = 1$, we set the perturbation budget to be $\varepsilon \in [0.01, 0.3]$. Moreover, the normal direction is alone the radius of the circle.

In the 3D case, we set the manifold to be $\mathcal{M} : x_3 = 0$ and generate training data in region $[-\pi, \pi] \times [-\pi, \pi]$ on x_1x_2-plane. We set

$$y = 2\mathbb{1}\left[x_1 > sin(x_2)\right] - 1(\text{Single})$$
$$y = 2\mathbb{1}\left[(x_1 - sin(x_2))x_2 > 0\right] - 1(\text{Double})$$

Figures 3C, D show these two cases. For the single decision boundary example, due to the manifold being flat, we have $\Delta = \infty$, and we explore the ϵ value in range $[0.1, 0.8]$. For the double decision boundary, the distance to the decision boundary is half of the distance in the single boundary case. Therefore, we set the range of perturbation to be $[0.1, 0.4]$.

3.2 Algorithm for estimating different risks

To empirically estimate the decomposition of adversarial risk, we need to estimate the normal adversarial risk R_{adv}^{nor}, the in-manifold adversarial risk R_{adv}^{in}, the classic adversarial risk R_{adv}, and the standard risk R_{std}. The standard risk is obtained by evaluating on the standard classifier f trained by the original training data set. For the classic adversarial risk R_{adv}, we follow the classic approach and train the adversarial classifier f^{adv} following the classic adversarial training Algorithm (Madry et al., 2017). The risk is evaluated on perturbed example x^{adv} computed by the classic Projected Gradient Descent Algorithm (Madry et al., 2017). To estimate the other two risks, R_{adv}^{nor} and R_{adv}^{in}, we generate adversarial perturbations along normal and in-manifold directions and use these perturbations to train different robust classifiers.

To compute the in-manifold perturbation, we design two methods. The first one is using grid search to go through all the perturbations in the manifold within the ϵ budget and return the point with maximum loss as in-manifold perturbation x^{in}. Although this seems to be the best solution, it is quite expensive due to the grid-search procedure. Therefore, we resort to a second method in our experiments using Projected Gradient Descent (PGD) method to find a general adversarial point x^{adv} in ambient space and then project x^{adv} back to the data manifold $\mathcal{M}$. In Supplementary material, we will further compare these two methods.

Next, we explain how to obtain normal direction perturbations x^{nor}. Note that in both the 2D and 3D toy datasets, the dimension of the normal space is 1. Therefore, the normal space at point x can be represented by $N_x\mathcal{M} = \{x + t \cdot v | 0 < t < \epsilon\}$. Here, v is the unit normal vector and can be computed exactly in close-form in our toy data.

We list the R_{adv} and RHS value for 2D and 3D datasets for all classifiers in Tables 1, 2.

3.3 Empirical results and discussion

2D dataset: We generate 1,000 2D training data uniformly. The classifier is a two-layer feed-forward network. Each classifier is trained with Stochastic Gradient Descent (SGD) with a learning rate of 0.1 for 1,000 epochs. In addition, since $\Delta = 1$ for the unit circle, the upper bound of ϵ value is up to 1. Hence, we run experiments for ϵ from 0.01 to 0.3. We leave more discussion and visualization of this phenomenon in Supplementary material. The right hand side values of the inequality for all three classifiers are presented in Table 1. We could observe that the upper bounds hold for 2D data, at least for all these classifiers.

3D dataset: We generate 1,000 training data from the data set. The classifier is a four-layer feedforward network. We use SGD with a learning rate of 0.1 and weight decay of 0.001 to train the network. The total training epoch is 2,000. In Table 2, we list same classifiers trained on the 3D dataset. Similar to the 2D dataset, for all classifiers, inequality 1 holds. Due to the limit of the space, we provide additional empirical results in Supplementary material.

4 Experiment: real-world datasets

In this section, we verify our theoretical results on real-world dataset experiments. The challenge is to find a manifold representation and generate in-manifold/normal perturbations. We use an Autoencoder to represent the manifold. Next, we use the TNAR algorithm to generate in-manifold perturbations. We also extend TNAR to generate normal perturbations. These in-manifold/normal perturbations allow us to estimate different risks.

In Section 4.1, we explain how to learn the manifold representation. In Sections 4.2 and 4.3, we provide details on finding in-manifold and normal adversarial perturbation, respectively. Finally, in Section 4.4, we validate our theoretical bound.

Datasets: We utilize three commonly used datasets, two of which are grayscale: MNIST and FashionMNIST. Both of these datasets comprise 28×28 pixel images. **MNIST dataset** contains handwritten digits ranging from 0 to 9, each labeled accordingly. The dataset is divided into 60,000 training samples and 10,000 testing samples. **FashionMNIST dataset** consists of images of clothing items, with each item labeled into one of ten different categories. It includes 60,000 training samples and 10,000 testing samples. In addition to the grayscale datasets, we also incorporate

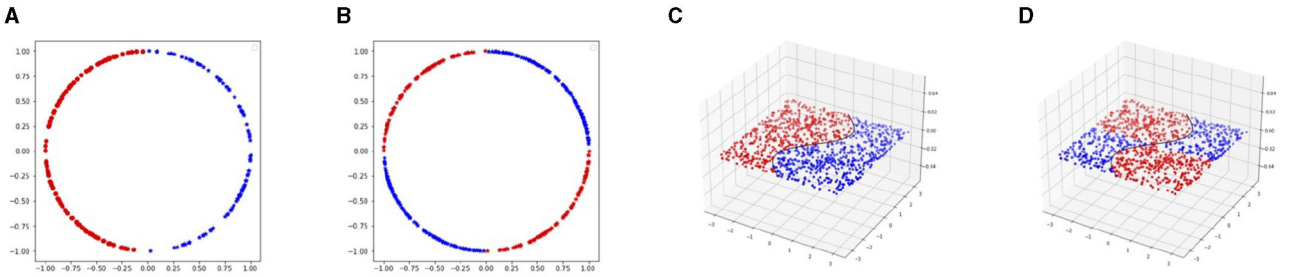


FIGURE 3
In this figure, we show our four toy data set. On the left side, 2D data is set on a unit circle. The single decision boundary data are linearly separated by the y-axis. Moreover, in the double decision boundary case, the circle is separated into four parts with x-axis and y-axis. On the right side, the 3D data is set. The data are distributed in a square area on x_1x_2-plane. In the single decision boundary example, the data are divided by the curve $x_1 = sin(x_2)$. Moreover, in the double decision boundary situation, we add the y-axis as the extra boundary. **(A)** 2D single decision boundary. **(B)** 2D double decision boundary. **(C)** 3D single decision boundary. **(D)** 3D double decision boundary.

TABLE 1 2D adversarial risk comparison.

Single boundary	f		f^{adv}		Double boundary	f		f^{adv}	
ϵ	R^{adv}	RHS	R^{adv}	RHS	ϵ	R^{adv}	RHS	R^{adv}	RHS
0.01	0.0110	0.022	0.0110	0.022	0.01	0.0080	0.0286	0.0060	0.0296
0.02	0.0130	0.0449	0.0130	0.0449	0.02	0.0240	0.0694	0.0230	0.2525
0.03	0.0230	0.063	0.0250	0.0671	0.03	0.0510	0.1333	0.0460	0.1363
0.05	0.0280	0.0794	0.0300	0.0784	0.05	0.0620	0.1810	0.0620	0.1640
0.1	0.0709	0.1652	0.0699	0.1645	0.1	0.1170	0.3398	0.1169	0.3071
0.15	0.0979	0.2831	0.1009	0.2886	0.15	0.1850	0.6059	0.1860	0.4895
0.2	0.128	0.3951	0.126	0.3971	0.2	0.242	0.8763	0.247	0.8002
0.25	0.1660	0.4966	0.1630	0.4931	0.25	0.3139	1.	0.3169	0.9971
0.3	0.1979	0.4509	0.1979	0.5613	0.3	0.386	0.9615	0.379	1

TABLE 2 3D adversarial risk comparison.

Single boundary	f		f^{adv}		Double boundary	f		f^{adv}	
ϵ	R^{adv}	RHS	R^{adv}	RHS	ϵ	R^{adv}	RHS	R^{adv}	RHS
0.1	0.0450	0.0992	0.0410	0.092	0.1	0.0649	0.1654	0.0789	0.153
0.2	0.1139	0.2297	0.0999	0.229	0.2	0.1700	0.3858	0.1370	0.3341
0.3	0.1550	0.3106	0.136	0.3216	0.3	0.2159	0.4740	0.1810	0.4208
0.4	0.2089	0.3765	0.1680	0.3889	0.4	0.3000	0.6051	0.2069	0.5325

one color dataset. **SVHN dataset** contains 10 different classes of digit images, each with 3×32×32 pixels.

Classifier: We selected ResNet18 as our classifier and employed the Adam optimizer with learning rate to be 0.001 for our experiments. To train the ResNet18 network for each dataset, we continued training until the training accuracy reached 99%. On the MNIST dataset, our trained classifier achieved an test accuracy of 99.24%. When applied to the FASHIONMNIST dataset, the classifier demonstrated a test accuracy of 94.78%. Moreover, the SVHN dataset obtain a test accuracy of 96.74%.

Classic adversarial training: To evaluate the robustness of the classifier, we generated Projected Gradient Descent (PGD) (Madry et al., 2017) attacks using L_2 norms. For creating an adversarial attack, we set the L_2 attack budget to 1.5 for the MNIST and FASHIONMNIST datasets and 0.25 for SVHN. For L_∞ attacks, the perturbation budget was set to 0.3 for grayscale datasets and 8/255 for color images.

4.1 Approximation of data manifold

We employed an autoencoder structure consisting of 7 VGG blocks to approximate the underlying data manifold. The autoencoder was trained using Mean Square Loss of 400 epochs.

The output of the trained autoencoder is presented in Figure 4. We observe that for MNIST and FASHIONMNIST datasets, the reconstruction results are very close to the input data. For the SVHN dataset, while the reconstruction images are reasonably close

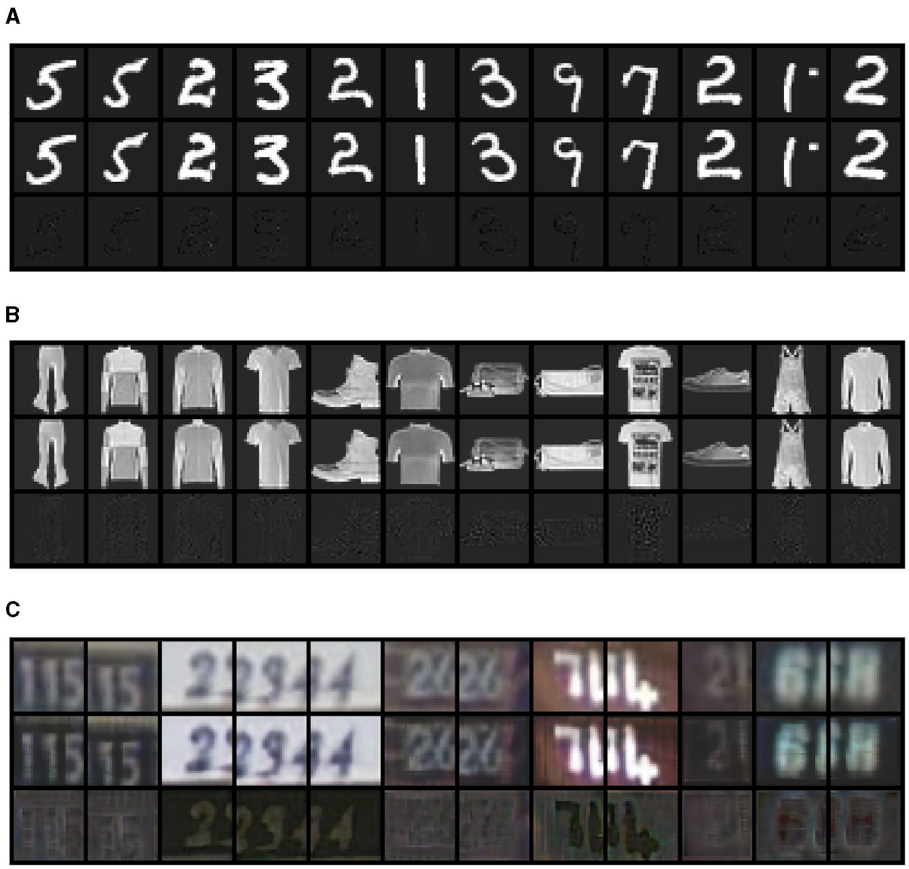


FIGURE 4
The manifold reconstruction from VGG-like Autoencoder Network on **(A)** MNST, **(B)** FASHIONMNST, and **(C)** SVHN datasets. For each dataset, we randomly sampled 12 examples. We plot the reconstructed images in the first row, the original input images in the middle row, and the difference between them in the last row.

to the input images, the reconstruction error is relatively large. We provide quantitative measures of the reconstruction quality in the Supplemental material.

4.2 Generating in-manifold perturbations

We use TNAR (Yu et al., 2019) to generate in-manifold examples. TNAR formulates the in-manifold adversarial attack as a linear optimization problem. Using power iteration and conjugate gradient algorithms, the tangent direction along the data manifold is identified. Next, a search along the tangent direction is performed to find valid L_p-norm adversarial perturbations.

Figure 5 shows the in-manifold perturbations generated using the TNAR method. Similar to commonly believed, the in-manifold perturbations are mainly "semantical". We observe that the perturbations mainly occur at the edges of the image content for datasets such as MNIST or inside the items to change their texture or details, as observed in FASHIONMNIST. In the case of SVHN, the perturbations are primarily focused on the background part of the images to reshape the meaning of the digits.

4.3 Generating normal perturbations

We extend TNAR to compute the normal direction perturbation. In the original TNAR, a single random normal direction is generated without fully exploring the vast ambient space. However, by no means, the normal space is one-dimensional. We need to explore the whole normal space to find good normal perturbations. To this end, we employ an iterative process to repeatedly generate normal vectors. Along each normal vector, we perform a search until the perturbation limit is reached. This iterative process is crucial, and it enables us to explore the whole normal space, test a broader range of perturbation patterns, and increase the chance of obtaining better normal adversarial perturbations.

Sample normal perturbations are presented in Figure 6. Consistent with our initial expectations, the normal perturbations do not directly modify the meaning of the image. Instead, they add noise to various parts of the images, effectively deceiving the classifier.

For the MNIST dataset, we observed that the normal perturbations primarily occur in the background, an area that in-manifold attacks would not typically alter. Similarly, in the FASHIONMNIST dataset, the attack expands to the background

FIGURE 5
We present the in-manifold examples in the first row, followed by the original images in the second row, and the differences are shown in the last row. Clearly, for MNIST **(A)** and FASHIONMNIST **(B)** datasets, the attacks only affect the object part. As for SVHN **(C)**, visualizing the difference between attacks on the object and the background is challenging. Nonetheless, when comparing with Figure 6, we can discern that the perturbations contain some information about the target object. For instance, in the eighth example, the attack mainly targets the object representing the number five and modifies it to be the number three. Moreover, in cases where multiple numbers are present in the image, such as the fifth example, the attack first merges the number two into the background and alters the appearance of the number six to be an eight.

areas as well. On the other hand, for SVHN, the noise covers the entire images, not restricted to the background of the digits as the in-manifold perturbations.

4.4 Validate our theoretical findings

In this section, we validate the inequality on the classifiers. We focus on L_2 normal attacks. We employ PGD attack with 40 search steps. As shown in Table 3, we report in column 1 the adversarial risk, which is the left-hand side (LHS) of Inequality 2. In columns 2, 3, and 4, we report the standard risk, in-manifold perturbation risk (evaluated on in-manifold perturbations), and normal adversarial risk (evaluated on normal perturbations). In column 5, we report their sum. Unfortunately, we have no close-form solution of the *NNR* term (the forth term in RHS). So, we know that column 5 is smaller than the actual RHS of the inequality.

Upon examining the table, we find that our theoretical findings hold for the FASHIONMNIST and SVHN datasets; the first column is smaller than the fifth column, which is smaller than the RHS. These results validate our theoretical result.

We do not observe similar trend in MNIST; the fifth column is smaller than the first column. This could be due to two potential reasons: (1) the missing term *NNR* is very large, causing the fifth column to be small while the actual RHS is still larger than LHS; (2) we underestimated the in-manifold and normal adversarial risks, as we are unable to find good quality in-manifold/normal perturbations. The second potential issue might be related to the separation of classes in MNIST.

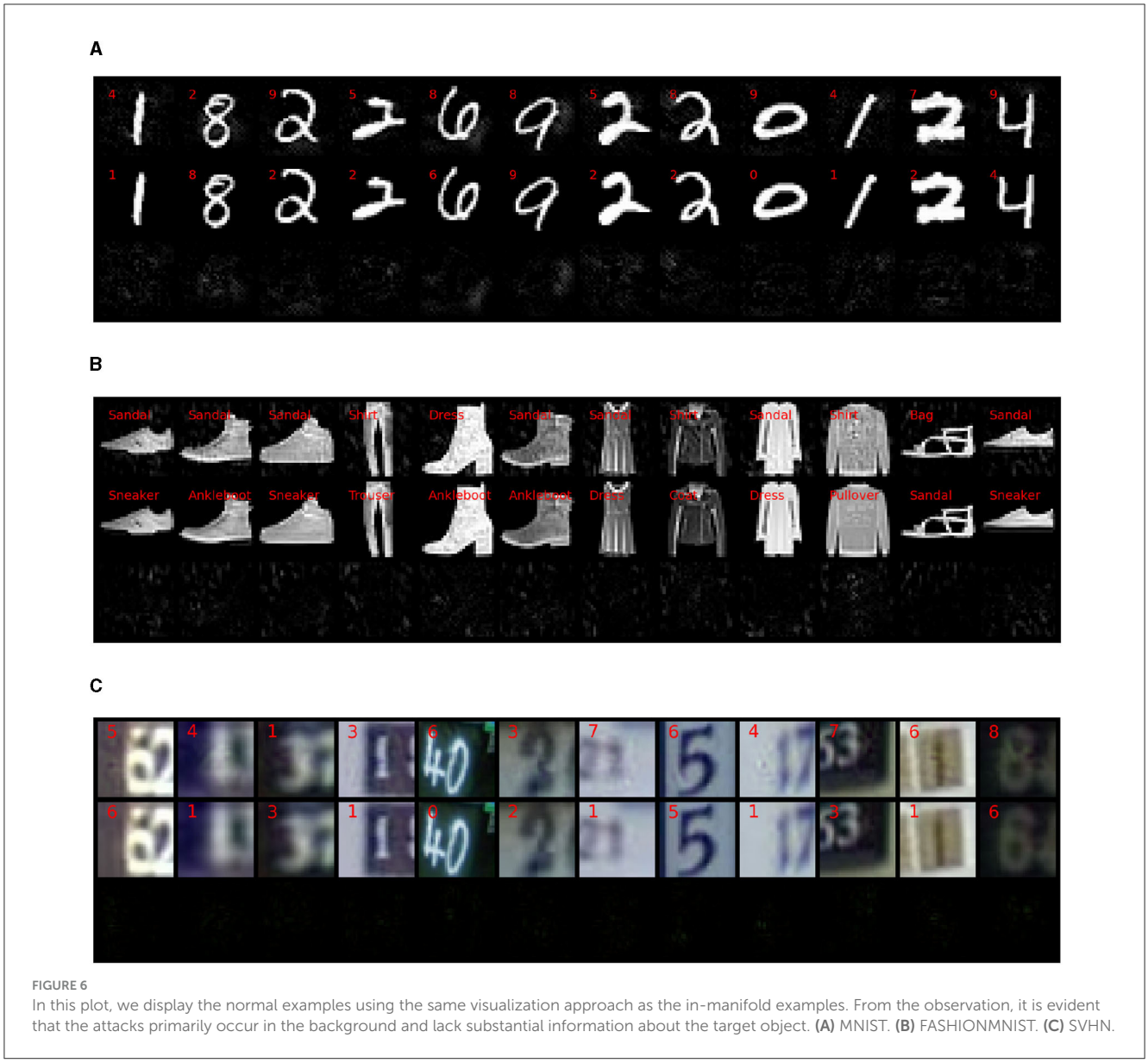


FIGURE 6
In this plot, we display the normal examples using the same visualization approach as the in-manifold examples. From the observation, it is evident that the attacks primarily occur in the background and lack substantial information about the target object. **(A)** MNIST. **(B)** FASHIONMNIST. **(C)** SVHN.

TABLE 3 **In the table, we validate our theoretical findings using L_2 norm.**

Dataset	L_2 attack risk	Standard risk	In-manifold adversarial risk	Normal adversarial risk	Sum of RHS
MNIST	0.856	0.0076	0.0702	0.5109	0.5887
FASHIONMNIST	0.98	0.0522	0.1047	0.8647	1.0216
SVHN	0.55	0.0326	0.1715	0.4783	0.6824

We report different risk terms in the Inequality 1 in separate columns. The first column (L_2 attack risk) is the adversarial risk R^{adv}, corresponding to the LHS of the inequalities. In the last column, we report RHS of 1, which is approximately the sum of the standard risk, in-manifold adversarial risk, and normal adversarial risk.

4.5 Limitations and future work

Our empirical experiments are limited to low-dimensional datasets due to the computational complexity of the TNAR algorithm, which is used to find the normal and in-manifold directions. The TNAR algorithm employs power iteration to compute the approximation of the largest eigenvector of the Jacobian matrix of the network. As the dimension of input images increases, the computation complexity of generating the normal and in-manifold directions grows quadratically. This would be costly to compute for high-resolution datasets, as the computations are performed on CPU instead of GPU. Therefore, addressing the application of our approach to high-dimensional datasets is a future direction worth exploring further.

Extending our experiments to high-dimensional datasets for future studies would provide valuable insights into the generalizability and effectiveness of our approach in real-world scenarios. Additionally, investigating the behavior of the normal and in-manifold directions in high-dimensional spaces could shed light on the robustness of the proposed method against more complex and diverse adversarial attacks.

5 Conclusion

In this study, we study the adversarial risk of the machine learning model from the manifold perspective. We report theoretical results that decompose the adversarial risk into the normal adversarial risk, the in-manifold adversarial risk, and the standard risk with the additional Nearby-Normal-Risk term. We present a pessimistic case suggesting that the additional Nearby-Normal-Risk term can not be removed in general. Our theoretical analysis suggests a potential training strategy that only focuses on the normal adversarial risk.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding authors.

Author contributions

WZ: Writing – original draft. YZ: Writing – original draft. XH: Writing – original draft. YY: Writing – review & editing. MG: Writing – review & editing. CC: Writing – review & editing. DM: Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. MG would like to acknowledge support from the US National Science Foundation (NSF) 476 awards CRII-1755791 and CCF-1910873. This material is partially based on study supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR0011-22-9-0077. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. The content of this manuscript has been presented at the AISTATS (Zhang et al., 2022).

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.

The author(s) declared that they were an editorial board member of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2023.1274695/full#supplementary-material

References

Awasthi, P., Dutta, A., and Vijayaraghavan, A. (2019). "On robustness to adversarial examples and polynomial optimization," in *Advances in Neural Information Processing Systems*, 32.

Bredon, G. E. (2013). *Topology and Geometry, Volume 139*. Cham: Springer Science & Business Media.

Carlini, N., and Wagner, D. (2017). "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (sp)* (New York, NY: IEEE), 39–57.

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. (2019). Unlabeled data improves adversarial robustness. *arXiv*.

Cayton, L. (2005). *Algorithms for Manifold Learning*. San Diego: University of California at San Diego Tech. Rep, 1.

Dey, T. K., and Goswami, S. (2006). Provable surface reconstruction from noisy samples. *Comp. Geomet.* 35, 124–141. doi: 10.1016/j.comgeo.2005.10.006

Dohmatob, E. (2019). "Generalized no free lunch theorem for adversarial robustness," in *International Conference on Machine Learning* (London: PMLR), 1646–1654.

Edelsbrunner, H., and Shah, N. R. (1994). "Triangulating topological spaces," in *Proceedings of the Tenth Annual Symposium on Computational Geometry* (New York, NY: Association for Computing Machinery), 285–292.

Fawzi, A., Fawzi, H., and Fawzi, O. (2018). "Adversarial vulnerability for any classifier," in *Advances in Neural Information Processing Systems*, 31.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., et al. (2018). Adversarial spheres. *arXiv*.

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. (2020). Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv*.

Guo, M., Yang, Y., Xu, R., Liu, Z., and Lin, D. (2020). "When nas meets robustness: In search of robust architectures against adversarial attacks," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (New York, NY: IEEE), 631–640.

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* (New York, NY: IEEE), 770–778.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Adv. Neural Inform. Proc. Syst.* 25, 1097–1105. doi: 10.1145/3065386

Lau, C. P., Liu, J., Souri, H., Lin, W.-A., Feizi, S., and Chellappa, R. (2023). Interpolated joint space adversarial training for robust and generalizable defenses. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 13054–13067. doi: 10.1109/TPAMI.2023.3286772

Levine, S., and Abbeel, P. (2014). "Learning neural network policies with guided policy search under unknown dynamics," in *NIPS* (Pittsburgh: CiteSeerX), 1071–1079.

Lin, W.-A., Lau, C. P., Levine, A., Chellappa, R., and Feizi, S. (2020). Dual manifold adversarial robustness: Defense against lp and non-lp adversarial attacks. *Adv. Neural Inform. Proc. Syst.* 33, 3487–3498.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint*.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2018). "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (New York, NY: IEEE), 7559–7566.

Narayanan, H., and Mitter, S. (2010). "Sample complexity of testing the manifold hypothesis," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems* (Red Hook, NY: Curran Associates Inc.), 1786–1794.

Niyogi, P., Smale, S., and Weinberger, S. (2008). Finding the homology of submanifolds with high confidence from random samples. *Discrete & Comp. Geomet.* 39, 419–441. doi: 10.1007/s00454-008-9053-2

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J., and Liang, P. (2020). "Understanding and mitigating the tradeoff between robustness and accuracy," in *Proceedings of the 37th International Conference on Machine Learning*, 7909–7919.

Rice, L., Wong, E., and Kolter, Z. (2020). "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning* (London: PMLR), 8093–8104.

Rifai, S., Dauphin, Y. N., Vincent, P., Bengio, Y., and Muller, X. (2011). The manifold tangent classifier. *Adv. Neural Inform. Proc. Syst.* 24, 2294–2302.

Saul, L. K., and Roweis, S. T. (2003). "Think globally, fit locally: unsupervised learning of low dimensional manifolds," in *Departmental Papers (CIS)* (Brookline, MA: Microtome Publishing), 12.

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., et al. (2019). Adversarial training for free! *arXiv*.

Shaham, U., Yamada, Y., and Negahban, S. (2018). Understanding adversarial training: increasing local stability of supervised models through robust optimization. *Neurocomputing* 307, 195–204. doi: 10.1016/j.neucom.2018.04.027

Shamir, A., Melamed, O., and BenShmuel, O. (2021). "The dimpled manifold model of adversarial examples in machine learning," in *Advances in Neural Information Processing Systems,* 30.

Stutz, D., Hein, M., and Schiele, B. (2019). "Disentangling adversarial robustness and generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (New York, NY: IEEE), 6976–6987.

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. (2018). "Is robustness the cost of accuracy?-A comprehensive study on the robustness of 18 deep image classification models," in *Proceedings of the European Conference on Computer Vision (ECCV)* (Berlin: Springer Science+Business Media), 631–648.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., et al. (2013). Intriguing properties of neural networks. *arXiv*.

Tanay, T., and Griffin, L. (2016). A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv*.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 2319–2323. doi: 10.1126/science.290.5500.2319

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. *arXiv*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Advances in Neural Information Processing Systems,* 30.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., et al. (2016). Google's neural machine translation system: bridging the gap between human and machine translation. *arXiv*.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. (2020). "Adversarial examples improve image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (New York, NY: IEEE), 819–828.

Yang, Y.-Y., Rashtchian, C., Zhang, H., Salakhutdinov, R., and Chaudhuri, K. (2020). A closer look at accuracy vs. robustness. *Adv. Neural Inform. Proc. Syst.* 33, 8. doi: 10.1007/978-3-030-63823-8

Yu, B., Wu, J., Ma, J., and Zhu, Z. (2019). "Tangent-normal adversarial regularization for semi-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (New York, NY: IEEE), 10676–10684.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019). "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning* (London: PMLR), 7472–7482.

Zhang, W., Zhang, Y., Hu, X., Goswami, M., Chen, C., and Metaxas, D. N. (2022). "A manifold view of adversarial risk," in *International Conference on Artificial Intelligence and Statistics* (London: PMLR), 11598–11614.

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 14 February 2024
DOI 10.3389/fcomp.2024.1274181

OPEN ACCESS

EDITED BY
Pavan Turaga,
Arizona State University, United States

REVIEWED BY
Chao Tong,
Beihang University, China
Henry Kirveslahti,
Swiss Federal Institute of Technology
Lausanne, Switzerland

*CORRESPONDENCE
Aaron Mahler
✉ aaron.mahler@teledyne.com
Tyrus Berry
✉ tberry@gmu.edu

RECEIVED 07 August 2023
ACCEPTED 22 January 2024
PUBLISHED 14 February 2024

CITATION
Mahler A, Berry T, Stephens T, Antil H, Merritt M, Schreiber J and Kevrekidis I (2024) On-manifold projected gradient descent. *Front. Comput. Sci.* 6:1274181. doi: 10.3389/fcomp.2024.1274181

On-manifold projected gradient descent

Aaron Mahler[1]*, Tyrus Berry[2]*, Tom Stephens[1], Harbir Antil[2], Michael Merritt[1], Jeanie Schreiber[2] and Ioannis Kevrekidis[3]

[1]Teledyne Scientific & Imaging, LLC, Durham, NC, United States, [2]Center for Mathematics and Artificial Intelligence, George Mason University, Fairfax, VA, United States, [3]Departments of Chemical and Biomolecular Engineering and Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, United States

This study provides a computable, direct, and mathematically rigorous approximation to the differential geometry of class manifolds for high-dimensional data, along with non-linear projections from input space onto these class manifolds. The tools are applied to the setting of neural network image classifiers, where we generate novel, on-manifold data samples and implement a projected gradient descent algorithm for on-manifold adversarial training. The susceptibility of neural networks (NNs) to adversarial attack highlights the brittle nature of NN decision boundaries in input space. Introducing adversarial examples during training has been shown to reduce the susceptibility of NNs to adversarial attack; however, it has also been shown to reduce the accuracy of the classifier if the examples are not valid examples for that class. Realistic "on-manifold" examples have been previously generated from class manifolds in the latent space of an autoencoder. Our study explores these phenomena in a geometric and computational setting that is much closer to the raw, high-dimensional input space than what can be provided by VAE or other black box dimensionality reductions. We employ conformally invariant diffusion maps (CIDM) to approximate class manifolds in diffusion coordinates and develop the Nyström projection to project novel points onto class manifolds in this setting. On top of the manifold approximation, we leverage the spectral exterior calculus (SEC) to determine geometric quantities such as tangent vectors of the manifold. We use these tools to obtain adversarial examples that reside on a class manifold, yet fool a classifier. These misclassifications then become explainable in terms of human-understandable manipulations within the data, by expressing the on-manifold adversary in the semantic basis on the manifold.

KEYWORDS

diffusion maps, kernel methods, manifold learning, Nyström approximation, adversarial attack, image classification, projected gradient descent

1 Introduction

Despite their superior performance at image recognition, neural network (NN) classifiers are susceptible to adversarial attack, and their performance can degrade significantly with small perturbations to the input (Szegedy et al., 2014; Tabacof and Valle, 2016; Moosavi-Dezfooli et al., 2017). The brittle performance of NNs when given novel inputs can be attributed to their intricate high-dimensional decision boundaries, which fail to generalize robustly outside of the training data. This problem is epitomized by the observation that NNs are excellent interpolators but poor extrapolaters.

Crafting attacks to deceive NNs with minimal changes to the input has been shown to be remarkably easy when the attacker has full access to the NN architecture and weights. The fast gradient sign method is one of the earliest attack methods that crafts adversarial examples by taking the sign of the gradient of the loss function to perturb the input in the direction that maximizes the loss in pixel space (Goodfellow et al., 2015). Other methods take a number of smaller steps in directions to find the smallest perturbation required to misclassify an input (Moosavi-Dezfooli et al., 2016; Carlini and Wagner, 2017; Samy Bengio, 2018; Madry et al., 2019). Most of these methods use the gradient of the NN loss function for a given input as a way to determine directions of maximal confusion, i.e., directions leading to the closest decision boundary in the high-dimensional pixel space. There also exists single-pixel attacks that use differential evolution with no gradient information and are able to reliably fool NNs (Su et al., 2019).

Various methods have been proposed to make NNs more robust to adversarial attack. Adversarial training is a common choice because it involves using attack inputs as additional training data, thereby allowing the NN decision boundary to more correctly classify that data. Commonly, the gradient of the NN will be used to augment the data set for this purpose (Goodfellow et al., 2015; Madry et al., 2019). On the other hand, gradient masking is a method that attempts to create a network that does not have useful information in the gradient, so that it cannot be exploited for creating attacks (Papernot et al., 2017). These types of networks have been found to still be vulnerable though to similar attacks that work on NNs with useful gradients (Papernot et al., 2017; Athalye et al., 2018). Defensive distillation is a gradient-free method that uses two networks, where the second network is trained using the distilled (softened) outputs of the first network (Papernot et al., 2017). Training on distilled outputs is done to create less irregular decision boundaries, which in turn results in being less prone to misclassifying small perturbations. Ensemble methods use the output of multiple models, which results in less effective attacks since it is unlikely the models are sensitive to the exact same attacks (Tramèr et al., 2020). Input preprocessing can also be applied to try to mitigate or remove adversarial perturbations. This can be done in a model agnostic way such as filtering or compressing the data (Yin et al., 2020), detecting adverasial inputs with feature squeezing (Xu et al., 2018), or using an autoencoder to denoise the input (Cho et al., 2020).

One popular method of adversarial training uses small steps along the network gradient that are only allowed to step so far away from the original input, called projected gradient descent (PGD) (Madry et al., 2019). The data set is augmented with examples that are maximally confusing to the NN during training, but the augmented data points are only allowed to be ϵ far away from true data points. This results in a marked improvement to the NN when it is attacked with perturbations of the same strength. However, PGD trained networks show a decrease in accuracy on clean inputs and the accuracy goes down as the size of the ϵ-ball allowed for augmentations increases (Engstrom et al., 2019). The degradation of accuracy and the rise of robustness could be due to several factors, such as overfitting the model to adversarial examples or from adversarial examples that are not actually representative of the class of the input that was perturbed. The trade-off between robustness and accuracy has been noted to occur with many flavors of adversarial training, and it has even been conjectured that robustness and accuracy may be opposed to each other for certain NNs (Su et al., 2018; Tsipras et al., 2019).

On the other hand, it has also been shown that in some cases, a more careful choice of adversarial examples can create robust NNs that are also on-par with standard networks at generalizing to unseen validation data (Stutz et al., 2019). This was explained by the fact that adversarial training such as PGD creates samples that are not truly on the manifold of that data's class label. The NN is then tasked with learning a decision boundary for the training data as well as randomly noisy data, resulting in the compromise between accuracy and robustness for those types of adversarial training. In Stutz et al. (2019), they found perturbations in a latent space learned from the training data. Perturbing in an ϵ ball in the latent space was surmised to be on a class manifold and therefore a new augmentation that was representative of that class. Adversarial training in a way that is agnostic to the underlying geometry of the data itself therefore seems to be a root cause for the trade-off between robustness and accuracy.

The above mitigations to adversarial attacks all proceed from the perspective that the neural network has simply not been fed enough variation in the collected training data to learn decision boundaries that adhere to the full underlying data manifolds. From this perspective, injecting adversarial examples into the original training data "pushes out" incursions by the decision boundary into the true manifold. An alternative perspective is that the adversarial examples do not result from so-called bugs in the decision boundaries but are instead features of the data (Ilyas et al., 2019). From the features perspective, adversarial training is a data cleaning process; the original data has pixel correlations across classes that our eyes cannot detect, and computed adversarial perturbations act to wash those away. While we do not embark on our applications from this perspective, the mathematical tools developed here are ideally suited for extending their hypotheses and results.

Our application of on-manifold adversarial training connects the learning problem to the manifold hypothesis and manifold modeling techniques. For natural images, the manifold hypothesis suggests that the pixels that encode an image of an object, together with the pixel-level manipulations that transform the scene through its natural, within-class variations (rotations, articulations, etc), organize along class manifolds in input space. In other words, out of all possible images drawn from an input space, while the vast majority look like random noise, the collection of images that encode a recognizable object (a tree, a cat, or an ocean shoreline) are incredibly rare, and the manifold hypothesis claims that those images should be distributed throughout input space along some coherent geometric structure. On-manifold adversarial training aims for an NN to better capture the underlying structure in the data. In this study, we use novel manifold modeling techniques that do not rely on autoencoders or black box neural networks.

To construct and leverage a data-driven representation of an embedded manifold, we use a collection of tools that have grown out of the diffusion maps-based methods in the manifold learning community. Diffusion maps methods are based on learning a manifold by estimating a certain operator called

the Laplace–Beltrami operator, which encodes all the geometric properties of the manifold. The advantage of this approach is that is does not require constructing a simplicial complex (or triangularization) from data, which can be quite challenging. However, diffusion maps-based manifold learning requires several additional tools to access needed geometric quantities for on-manifold adversarial learning. In particular, we need the ability to find the tangent directions to the manifold, walk along a tangent direction, and then project down onto the manifold. To find the tangent directions to the manifold, we use a recent development known as the Spectral Exterior Calculus (SEC) that builds these estimators directly from the diffusion maps constructions. The SEC uses global information to find the principal tangent directions, and is thus less susceptible to noise and large ambient dimensions than local (nearest neighbor)-based methods. Finally, to project a point from data space onto the manifold, we discover a surprising and powerful new tool which we call the *Nyström projection*. Using these methods, we demonstrate creating on-manifold adversarial examples that are explainable in terms of their semantic labels.

1.1 Manifold learning and CIDM

Manifold learning emerged as an explanation for how kernel methods were able to perform regressions and identify low-dimensional coordinates from much higher dimensional data sets than would be possible according to normal statistics. Assuming that the data were lying on a submanifold of the data space, it appeared that the kernel methods (kernel regression, kernel PCA, etc.) were able to leverage this intrinsically low-dimensional structure.

The first advance in understanding this effect rigorously was Laplacian eigenmaps (Belkin and Niyogi, 2003). They employed a Gaussian kernel to build a complete weighted graph on the data set with weights $K_{ij} = k(x_i, x_j) = \exp(-||x_i - x_j||^2/(2\epsilon^2))$. This was a very common choice of radial basis kernel at the time and a natural first choice for analysis. Laplacian Eigenmaps then constructs the weighted graph Laplacian $L = \frac{D-K}{\epsilon^2}$, where diagonal matrix $D_{ii} = \sum_j K_{ij}$ is called the degree matrix. In the limit, as the number of data points goes to infinity, the Laplacian matrices become larger and larger, and if the bandwidth, ϵ, is taken to zero at an appropriate rate, this sequence of matrices are shown to converge to the Laplace–Beltrami operator on the manifold that the data were sampled from. This was the first rigorous connection between the somewhat *ad hoc* kernel methods and the intrinsic geometry of the data.

Unfortunately, the assumptions required to prove the key theorem of Laplacian eigenmaps were overly restrictive in practical settings. In addition to only applying to a single kernel function (when in practice, many different kernel functions were known empirically to have similar behavior), Laplacian eigenmaps also required the data to be sampled uniformly from the underlying manifold. This is a somewhat technical requirement, an embedded manifold (such as the one the data are assumed to lie on), that inherits a natural measure from the ambient data space which is called the *volume form*. We can think of this volume form as a distribution, and when the data are sampled from this natural distribution, it is called *uniformly* sampled. However, there is no reason for the data to have been be uniformly collected in this sense. For example, if your data lie on a unit circle, there is no reason that the data could not be more densely collected on one side of the circle and more sparsely collected on the other side, but Laplacian eigenmaps did not allow for this in their theorem. These restrictions meant that the applicability of kernel methods to resolving the intrinsic geometry of a real data set was still seen as rather tenuous.

Diffusion maps (Coifman and Lafon, 2006a) resolved these concerns and solidified the connection between a large class of kernel methods and the intrinsic geometry of the data. The idea of diffusion maps turns out to be fairly simple, although the technical details of the theorems are somewhat challenging. The key idea is that the degree matrix, $D_{ii} = \sum_j k(x_i, x_j)$, is actually a classical kernel density estimator, meaning that if the data are not sampled uniformly, then D_{ii} will be proportional to the true sampling density (up to higher order terms in ϵ which can be carefully accounted for). Diffusion maps begins by generalizing the kernel density estimation results to data sampled on manifolds, and then uses the estimated density to de-bias the kernel. De-biasing the kernel turns out to be a simple normalization procedure called the *diffusion maps normalization*, which constructs the normalized kernel,

$$\hat{K} = D^{-1}KD^{-1}$$

and then recomputes the new degree matrix $\hat{D}_{ii} = \sum_j \hat{K}_{ij}$ and finally the diffusion maps graph Laplacian $\hat{L} = \frac{\hat{D}-\hat{K}}{\epsilon^2}$. The diffusion maps theorems showed that for any radial basis kernel that had exponential decay in distance, and for data collected from any smooth distribution on the manifold, their new graph Laplacian, $\hat{L}$, converged to the Laplace–Beltrami operator on the manifold. Moreover, the diffusion maps theorems also showed (although this was only realized in later works, e.g., Berry and Sauer, 2016) that even when their normalization was not used, the classical graph Laplacian converged to a Laplace–Beltrami operator with respect to a conformal change of metric. This ultimately showed that all kernel methods with radial basis kernels that had fast decay were finding the intrinsic geometry of the data (possibly up to a change of measure). Later works would generalize the diffusion maps theorems to include all kernels that had sufficiently fast decay in the distance between points (so not just radial basis functions) (Berry and Sauer, 2016).

At this point, we should address why both Laplacian eigenmaps and diffusion maps have the word "Maps" in them. This goes back to the motivation that was driving the development of these new theories. In particular, both methods were motivated by Kernel PCA, which interpreted the eigenvectors of the kernel matrix as providing the coordinates of a mapping into a new space, often called a 'feature space'. Ironically, this mapping interpretation arose from the theory of Reproducing Kernel Hilbert Spaces, where the kernel induces a map into a *function* space (not a Euclidean space). However, since the kernel matrix, K, has as many rows and columns as there were data points, the eigenvectors of the kernel matrix have as many entries as there are data points, so inevitably these were visualized and interpreted as new coordinates. Diffusion maps and Laplacian eigenmaps were trying to show

that this "mapping" preserved intrinsic aspects of the geometry while also reducing dimension, and while the first part is partially correct, the dimensionality reduction aspect of the diffusion maps turns out to not be guaranteed. However, this was merely a case of applying the wrong interpretation to the results. In fact what diffusion maps had proven was much better than any fact about a mapping. By recovering the Laplace–Beltrami operator on the manifold, and its eigenfunctions, diffusion maps unlocked the door and allowed access to every single aspect of the geometry of the data. Moreover, the eigenfunctions provide a generalized Fourier basis for analysis of functions and operators on the data set, and have been used in regression, interpolation, forecasting, filtering, and control applications.

To leverage the opening that diffusion maps has created to learning manifold geometry from data, we will need several recent advances that improve and apply the original theory. First, it turns out that for real data sets in high dimensions, the fixed bandwidth kernels discussed so far have difficulty adjusting to large variations in sampling density. To compensate for this, a variable bandwidth kernel is needed (Berry and Harlim, 2016), which can automatically adjust to have a small bandwidth and high resolution in areas of data space that are densely sampled, while keeping a large bandwidth and a lower resolution representation of sparsely sampled regions of data space. The ultimate evolution of the variable bandwidth kernels is the *Conformally Invariant Diffusion Map* (CIDM) (Berry and Sauer, 2016, 2019), which we introduce in Section 1.1.1.

The next tool we will need is a rigorous method for extending/interpolating all of the discrete representations of functions, mappings, and operators to be able to operate on any new input data. Here, we use a regularized version of a standard method called the Nyström extension, introduced in Section 1.1.2. Although this basic method of interpolation is well-established, we will apply it in ways that have never been considered before to achieve powerful new methods and results.

Finally, we mentioned above that the Laplace–Beltrami operator unlocks the door to access all the hidden geometry of the data. This is due to a technical result which says that if you know the Laplace–Beltrami operator, you can recover the Riemannian metric on the manifold, and the Riemannian metric completely determines all aspects of the geometry (from dimension and volume to curvature to geodesics and everything in between). However, until recently, this was a purely abstract possibility, and there was no actual method for constructing these geometric quantities starting from the Laplace–Beltrami operator. This was achieved in 2020 with the creation of the Spectral Exterior Calculus (SEC), which re-builds all of differential geometry starting just from the Laplace–Beltrami operator. While we will not require every aspect of the SEC here, the basic philosophy of its construction will be fundamental to the way that we will construct vector fields and in a particular tangent vectors on the manifold, so a brief introduction will be given in Appendix A.1.

1.1.1 Conformally invariant diffusion map

As mentioned above, the original version of diffusion maps uses a fixed bandwidth kernel of the form $J(x,y) = h(||x-y||^2/\epsilon^2)$. Here, h is called the shape function and is assumed to decay quickly to zero as the input (distance) goes to infinity. A typical choice for h is the exponential function $h(z) = \exp(-z)$, so moderate differences in distances leads to large differences in the values of h. This becomes particularly problematic in terms of the distance to the nearest neighbors. If the distances from a data point to its nearest neighbors are large (relative to the bandwidth ϵ), then the values of the kernel become very close to zero. This means that even though our weighted graph is technically still connected, the weights are so close to zero that it becomes numerically disconnected, which causes the diffusion map to interpret such data points as disconnected from the rest of the data set. On the other hand, we want the kernel function to decay quickly beyond the nearest neighbors to localize the analysis and make the resulting kernel matrix approximately sparse.

When the density of points varies widely, it becomes very difficult to find a single bandwidth parameter ϵ which achieves these two goals across the data set. One tends to have to choose the bandwidth large enough to connect with the sparsest region of data, and this large bandwidth value results in a loss of resolution in the denser sampled regions. This trade-off is examined rigorously in Berry and Harlim (2016), which introduces variable bandwidth kernels and generalizes the diffusion maps expansions for such kernels. The best practical implementation of a variable bandwidth approach was introduced in Berry and Sauer (2019), which is a variable bandwidth version of the Conformally Invariant Diffusion Map (CIDM) that was introduced in Berry and Sauer (2016).

The CIDM starts by re-scaling the distance using the distances to the nearest neighbors, namely

$$\delta(x,y) \equiv \frac{d(x,y)}{\sqrt{d(x,\text{kNN}(x))d(y,\text{kNN}(y))}}$$

Where kNN returns the k-th nearest neighbor of the input point from the training data set. Note that the distance to the k-th nearest neighbor is a consistent estimator of the density to the power of $-1/d$ where d is the dimension of the manifold. Thus, when the local density is high, the distance to the kNN will be small, and conversely when the local density is sparse, the distance to the kNN will be large. Thus, $\delta(x,y)$ has re-scaled the distances into a unit-less quantity which will be on the same order of magnitude for the k-th nearest neighbors of all the data points.

Inside the kernel, we will use the square of this quantity,

$$\delta(x,y)^2 = \frac{d(x,y)^2}{d(x,\text{kNN}(x))d(y,\text{kNN}(y))}$$

which is also more convenient for derivatives. Next, we use the dissimilarity δ in a kernel,

$$k(x,y) \equiv h(\delta(x,y)^2/\epsilon^2)$$

Where ϵ is a global bandwidth parameter and $h:[0,\infty) \to [0,\infty)$ is a called a shape function (examples include $h(z) = e^{-z}$ as mentioned above or even simply the indicator function $h(z) = 1_{[0,1]}(z)$). We can then build the kernel matrix $K_{ij} = k(x_i, x_j)$ on the training data set, and the diagonal degree matrix $D_{ii} = \sum_j K_{ij}$ and the normalized graph Laplacian $L \equiv I - D^{-1}K$.

We should note that in Berry and Sauer (2019), it was shown that, uniquely for the CIDM, the unnormalized Laplacian $L_{\text{un}} \equiv D - K$ has the same limit as the normalized Laplacian in the limit of large data; however, the normalized Laplacian, L, has some numerical advantages. Numerically, it is advantageous to maintain the symmetry of the problem by finding the eigenvectors of the similar matrix, $L_{\text{sym}} \equiv D^{1/2}LD^{-1/2} = I - D^{-1/2}KD^{-1/2}$. Finally, we are interested in the smoothest functions on the manifold, which are the minimizers of the energy defined by L; however, it is easier to find the largest eigenvalues of $K_{\text{sym}} \equiv D^{-1/2}KD^{-1/2}$ (Recall that maximal eigenvalues can be found with power iteration methods which are fast than the inverse power iterations required for finding smallest eigenvalues). Once we have computed the eigenvectors $K_{\text{sym}}\vec{v} = \lambda\vec{v}$, then it is easy to see that $\vec{\phi} = D^{-1/2}\vec{v}$ are eigenvectors of $D^{-1}K$ with the same eigenvalues, and $\vec{\phi}$ are also eigenvectors of L with eigenvalues $\xi = 1-\lambda$. Thus, when λ are the largest eigenvalues of K_{sym}, the corresponding ξ will be the smallest eigenvalues of L.

We will refer to L as the CIDM Laplacian, and we will use the eigenvectors and eigenvalues of L to represent the geometry of the data manifold. The eigenvectors of the CIDM Laplacian, $L\vec{\phi} = \lambda\vec{\phi}$ are vectors of the same length as the data set, so the entries of these eigenvectors are often interpreted as the values of a function on the data set, namely $\phi(x_i) = \vec{\phi}_i$. Of course, we have not really defined a function ϕ since we have only specified its values on the data set. However, in the next section, we will show how to define a function ϕ on the whole data space that takes the specified values on data set. This method is called the *Nyström extension* because it extends the function from the training data set to the entire data space.

In Berry and Sauer (2016), the CIDM Laplacian, L, was shown to converge (in the limit of infinite data and bandwidth going to zero) to the Laplace–Beltrami operator of the hidden manifold with respect to a conformal change of metric that has volume form given by the sampling density. The Laplace–Beltrami operator encodes all the information about the geometry of the manifold (see Appendix A for details), which is why methods such as diffusion maps and the CIDM are called *manifold learning* methods. Moreover, it was shown in Berry and Harlim (2016) and Berry and Sauer (2016, 2019) that the CIDM construction using the k-nearest neighbors density estimator, as described above, does not require the so-called "Diffusion Maps normalization". The CIDM gives the unique choice of conformal geometry for which a standard unnormalized graph Laplacian is a consistent estimator of a Laplace–Beltrami operator (Berry and Sauer, 2019). Empirically, we have found that this variable bandwidth kernel construction is much more robust to wide variations in sampling density.

We should note that in the Nyström extension section below, we will make use of the following normalized kernel,

$$\hat{k}(x,y) = \frac{k(x,y)}{\sum_{i=1}^{N} k(x,x_i)} = \frac{h(\delta(x,y)^2/\epsilon^2)}{\sum_{i=1}^{N} h(\delta(x,x_i)^2/\epsilon^2)}$$

since this corresponds to $D^{-1}K$ as discussed above [namely if $K = k(x_i,x_j)$, then $\hat{k}(x_i,x_j) = (D^{-1}K)_{ij}$]. Finally, to reduce sensitivity, we often use the average of the distances to the k-nearest neighbors in the re-scaling, so the dissimilarity would then be,

$$\delta(x,y)^2 = \frac{d(x,y)^2}{\frac{1}{k}\sum_{i=1}^{k} d(x,\text{iNN}(x))\,\frac{1}{k}\sum_{j=1}^{k} d(y,\text{jNN}(y))}$$

Where iNN refers to the i-th nearest neighbor so the summations are averaging the distances to the k-nearest neighbors.

1.1.2 Nyström extension: interpolation and regularization

In this section, we introduce the Nyström extension, which is the standard approach for extending diffusion maps eigenfunctions (and thus the "diffusion map") to new data points. Once the eigenfunctions can be extended, arbitrary functions can also be extended by representing them in the basis of eigenfunctions; this approach can be used to extend any sufficiently smooth function to new data points in input space. Since, in practice, we can only represent a function with finitely many eigenfunctions, the truncation onto this finite set gives us a regularized, or smoothed, regression. The Nyström extensions of the diffusion maps eigenfunctions are sometimes called *geometric harmonics* (Coifman and Lafon, 2006b) and these out-of-sample extensions are related to the method of Kriging in Gaussian Processes, a connection which is explored in Dietrich et al. (2021).

Given an eigenvector $K\vec{\phi} = \lambda\vec{\phi}$ of a kernel matrix $K_{ij} = k(x_i,x_j)$, we can extend the eigenvector to the entire input space by,

$$\phi(x) \equiv \frac{1}{\lambda}\sum_{j=1}^{N} k(x,x_j)(\vec{\phi})_j \tag{2}$$

which is called the Nyström extension. Note that here k is an abstract kernel which may incorporate CIDM normalizations inside the shape function as well as normalization such as the diffusion maps normalization and/or Markov normalization outside of the shape function. For example, for CIDM, the Nyström extension of an eigenfunctions is,

$$\phi(x) = \frac{1}{\lambda}\sum_{j=1}^{N} \hat{k}(x,x_j)\phi(x_j) = \frac{\sum_{j=1}^{N} h(\delta(x,x_j)^2/\epsilon^2)\phi(x_j)}{\lambda\sum_{i=1}^{N} h(\delta(x,x_i)^2/\epsilon^2)}$$

Where the CIDM kernel $\hat{k}$ takes the place of the abstract kernel k in Equation (2). Notice that evaluating $\hat{k}$ involves computing the dissimilarity δ between arbitrary point x and a training data point x_j, which in turn requires finding the k nearest neighbors of the point x from the training data set. Thus, evaluating the abstract kernel k may actually depend on the entire training data set; however, in this section, we will consider the training data set as fixed and treat its influence on k as hidden parameters that define the kernel k. We should note that although everything in this section can be applied to any kernel, a simple radial basis function kernel with no normalizations and a fixed bandwidth has fairly poor performance for the off-manifold extensions we will discuss later in this section.

A key property of the Nyström extension is that on the training data points, we have

$$\phi(x_i) = \frac{1}{\lambda}\sum_{j=1}^{N} k(x_i,x_j)(\vec{\phi})_j = \frac{1}{\lambda}(K\vec{\phi})_i = \frac{1}{\lambda}(\lambda\vec{\phi})_i = (\vec{\phi})_i.$$

So if we interpret $(\vec{\phi})_i$ as the value of a function on the data point x_i, then the Nyström extension agrees with these function

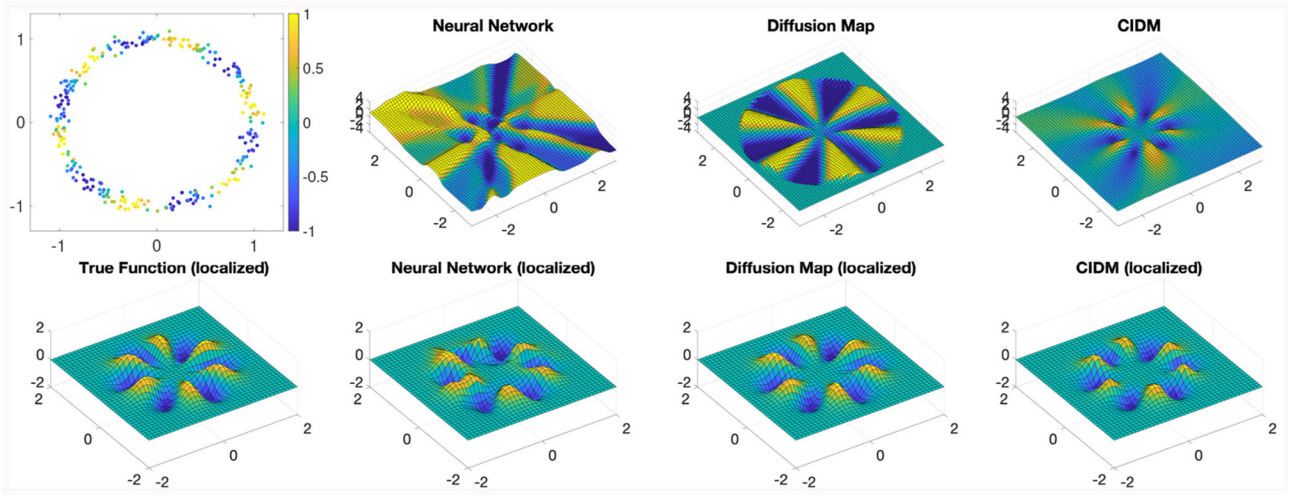


FIGURE 1
Nyström extension comparison. Consider data points near a unit circle **(top left)** and a function to learn given by the color (also shown **bottom left**) localized near the unit circle. We consider three methods of learning the function, a simple 2-layer neural net, the standard diffusion map Nyström extension, and the CIDM Nyström extension. Each extension is shown on a large region **(top row)** as well as localized near the unit circle **(bottom row)**. The CIDM provides the smoothest extension to the entire input space.

values on the original data set and extends the function to the entire input space.

Moreover, given an arbitrary vector of function values $\vec{f}_i$ on the data set, we can extend this function to the entire data set by representing $\vec{f}_i$ in the basis of eigenvectors and then applying the Nyström extension to these eigenvectors. Let $\{\phi_\ell\}_{\ell=1}^N$ be the collection of eigenvectors of the kernel matrix K. Note that $(\vec{\phi}_\ell)_i$ will refer to the i-th entry of the ℓ-th eigenvector. Notice that,

$$\vec{f} = \sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle \vec{\phi}_\ell$$

so if we replace the vector $\vec{\phi}_\ell$ with the Nyström extension, we have the Nyström extension of f given by

$$f(x) = \sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle \phi_\ell(x) = \sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle \frac{1}{\lambda_\ell} \sum_{j=1}^{N} k(x, x_j)(\vec{\phi}_\ell)_j.$$

Notice that this can be viewed as a kernel extension of f by rewriting the above as

$$f(x) = \sum_{j=1}^{N} k(x, x_j) \left(\sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle \frac{1}{\lambda_\ell} (\vec{\phi}_\ell)_j \right)$$

In other words, the Nyström extension of a function is given by $f(x) = \sum_{j=1}^{N} k(x, x_j) c_j$, which is a linear combination of the kernel basis functions $k(\cdot, x_i)$, with coefficients $c_j \equiv \sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle \frac{1}{\lambda_\ell} (\vec{\phi}_\ell)_j$. This formula can be truncated for $\ell = 1, ..., L$, with $L < N$ to get a smoothed, low-pass filtered version of the function. When all of the eigenvectors are used, we have

$$f(x_i) = \sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle \frac{1}{\lambda_\ell} \sum_{j=1}^{N} k(x_i, x_j)(\vec{\phi}_\ell)_j = \sum_{\ell=1}^{N} \left\langle \vec{f}, \vec{\phi}_\ell \right\rangle (\vec{\phi}_\ell)_i = \vec{f}_i$$

So again the Nyström extension agrees with the original vector of function values on the original data points. When fewer than N eigenfunctions are used, the Nyström extension is a smoothing of the original function, so it does not interpolate the values on the training data, which can be useful for de-noising. Finally, if we substitute in the definition of the vector inner product, we have the following expression for the Nyström extension:

$$f(x) = \sum_{j=1}^{N} k(x, x_j) \left(\sum_{\ell=1}^{L} \frac{1}{\lambda_\ell} (\vec{\phi}_\ell)_j \sum_{i=1}^{N} \vec{f}_i (\vec{\phi}_\ell)_i \right)$$

Where L is the number of eigenfunctions used and is typically much less than N to smooth and denoise the function.

2 Methods

Here, we introduce some tools for analyzing data on manifolds in the input data space. The first tool is a novel method of projecting arbitrary data points non-linearly down onto the manifold. This method is based on using the Nyström extension to build a projection, so we call this new method the *Nyström projection* in Section 2.1. The next tool is the Spectral Exterior Calculus (SEC), which was developed in Berry and Giannakis (2020) and is able to identify vector fields that respect the global structure of the data. Here, we overview the interpretation of the SEC on vector fields in Section 2.2 and describe how we use these vector fields to approximate the tangent space to the manifold in a way that is more robust than local linear methods. Together, by using linear projection of vectors (such as perturbation directions) onto the tangent space and the non-linear Nyström projection of perturbations of data points down onto the manifold itself, we introduce an on-manifold technique for projected gradient descent in Section 2.3.

2.1 The Nyström projection: mapping off-manifold points onto the manifold

In Section 1.1.2, we reviewed the Nyström extension as an established method of out-of-sample extension for functions on the data space. In this section, we introduce a novel application of this extension, which we will call the *Nyström projection*. The Nyström projection will be defined below as the Nyström extension of the original data coordinate functions. This special case is deserving of extra scrutiny because it is the only function which maps a new data point through the diffusion maps embedding and back into the original data space, meaning that the Nyström projection can be iterated with surprising and useful results.

The next (and crucial) question is: How does the Nyström extension perform out-of-sample? In the case of manifold learning, this question has two cases, first, when the out-of-sample data lie on the manifold, and, second, when they are off the manifold (and potentially far from the manifold). For data points on the manifold, the behavior of Nyström is well-understood as a band-limited interpolation of the function f, which minimizes a certain cost function. The on-manifold out-of-sample interpretation is easy because we started by assuming that there was a given function on the manifold and that we had sampled values of that function on our in-sample training data. Thus, there is a natural "true" function in the background to compare our Nyström interpolation to.

The case of extension to off-manifold points is much more interesting and less is known about this case. Clearly, for any fixed "true" function defined on the manifold, there will be infinitely many smooth extensions to the entire space, so the Nyström extension is selecting one of these extensions, and in the limit of infinitely many data points and eigenfunctions, this extension is minimizing a certain functional. While this is an open area of research, empirically, we observe that for a normalized CIDM kernel, the Nyström extends the function to an off-manifold data point by essentially taking the function value of the nearest point on the manifold. Since almost every point in the ambient data space has a unique nearest point on the manifold, this is well-defined up to a set of measure zero, and, in practice, there is a smoothing effect in a neighborhood of this measure zero set; however, we will ignore these effects for simplicity.

To demonstrate empirically how the Nyström extension performs far from the training data set, in Figure 1 we show an example of a data set lying near the unit circle in the plane. Given a simple smooth function, shown in the first panel of Figure 1, we can use various methods to learn this function and attempt to extend it to the entire input data space (the plane in this case). Notice that when well-tuned, performance near the training data set, shown by the "localized" panels of Figure 1, is comparable for a simple two-layer neural network as well as the Nyström extension with both the standard diffusion maps kernel and the CIDM kernel. However, Figure 1 shows that these methods have very different behavior far from the data set, with the neural network behaving somewhat unpredictably, and the standard diffusion map kernel having difficulty extrapolating when far from the training data, whereas the CIDM makes a smooth choice of extension which is well-defined even very far from the training data.

This interpretation of the Nyström extension as taking the value of the nearest point on the manifold is critical since it lead us to a novel and powerful method of achieving a non-linear projection onto the manifold. The idea is actually quite simple, think of the original data set as a function on the manifold and build the Nyström extension of this function. In fact, this is how we often think of a data set mathematically in the manifold learning literature. Thus, we will apply the Nyström extension of the original data coordinates into a function on the entire data space, and we call the resulting function the *Nyström projection*.

While it is perfectly valid to consider the data manifold as a subset of the ambient data space, $\mathcal{M} \subset \mathbb{R}^n$ in differential geometry, it is useful to think of an abstract manifold $\mathcal{N}$ that is simply an abstract set of points, and then think of the data set as the image of this abstract manifold under an embedding into Euclidean space, so $\iota : \mathcal{N} \to \mathcal{M} \subset \mathbb{R}^n$. Now, the points in data space, $x_i \in \mathbb{R}^n$, are the images of an abstract point $\tilde{x}_i \in \mathcal{N}$, such that $\iota(\tilde{x}_i) = x_i$. In this interpretation, each of the coordinates of the data are actually scalar valued component functions of the embedding function, so $(x_i)_s = \iota_s(\tilde{x}_i)$, where $\iota_s : \mathcal{N} \to \mathbb{R}$ are the component functions of the embedding. Of course, since we know the value of these coordinate functions on the training data set, we can apply the Nyström extension to each of the ι_s functions, and extend the entire ι embedding map to the entire data space. In this way, we obtain the Nyström projection $\tilde{\iota} : \mathbb{R}^n \to \mathbb{R}^n$, which is given by

$$\tilde{\iota}_s(x) = \sum_{j=1}^{N} k(x, x_j) \left(\sum_{\ell=1}^{L} \frac{1}{\lambda_\ell} (\vec{\phi}_\ell)_j \sum_{i=1}^{N} (\vec{x}_i)_s (\vec{\phi}_\ell)_i \right)$$

Where $(\vec{x}_i)_s$ is the s-th coordinate of the i-th data point. We can also write the Nyström projection more compactly in terms of the Nyström extension of the eigenfunctions, as

$$\tilde{\iota}(x) = \sum_{\ell=1}^{L} \hat{x}_\ell \phi_\ell(x)$$

Where $\hat{x}_\ell = \left\langle \vec{x}, \vec{\phi}_\ell \right\rangle$ is a vector-valued generalized Fourier coefficient given by $(\hat{x}_\ell)_s = \left\langle (\vec{x})_s, \vec{\phi}_\ell \right\rangle = \sum_{i=1}^{N} (\vec{x}_i)_s (\vec{\phi}_\ell)_i$. Thus, we can think of $\hat{x}$ as encoding the embedding function that takes the abstract manifold to the realized data coordinates.

In fact, $\tilde{\iota}$ does much more than the original embedding function, since it extends the embedding function to the entire data space. This is because when input data points are off-manifold, the Nyström extension of a function is well-approximated by selecting the values of the function for the nearest point on the manifold. Since we are applying the Nyström extension to the embedding function itself, this means that for an off-manifold point, the Nyström projection $\tilde{\iota}$ will actually return the coordinates of the nearest point on the manifold. In other words, Nyström projection $\tilde{\iota}$ acts as the identity for points on the manifold and projects off-manifold points down to the nearest point on manifold. This novel yet simple tool gives us a powerful new ability in manifold learning and has opened up several promising new research directions. Moreover, the choice of L in the Nyström projection gives us control over the resolution of the manifold we wish to project on. Thus, for a noisy data, we can intentionally choose a smaller L value to project down through the noise to a manifold that cuts through the noisy data. This is demonstrated in Figure 2 (rightmost

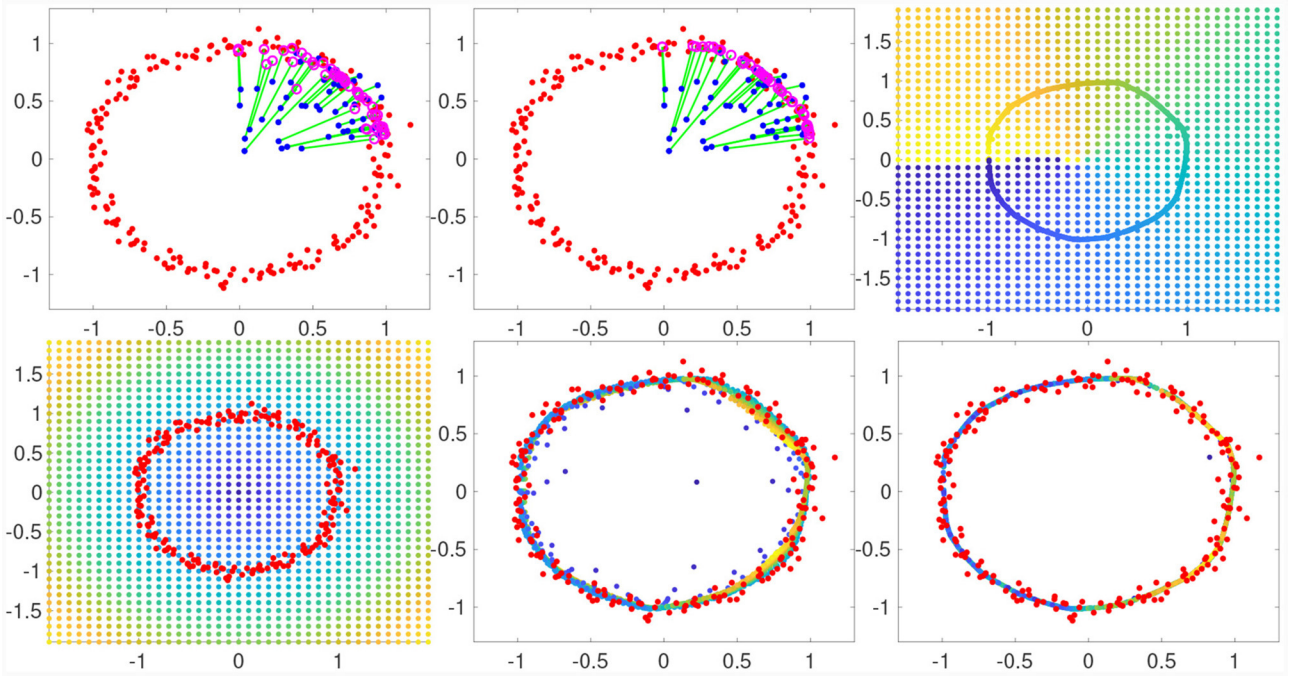


FIGURE 2
Nyström projection onto the Unit Circle. We use the Nyström projection to project points on the plane onto the unit circle using a noisy training data set (red) to learn the manifold. **Top:** After learning from the red training data set, blue data points far from the manifold are projected onto the magenta points using a single iteration of the Nyström projection **(left)** and two iterations **(middle)**; the green line connects each initial point to its Nyström projection. Top Right: Applying two iterations to a grid in the plane projects onto a circle, the original grid points are colored by the angle computed after they are projected to show where they land. **Bottom Row:** A grid colored by radius **(left)** is projected once **(middle)** and twice **(right)** using the Nyström projection learned from the same data set (red) as the top row.

panels), where we set $L = 20$ and recovered a smooth circle that cuts through the noisy input data set (red points).

It is useful to formulate the Nyström projection as a composition of a non-linear map (related to the Diffusion Map) and a linear map. Often, we consider the map which takes an input point in data space and returns the coordinates of the first L eigenfunctions, $\Phi : \mathbb{R}^n \to \mathbb{R}^L$ given by $\Phi(x) = (\phi_1(x), ..., \phi_L(x))^\top$. This is the so-called *diffusion map* (with $t = 0$). While we have argued above that this is not necessarily the best embedding of the data, it is useful to express the projection we have just constructed. Note that the $\hat{x}$ is an $n \times L$ matrix containing the first L generalized Fourier coefficients of each of the n coordinates of our data set. Thus, $\hat{x}$ defines a linear map $\hat{X} : \mathbb{R}^L \to \mathbb{R}^n$ from $\mathbb{R}^L$ (the image of Φ) back to the data space, $\mathbb{R}^n$ (where $\hat{X}$ is simply given by left multiplication by the matrix $\hat{x}$). The Nyström projection onto the manifold is the composition of these two maps, namely $\tilde{\imath} = \hat{X} \circ \Phi$, so that we have a map,

$$x \mapsto_\Phi (\phi_1(x), ..., \phi_L(x))^\top \mapsto_{\hat{X}} \tilde{\imath}(x)$$

such that $\tilde{\imath} \equiv \hat{X} \circ \Phi : \mathbb{R}^n \to \mathcal{M} \subset \mathbb{R}^n$ and $\hat{X} \circ \Phi\big|_{\mathcal{M}} = \mathrm{Id}_{\mathcal{M}}$.

The fact that the Nyström projection maps back into the original data space has the significant consequence that it actually forms a dynamical system which can be iterated. Moreover, we have seen that its behavior can be extremely useful in that it tends to project data points "down onto onto the manifold". This feature, that applying (and iterating when necessary) the Nyström projection pushes new data points down toward the manifold inside of the ambient data space, is what allows us to search for on-manifold adversaries.

Before continuing, we consider a future application of the Nyström projection as an input layer to a neural network. If we are performing optimization with respect to a loss function $L : \mathbb{R}^n \to \mathbb{R}$ and we want to restrict our optimization to the manifold, we can simply compose with the projection $\tilde{\imath}$ to find,

$$L|_{\mathcal{M}}(x) = L(\hat{X} \circ \Phi)(x)$$

which has gradient,

$$\nabla L|_{\mathcal{M}}(x) = D\Phi(x)^\top \hat{x}^\top \nabla L((\hat{X} \circ \Phi)(x)) \in T_x\mathcal{M} \subset \mathbb{R}^n.$$

Here, we assume that the gradient of L is already computable, and we are merely evaluating grad L on $\tilde{\imath}(x) = \hat{X} \circ \Phi(x)$ which is still a point in data space and just happens to have been projected down onto the manifold. Moreover, we already have the matrix $\hat{x}$, so the only additional component that is needed is the gradient of Φ, which simply requires computing the gradient of each of the Nyström eigenfunctions.

2.2 SEC vectors

Our goal is to find vector fields that span the tangent spaces of the manifold at each point. While this is not always possible

with just d vector fields (where d is the intrinsic dimension of the manifold), the Whitney embedding theorem guarantees that it is always possible with $2d$ vector fields. The L^2 inner product,

$$G(v, w) \equiv \langle v, w \rangle_{L^2} \equiv \int_{\mathcal{M}} v \cdot w \, d\text{vol},$$

induced on vector fields by the Riemannian metric provides a natural notion of orthogonality that can help to identify non-redundant vector fields. Here, we use the notation $v \cdot w \equiv g(v, w)$ to denote the function which at each point is the Riemannian dot product of the two vectors at fields at that point.

In addition to being orthogonal, we also need these vector fields to cut through noise and follow the coarse (principal) geometric structure of the manifold. For this purpose, we introduce the Dirichlet energy on vector fields, induced by the weak form of the Hodge 1-Laplacian, $\delta_1 = d\delta + \delta d$, where d, δ are the exterior derivative and codifferential, respectively. These operators are defined on differential forms, which are dual to tensor fields, and, in particular, the dual of a vector field is a 1-form. The musical isomorphisms switch back and forth between forms and fields, with sharp, $\sharp$, turning forms into fields, and flat, $\flat$, going back. As an example, the codifferential on 1-forms is related to the divergence operator by,

$$\nabla \cdot v = -\delta(v^\flat)$$

Similarly, the exterior derivative, which acts on forms, induces an operator on smooth fields which generalizes the curl operator,

$$\nabla \times v \equiv (\star d(v^\flat))^\sharp.$$

This operator coincides with the curl when manifold is three-dimensional, so we use the same name, but in general, on an n-dimensional manifold, the output of the generalized curl operator will be a $n - 2$ tensor field. Using the generalized divergence and curl operators, we can now define the Dirichlet energy on smooth vector fields as

$$E(v, w) \equiv \int_{\mathcal{M}} (\nabla \times v) \cdot (\nabla \times w) \, d\text{vol} + \int_{\mathcal{M}} (\nabla \cdot v)(\nabla \cdot w) \, d\text{vol}$$

Where we note that the dot product in the first term is the extension of the Riemannian metric to $n - 2$ forms. By minimizing this energy, we will ensure that we have the smoothest possible vector fields, as seen by a global (integrated) measure of smoothness. As discussed in Appendix A, the Dirichlet energy on vector fields is motivated by a dual energy on differential 1-forms. Note that while measuring smoothness on functions only requires a single term, the integral of the gradient of the function, measuring smoothness of vector fields requires two different types of derivatives. This is because neither the divergence nor the curl can completely measure the different types of oscillations a vector field can have, but when combined they provide a robust measure of smoothness.

The Dirichlet energy and Riemannian metric together will define a natural function for identifying good sets of vector fields, which in turn will reduce to a generalized eigenvalue problem. In Appendix A, we overview the SEC construction of the Dirichlet energy and how to find its minimizers relative to the Riemannian inner product. For now, we demonstrate the advantage of this approach to finding vector fields that respect the global structure of the data set. In Figure 3, we show a data set that while near a simple manifold exhibits variations in density and noise levels characteristic of real data. This example clearly demonstrates how the SEC principal vectors respect the principal global structure of the manifold, rather than getting lost in local details the way local linearization is. This is possible because the Dirichlet energy captures a measure of smoothness that is balanced over the entire global structure of the data set.

Finally, we should note an important caveat and related direction for future research. The vector fields identified here are only globally orthonormal, meaning in an integrated sense. This means that they are not required to be orthonormal in each coordinate chart, since positive alignment in some regions can be canceled by negative alignment in others. In future study, one could consider a local orthonormality condition. This is related to the search for minimal embeddings.

2.3 On-manifold projected gradient descent

As discussed in Section 1, projected gradient descent (PGD) relies on computing the network loss gradient with respect to the input rather than the weights of the model. Once these input gradients have been computed via backpropagation with respect to a target class, the input is perturbed in the direction of the gradient to create an adversarial example. If the resulting perturbed image is farther than ϵ away from the starting image for a given distance metric (typically ℓ_2 or ℓ_∞), then the perturbed image is projected onto the ϵ-ball surface around the starting image. This is done with the intent of keeping the adversarial example from becoming unrecognizable as the original class. We use the SEC and Nyström projection to create an on-manifold PGD that is representative of the original class but not constrained to an ϵ-ball. First, we take the gradient of the network for an input with respect to the input's class. Then, we find the input point's position on the manifold with the Nyström projection, see Section 2.1. Using the SEC, we then compute the vector fields that are tangent to the manifold and obtain the tangent bundle at the position of the starting point's position on the manifold, see Figure 3 for a comparison of vector fields from SEC to local PCA. Next, we project the gradient onto the orthonormalized subspace spanned by a subset of tangent vectors at the input point depending on the dimensionality of the underlying data. For instance, if the underlying manifold had an estimated dimension of 2 then you would choose the 2 tangent vectors from the smoothest vector fields computed by the SEC. We orthonormalize the subspace by using the non-zero eigenvectors of the singular value decomposition (SVD) to obtain an unbiased basis. Then, we step along the direction of the gradient in the tangent space by an amount determined by a hyperparameter so that the input is sufficiently perturbed. As the final step, we use the Nyström projection to return this perturbed sample back to the manifold, see Figure 2 for an example. The Nyström projection ensures that the resulting example is on that particular class's manifold and provides the information for determining

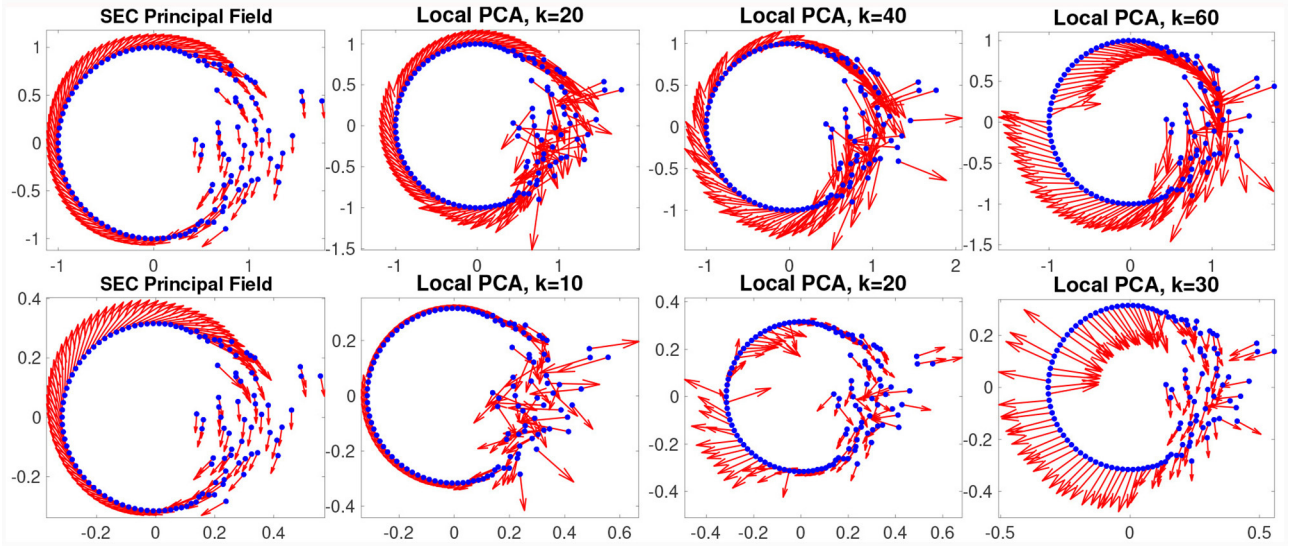


FIGURE 3
SEC vector fields **(left)** respect global structure. Here, we consider a data set (blue points) localized near the unit circle but with varying density and varying amounts of noise. In the top left plot, we show the principal vector field identified by the SEC (the global minimizer of the Dirichlet energy). We compare this to a local PCA approach using k = 20, 40, and 60 nearest neighbors of each point **(middle left to right)**. Note how the SEC vector field smoothly respects the dominant mode of variation in the data set; while the local PCA approach can find the tangent direction in the clean sections, it loses track in the noisy section of the data. The problem is further exacerbated in higher dimensions, and in the **bottom row**, we repeat the same experiment using an isometric embedding of the circle into four dimensions. Again, we show the SEC principal vector **(bottom left)** and local PCA with k = 10, 20, and 30 nearest neighbors from middle left to right.

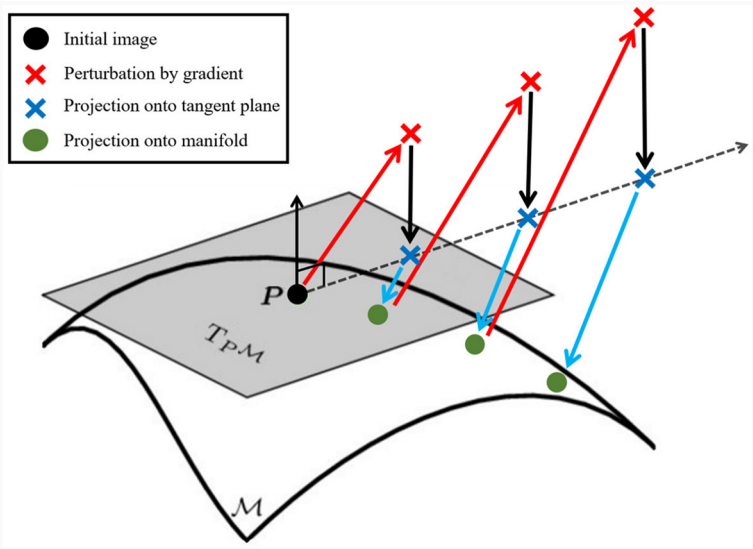


FIGURE 4
On-manifold PGD steps for a 2D tangent space. Starting with an initial image (black dot), we follow the network gradient (red arrow). The perturbed sample (red X) is then projected (black arrow) onto the tangent plane, $T_P\mathcal{M}$ (gray surface), from the geometry found around the initial image. The upwards black arrow pointing out of the tangent plane at the black dot is normal to the tangent plane for visual clarity. The on-plane perturbation (blue X) is then Nyström projected (blue arrow) onto the manifold (curved surface $\mathcal{M}$). The on-manifold point (green dot) is then classified to determine if the network has classified it correctly. If the on-manifold point is classified correctly, then it is fed back into the network and the process is repeated, although it reuses the same manifold and $T_P\mathcal{M}$ from the starting point.

the example's semantic labels (intrinsic coordinates) and tangent vectors. We can obtain the on-manifold example's semantic labels using the same methodology used to obtain the mapping from CIDM coordinates to pixel, see Figure 1 for an example of obtaining semantic labels on a learned manifold. The perturbation and projection steps are repeated until a misclassification is found, see Figure 4 for an pictoral overview and Algorithm 1 for a pseudo-code description. We use the same tangent basis from the starting

image at all steps because we found that updating the tangent basis at each step did not appreciably change the result after performing Nyström projection. If the on-manifold PGD is successful, then it produces an adversarial example that fools the classifier and is also on the input class's manifold.

Input: x_0, y_0, α, $T_{x_0}\mathcal{M}$ //initial image, class, step size, and tangent bundle
$x \leftarrow x_0$
$y \leftarrow \mathcal{C}(x)$ //network classification of image
$\mathcal{P}_{x_0} \leftarrow \sum_i v_i v_i^T;\ v_i \in T_{x_0}\mathcal{M}$ //projector from tangent bundle at x_0
while $y = y_0$ **do**
  $g \leftarrow \nabla_x \mathcal{L}(x, y)$ //classifier loss gradient w.r.t. x
  $x \leftarrow x + \alpha \mathcal{P}_{x_0}(g)$ //step along projected gradient
  $x \leftarrow \mathcal{N}(x)$ //Nyström project image onto manifold
  $y \leftarrow \mathcal{C}(x)$
end while
return x //on-manifold adversarial example

Algorithm 1. On-manifold PGD algorithm.

3 Results

The main result we present shows on-manifold adversarial examples that are explainable in human understandable terms by using the adversaries' semantic labels on the manifold. We present experimental results for finding on-manifold adversaries using a VGG11 classifier (Simonyan and Zisserman, 2014) and a synthetic data set. Our classifier is trained and validated to classify RGB images of various vehicles (see Figure 5 for examples), which were generated using synthetic data collected in Microsoft's AirSim platform, allowing for insertion of various vehicle classes in a range of locations. Each vehicle class is sampled over two sets of intrinsic parameters, represented by the azimuth angle and down look angle (DLA) from which the image is captured. The azimuth angles are sampled one degree apart from 1 to 360 and the down look angle is sampled one degree apart from 1 to 45 degrees. The dense sampling in intrinsinc parameters will enable CIDM and SEC computations, while also allowing for NN models trained on the data to be fairly robust.

3.1 VGG11 with static backgrounds

We trained a VGG11 classifier on a subset of the down look angles ($10^\circ - 30^\circ$) and all azimuth angles. See Figure 6 for the loss and decision boundaries of the classifier for a single class over all view angles available, including those the classifier was not trained on. We use an image from the training set that was in a region close to a decision boundary at 30° down look and 100° azimuth. We then apply our on-manifold PGD to that point using a manifold modeled on the points surrounding that point from 0° to 40° DLA and 80° to 120° azimuth, see Figure 7. The output of the on-manifold PGD for this example results in a misclassification that correlates with the decision boundary for this class near that sample in view angle as seen in Figure 6. Note that the results of projecting onto the manifold provide not only the on-manifold point in pixel space but also in intrinsic parameter coordinates. This means that the misclassification can be explained in terms of human-understandable features. In this case, the on-manifold adversarial example seen at step 5 in Figure 7 is shown to be at 39.36 DLA and 101.22 azimuth, which can be confirmed to be a region of misclassification by looking at the explicit sampling of that region in the decision boundary map of Figure 6.

When perturbing the on-manifold images with the gradient, $X' = X + \alpha \nabla X$, we use a fixed step size, $\alpha = 10^{6.38}$ for convenience. We tested a range of step sizes ranging on a log scale to find the smallest step that sufficiently perturbed the input into misclassifying with the on-manifold PGD algorithm. The size of the parameter α is due to the fact that the network gradient is mostly contained in a space not spanned by the tangent vectors. After projecting the gradient onto the tangent plane, the magnitude of the projected gradient is several orders of magnitude smaller than the raw gradient, which is visibly discernible in the second and third columns of Figures 7, 10. To obtain a sufficient perturbation for the classifier to misclass, this required a large value for α. Due to the fact that the ℓ_2 norm of the gradient was typically on the order of 10^{-3} for these examples, the gradient could also be normalized so that smaller values of α could be used.

We choose the first two tangent vectors from the SEC because the output of the SEC code returns the vector fields ordered by smoothness, which typically results in the first vector fields being best aligned with the manifold. We visually confirmed this in CIDM coordinate space as seen in Figure 8.

3.2 Simple CNN with random backgrounds

We present another on-manifold PGD result using an adversarially trained network. The model is a CNN with two convolutional layers and a final fully connected layer. Both of the 2D convolutional layers have a kernel size of 3, stride of 1, padding 1, and are followed by a 2D max pooling with a kernel size of 2 and then ReLu activations. The first layer has 12 filters and the second layer has 24 filters. For our data set with images sizes of 128×128, this leads to an input of size $32 \times 32 \times 24$ into the last fully connected layer. The output of the final layer is set to the number of classes in the data set and we apply the softmax activation function. The training paradigm for this experiment consists of using images with randomly generated backgrounds, as shown in Figure 5, with the result that the trained network will be background-agnostic. We train on images from all azimuth angles and DLA from 20 to 30 degrees. After six epochs of standard training, we switch to adversarial training, where adversarial images are generated by following the network gradient. We continue until we reach 100 total epochs of training. Figure 9 then plots the loss and prediction maps over azimuth and DLA for the luxury SUV class.

We note in Figure 9 that the majority of the misclassifications occur outside of the training regime, with a few rare misclassifications in the training regime, given as scattered predictions of the box truck and sedan classes. We seek to generate on-manifold adversarial examples in the range $20^\circ - 30^\circ$ for DLA

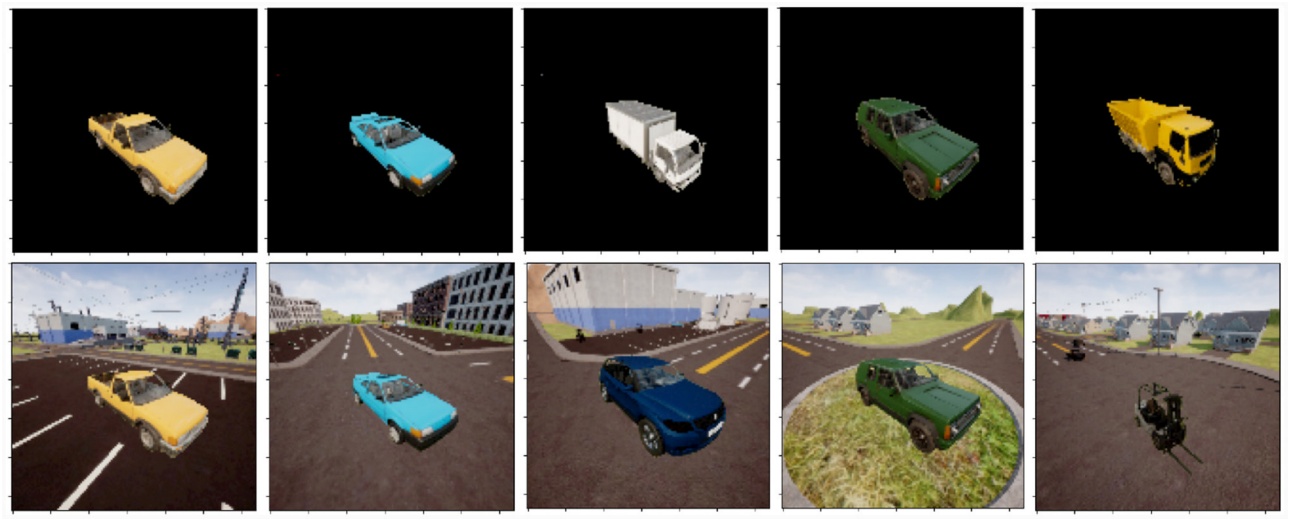


FIGURE 5
Examples of RGB vehicle datatset, consisting of seven vehicle types (pickup truck, SUV, sedan, dump truck, box truck, jeep, and fork lift), varying in 360 degrees azimuth and 45 degrees down look angle. The data contains six types of scene backgrounds including urban and rural environments. The full resolution data is 256 by 256 pixels, although it has been downsampled to 128 by 128. **Top row:** vehicle images with flat backgrounds generated using image segmentation maps. **Bottom row:** vehicle images with randomly sampled location backgrounds (images are again inserted into backgrounds using segmentation maps).

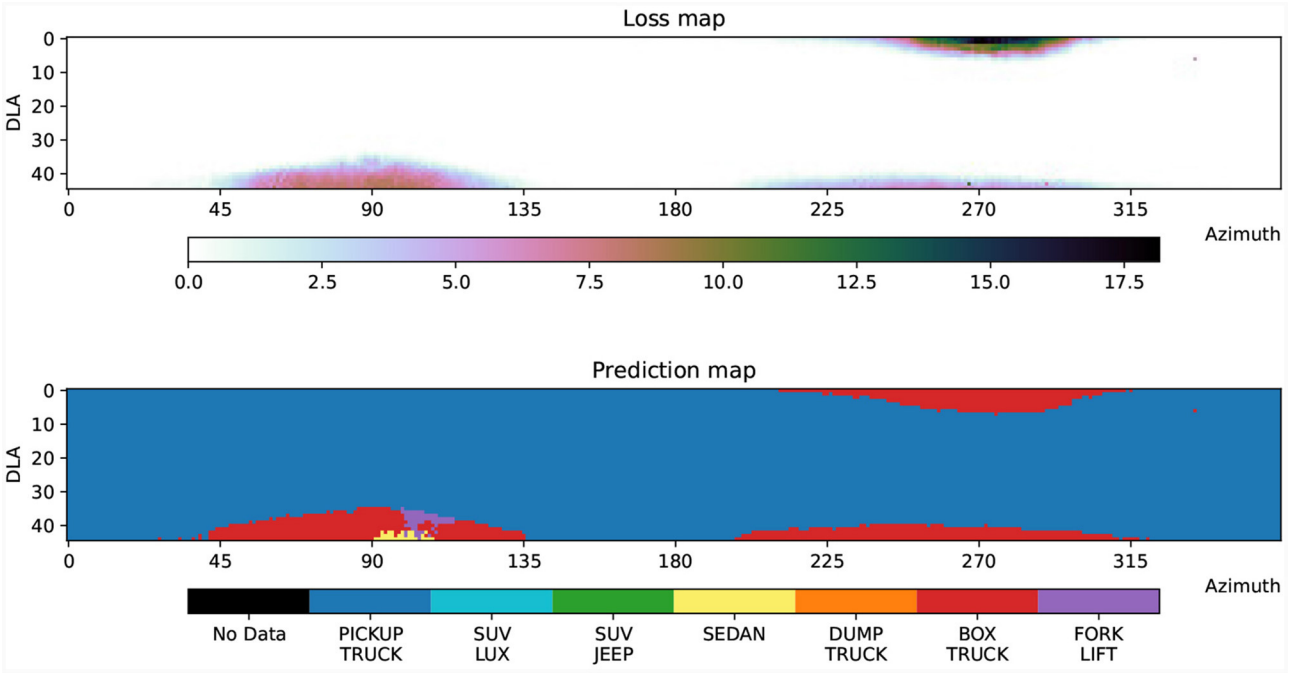


FIGURE 6
Network loss and prediction map (pickup truck class). Loss map and decision boundary map for the Pickup Truck class on a VGG11 network. The classifier was trained with the seven vehicle types shown in Figure 5 using down look angles from 10° to 30° and azimuths from 0° to 359°.

and $135^\circ - 165^\circ$ for azimuth, which is the range we use in geometry calculations. Notice, in this experiment, we are only providing our geometry tools with data from the intrinsic parameter range that the network was trained on. The goal of this experiment is to use our geometry tools to find and correctly explain an adversarial example that caused the network to mislcassify, without giving the geometry tools preferential treatment in the form of additional data that the network was not trained on. Figure 10 illustrates our on-manifold PGD iteration, as described in Section 2.3, with the starting point at DLA 20° and azimuth 150°.

Figure 10 depicts the iteration of on-manifold PGD to an on-manifold adversarial example, which is classified as a box truck

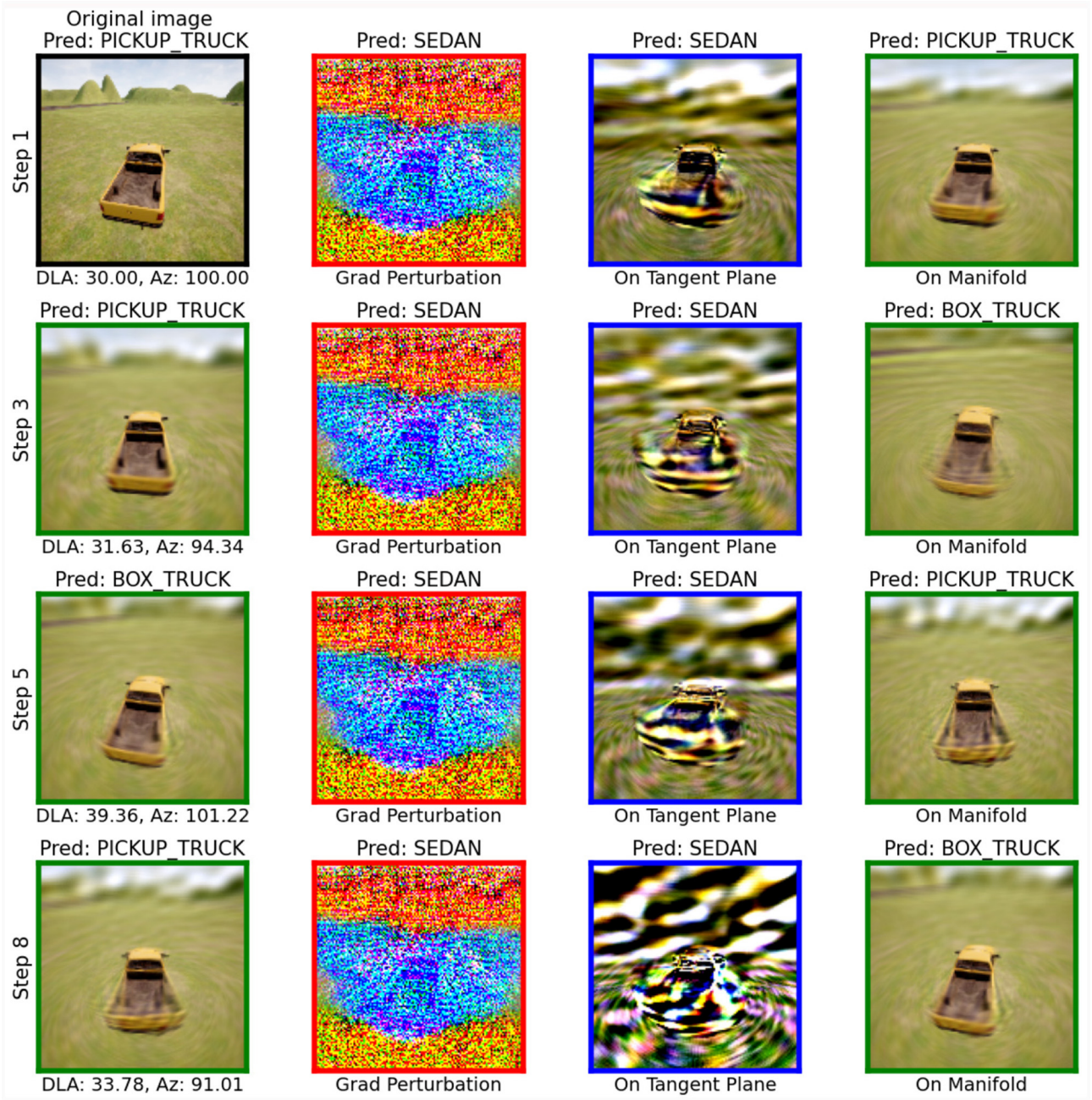


FIGURE 7
Generation of an on-manifold adversarial image. The first column is the Nyström projection of the input from the previous step with the step number on the vertical axis and the predicted intrinsic parameters on the horizontal axis. The second column is the first column after adding the network gradient times $10^{6.38}$. The third column adds the image from the first column and 10^{7} times the gradient after it is projected onto the manifold subspace spanned by the tangent subspace. The fourth column is the result of Nyström projecting the third column so that it is on the manifold. Each image includes the network prediction of that image at the top, and the outline color corresponds to the steps in Figure 4.

(as one would expect from Figure 9). In this experiment, we note that adversarial examples causing a misclassification often contain a reflection on the side of the vehicle. We have verified that this matches with a rare feature in the training data, where images containing a reflection commonly cause the luxury SUV to be misclassified as the white box truck. These reflections are a result of AirSim's environment and they represent an unexpected challenge that network mislcassified but was identified by our on-manifold PGD approach. Evidently, not only is this approach able to generate adversarial example which can be explained in terms of their semantic labels but it also provides explainable insights into which features of an image cause a network to misclassify.

4 Discussion

In conclusion, we have presented a formal introduction of CIDM, a type of variable bandwidth kernel diffusion map that

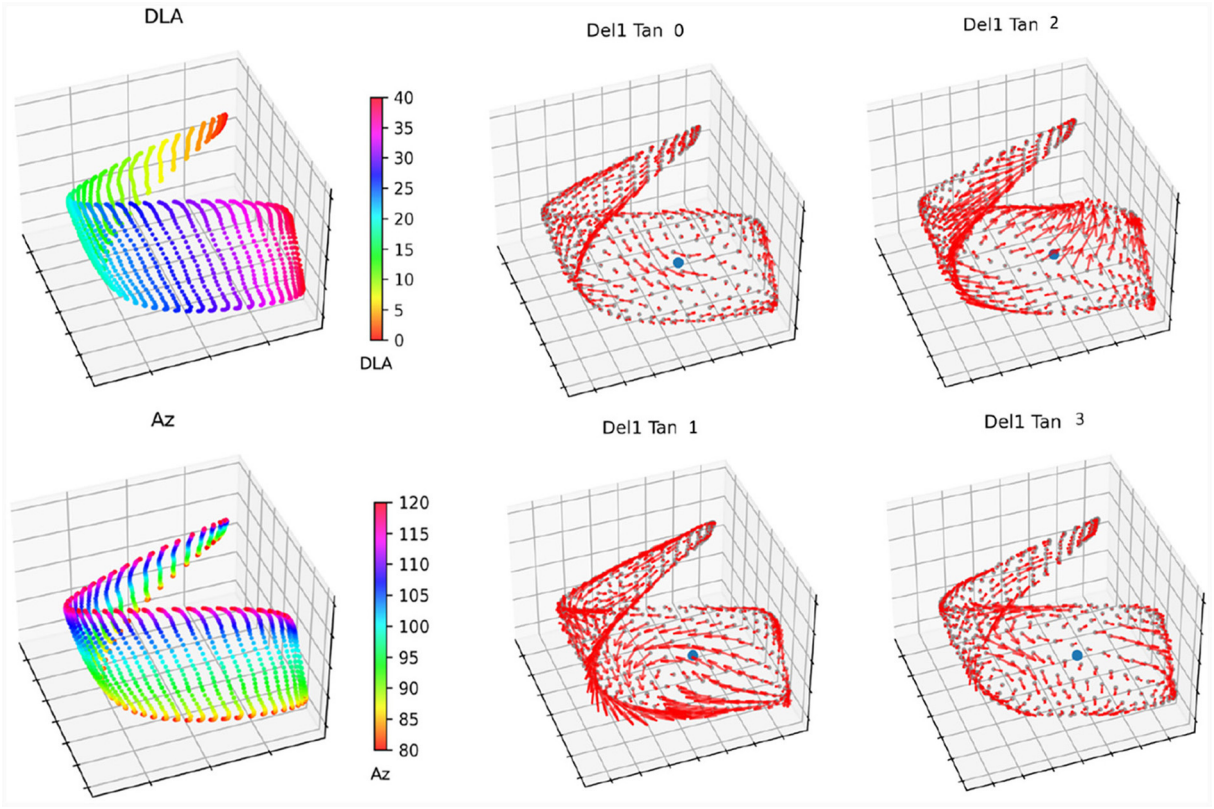


FIGURE 8
Pickup truck manifold approximation. The top two images show the manifold of the Pickup Truck plotted using the first three coordinates from CIDM. The top left image is colored by the azimuth angle, and the top right is colored by the down look angle. The bottom four plots show the tangent vector fields of the SEC as red arrows, and the initial point around which the geometry approximation was built as a blue dot.

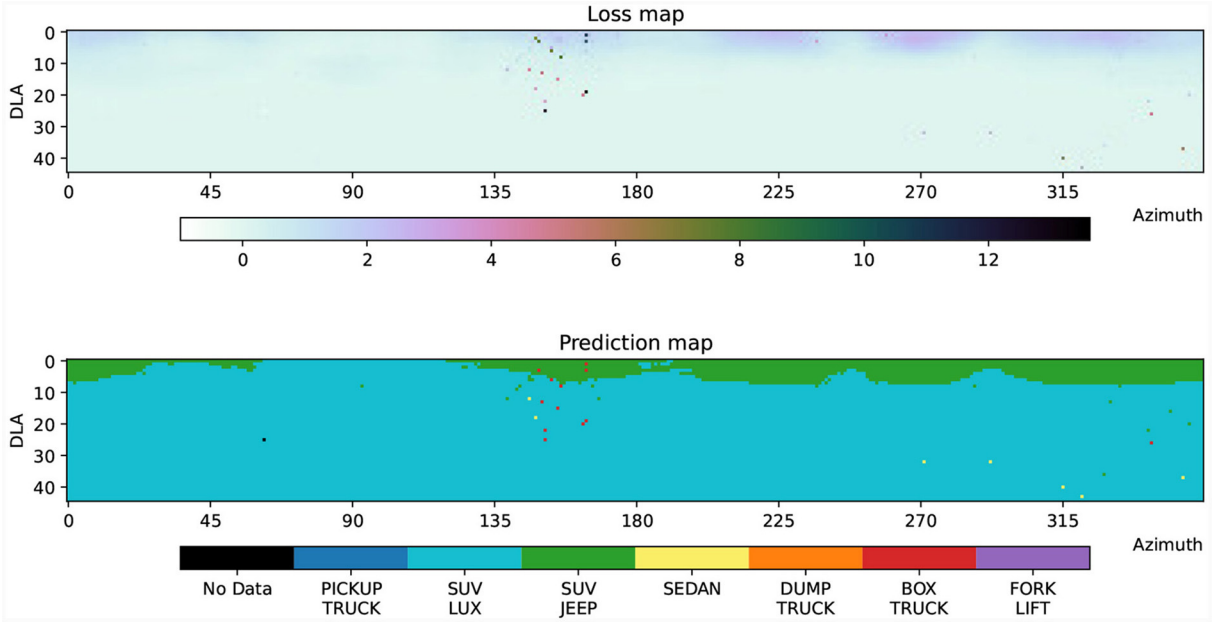


FIGURE 9
Network loss and prediction map (luxury SUV class). Loss map and decision boundary map for the Luxury SUV class on a CNN. The classifier was trained with seven vehicle types as shown in Figure 5 using down look angles from 20° to 30° and azimuths from 0° to 359°. Adversarial training begun after epoch 6 and continued until epoch 100, where the network gradients were used to generate adversaries. Network evaluation takes place with scene background images, while the network was trained with a randomized set of backgrounds to avoid an over-dependence of the network on the image background.

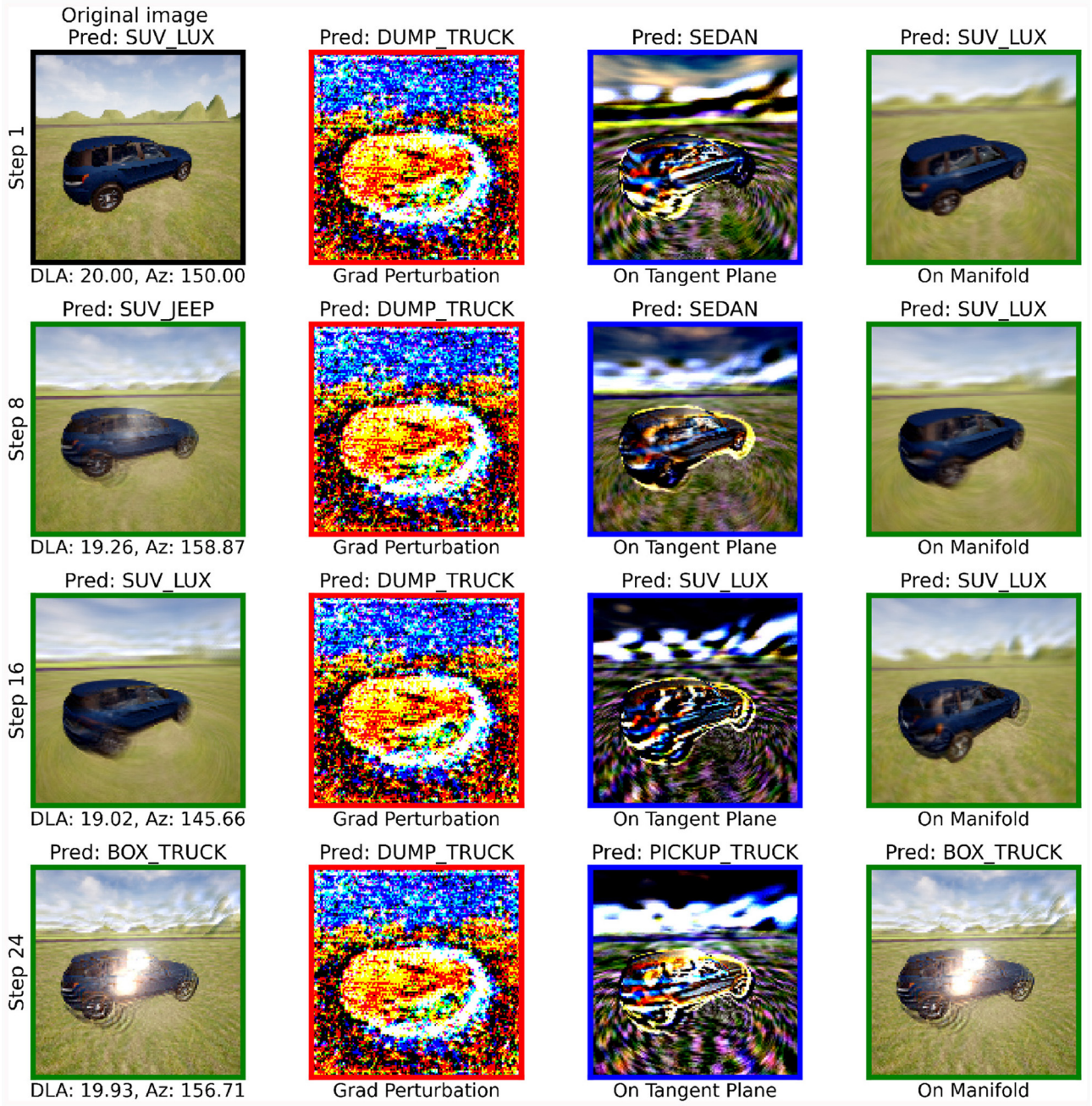


FIGURE 10
Generation of an on-manifold adversarial image. The first column is the Nystöm projection of the input from the previous step with the step number on the vertical axis and the predicted intrinsic parameters on the horizontal axis. The second column is the first column after adding the network gradient times $10^{5.23}$. The third column shows the gradient after it is projected onto the manifold subspace spanned by the tangent subspace. The fourth column is the result of Nystöm projecting the third column so that it is on the manifold. Each image includes the network prediction of that image at the top and the outline color corresponds to the steps in Figure 4.

is adept at dealing with heterogenous data density. We have also introduced a novel application of the Nyström method for extending the CIDM eigenfunctions to new data points. We use the Nyström projection to map off-manifold points onto the manifold inside PGD to implement an on-manifold PGD. Additionally, we showed how to use SEC to find vector fields of the manifold for points on the manifold, which we use as a local linear space around the data to project to for intermediate points in our on-manifold PGD implementation. We were able to successfully obtain on-manifold examples that the trained NN misclassifies, showing the promise of on-manifold examples that can be found in input space without reducing down to a latent space. Our reported results provided the geometry approximation tool with data that was outside the data used to train the NN classifier, meaning that

the output of the on-manifold PGD algorithm would not be a valid input for adversarial regularization. However, the experiment did provide novel tools for modeling the data manifold in a manner that allowed the on-manifold PGD algorithm to walk in the direction of the NN gradient while remaining on the manifold. This provided novel examples that were on-manifold but not simply part of the hold out data. In addition, the Nyström projection onto the manifold provided the intrinsic parameters of the adversarial examples so that the misclassification was human interpretable. The ability to report the intrinsic parameter of arbitrary points on the manifold opens the door to being able to explain NN decision boundaries in human understandable terms without explicitly sampling all possible inputs. In non-synthetic data, a single class will typically not have continuously varying intrinsic parameters, so additional work needs to done to transition these tools to real-world data sets.

Data availability statement

The datasets presented in this article are not readily available because the way the dataset was generated was described for reproducibility, but it is not available due to commercial restrictions. Requests to access the datasets should be directed to aaron.mahler@teledyne.com.

Author contributions

AM: Conceptualization, Investigation, Methodology, Software, Supervision, Visualization, Writing – original draft. TB: Conceptualization, Funding acquisition, Investigation, Methodology, Visualization, Writing – original draft. TS: Conceptualization, Funding acquisition, Investigation, Methodology, Software, Writing – review & editing. HA: Investigation, Project administration, Writing – review & editing. MM: Investigation, Software, Visualization, Writing – original draft. JS: Investigation, Software, Writing – review & editing. IK: Conceptualization, Methodology, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This material was based upon study supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112290079, and it has been approved for public release; distribution is unlimited.

Acknowledgments

The authors would also like to thank Juan M. Bello-Rivas for many helpful and insightful discussions related to these topics.

Conflict of interest

AM, TS, and MM were employed by Teledyne Scientific & Imaging, LLC.

The remaining 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.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2024.1274181/full#supplementary-material

References

Athalye, A., Carlini, N., and Wagner, D. (2018). "Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning* (Cambridge, MA: PMLR), 274–283.

Belkin, M., and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* 15, 1373–1396. doi: 10.1162/089976603321780317

Berry, T., and Giannakis, D. (2020). Spectral exterior calculus. *Commun. Pure Appl. Math.* 73, 689–770. doi: 10.1002/cpa.21885

Berry, T., and Harlim, J. (2016). Variable bandwidth diffusion kernels. *Appl. Comput. Harmon. Anal.* 40, 68–96. doi: 10.1016/j.acha.2015.01.001

Berry, T., and Sauer, T. (2016). Local kernels and the geometric structure of data. *Appl. Comput. Harmon. Anal.* 40, 439–469. doi: 10.1016/j.acha.2015.03.002

Berry, T., and Sauer, T. (2019). Consistent manifold representation for topological data analysis. *Foundations of Data Science* 1, 1. doi: 10.3934/fods.2019001

Carlini, N., and Wagner, D. (2017). Towards evaluating the robustness of neural networks. *arXiv* [preprint]. doi: 10.1109/SP.2017.49

Cho, S., Jun, T. J., Oh, B., and Kim, D. (2020). "DAPAS: denoising autoencoder to prevent adversarial attack in semantic segmentation," in *2020 International Joint Conference on Neural Networks (IJCNN), pages 1-8. Conference Name: 2020 International Joint Conference on Neural Networks (IJCNN)* (Glasgow: IEEE).

Coifman, R. R., and Lafon, S. (2006a). Diffusion maps. *Appl. Comput. Harmon. Anal.* 21, 5–30. doi: 10.1016/j.acha.2006.04.006

Coifman, R. R., and Lafon, S. (2006b). Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions. *Appl. Comput. Harmon. Anal.* 21, 31–52. doi: 10.1016/j.acha.2005.07.005

Dietrich, F., Bello-Rivas, J. M., and Kevrekidis, I. G. (2021). On the correspondence between gaussian processes and geometric harmonics. *arXiv* [preprint]. doi: 10.48550/arXiv.2110.02296

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. (2019). *Robustness*. Python Library. Available online at: https://github.com/MadryLab/robustness

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *arXiv* [preprint]. doi: 10.48550/arXiv.1412.6572

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. *arXiv* [preprint]. doi: 10.48550/arXiv.1905.02175

Kurakin, A., Bengio, A., and Goodfellow, A. K. (2018). "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security* (Boca Raton, FL: Chapman and Hall/CRC), 14.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2019). Towards deep learning models resistant to adversarial attacks. *arXiv*. [preprint]. doi: 10.48550/arXiv.1706.06083

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). "Universal adversarial perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 1765–1773. Available online at: https://ieeexplore.ieee.org/document/8099500

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). "DeepFool: a simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 2574–2582.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17* (New York, NY: Association for Computing Machinery), 506–519.

Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv* [preprint]. doi: 10.48550/arXiv.1409.1556

Stutz, D., Hein, M., and Schiele, B. (2019). Disentangling adversarial robustness and generalization. *arXiv* [preprint]. doi: 10.1109/CVPR.2019.00714

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. (2018). Is robustness the cost of accuracy?-A comprehensive study on the robustness of 18 deep image classification models. *arXiv* [preprint]. 631–648. doi: 10.1007/978-3-030-01258-8_39

Su, J., Vargas, D. V., and Kouichi, S. (2019). One pixel attack for fooling deep neural networks. *IEEE Transact. Evol. Comp*. 23, 828–841. doi: 10.1109/TEVC.2019.2890858

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., et al. (2014). Intriguing properties of neural networks. Technical Report. *arXiv*. doi: 10.48550/arXiv.1312.6199

Tabacof, P., and Valle, E. (2016). "Exploring the space of adversarial images," in *2016 International Joint Conference on Neural Networks (IJCNN)* (Vancouver, BC: IEEE), 426–433.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2020). Ensemble adversarial training: attacks and defenses. *arXiv* [preprint]. doi: 10.48550/arXiv.1705.07204

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. *arXiv* [preprint]. doi: 10.48550/arXiv.1805.12152

Xu, W., Evans, D., and Qi, Y. (2018). "Feature squeezing: detecting adversarial examples in deep neural networks," in *Proceedings 2018 Network and Distributed System Security Symposium* (San Diego, CA: Internet Society).

Yin, Z., Wang, H., and Wang, J. (2020). "War: an efficient pre-processing method for defending adversarial attacks," in *Machine Learning for Cyber Security: Third International Conference, ML4CS 2020, Guangzhou, China, October 8–10, 2020, Proceedings, Part II* (Berlin: Springer-Verlag), 514–524.

frontiers | Frontiers in Artificial Intelligence

TYPE Original Research
PUBLISHED 09 May 2024
DOI 10.3389/frai.2024.1255566

OPEN ACCESS

EDITED BY
Yunye Gong,
SRI International, United States

REVIEWED BY
Pavan Turaga,
Arizona State University, United States
Ankita Shukla,
University of Nevada, Reno, United States

*CORRESPONDENCE
Zhenzhen Liu
✉ zl535@cornell.edu
Jin Peng Zhou
✉ jz563@cornell.edu

†These authors have contributed equally to this work and share first authorship

RECEIVED 09 July 2023
ACCEPTED 25 March 2024
PUBLISHED 09 May 2024

CITATION
Liu Z, Zhou JP and Weinberger KQ (2024) Leveraging diffusion models for unsupervised out-of-distribution detection on image manifold. *Front. Artif. Intell.* 7:1255566. doi: 10.3389/frai.2024.1255566

Leveraging diffusion models for unsupervised out-of-distribution detection on image manifold

Zhenzhen Liu*†, Jin Peng Zhou*† and Kilian Q. Weinberger

Department of Computer Science, Cornell University, Ithaca, NY, United States

Out-of-distribution (OOD) detection is crucial for enhancing the reliability of machine learning models when confronted with data that differ from their training distribution. In the image domain, we hypothesize that images inhabit manifolds defined by latent properties such as color, position, and shape. Leveraging this intuition, we propose a novel approach to OOD detection using a diffusion model to discern images that deviate from the in-domain distribution. Our method involves training a diffusion model using in-domain images. At inference time, we lift an image from its original manifold using a masking process, and then apply a diffusion model to map it towards the in-domain manifold. We measure the distance between the original and mapped images, and identify those with a large distance as OOD. Our experiments encompass comprehensive evaluation across various datasets characterized by differences in color, semantics, and resolution. Our method demonstrates strong and consistent performance in detecting OOD images across the tested datasets, highlighting its effectiveness in handling images with diverse characteristics. Additionally, ablation studies confirm the significant contribution of each component in our framework to the overall performance.

KEYWORDS

out-of-distribution detection, diffusion models, score-based models, generative modeling, manifold learning

1 Introduction

The goal of out-of-distribution (OOD) detection is to ascertain if a given data point comes from a specific domain. This task is crucial given that machine learning models generally require that the distribution of test data mirrors the distribution of the training data. In cases where the test data deviates from the training distribution, the models can generate meaningless or deceptive results. This could be especially harmful for tasks in high-stake areas like healthcare (Hamet and Tremblay, 2017) and criminal justice (Rigano, 2019).

The OOD detection task has been examined under settings with access to varied amount of information. These settings can be categorized as supervised and unsupervised. Among supervised settings, the most informed scenario makes the assumption that exemplar out-of-domain data are available. One can then incorporate them in the training of neural networks to enhance their ability to recognize out-of-domain inputs (Hendrycks et al., 2018; Ruff et al., 2019). Various methods excel on identifying out-of-domain data when that resemble the training examples, but their performance deteriorates on out-of-domain

inputs that are not represented in the training process. In practical applications, inputs are often highly diverse, and it is challenging to construct a truly representative set of out-of-domain examples. A more feasible setting is to only leverage in-domain classifiers or class labels (Hendrycks and Gimpel, 2016; Liang et al., 2017; Lee et al., 2018; Huang et al., 2021; Wang et al., 2022). Although this setting is less restrictive, it still requires two essential conditions: well-defined categorization of the in-domain data and an adequate amount of labeled data. These conditions do not hold for many tasks. In contrast, the fully unsupervised setting only require access to unlabeled in-domain data, which can often be obtained with low cost and in abundant quantities. As a result, it is ideal to develop OOD detectors under the fully unsupervised setting.

Recently, the diffusion models (DMs), a type of generative models, have received increasing attention in the machine learning community (Ho et al., 2020; Song et al., 2020). DMs operate on two procedures: The forward operation performs iterative noise addition to an image's pixels and transforms it into a sample drawn from a noise distribution. The backward operation—performed by a dedicated neural network—gradually removes noise from the image, guiding a noise image toward a specific image manifold.

In this paper, we show that we can leverage DMs as a mapping to a manifold, and use it for unsupervised OOD detection. Conceptually, if an image is lifted from its manifold, a diffusion model trained over the same manifold can guide it back to its original manifold. However, if the diffusion model has been trained on a different manifold, it would lead the lifted image toward its own training manifold, resulting in a substantial distance between the original and the mapped images. Therefore, we can identify out-of-domain images based on this distance.

To this end, we introduce an innovative unsupervised method for out-of-distribution detection, **L**ift, **M**ap, **D**etect (LMD), that embodies the aforementioned concept. **Lifting** is performed through image corruption. For instance, a face image that has been masked in the center will no longer fit into the face image category. Previous research by Song et al. (2020) and Lugmayr et al. (2022) have demonstrated that the diffusion model can perform inpainting, i.e., restoring missing areas in an image with visually convincing content, without the need for additional training. This allows us to **map** the mapped image via inpainting with a in-domain diffusion model. We can employ a conventional image similarity metric to calculate the distance between the original and mapped images, and **detect** an out-of-domain image when there is a significant distance. In Figure 1, we provide an example: A diffusion model trained with face images maps a lifted in-domain face image closer to its original location, while moving an lifted fire hydrant, an out-of-domain image, further away.

Our main contributions include: (1) We propose an innovative unsupervised OOD detection technique, *Lift, Map, Detect* (LMD), that utilizes of the inherent manifold mapping capacity of diffusion models, and incorporates design choices that enhance the distinguishability between in-domain and out-of-domain data. (2) We conduct extensive experiments on various image datasets with different characteristics to illustrate the versatility of LMD. (3) We present in-depth analysis, visualizations and ablations to confirm LMD's underlying hypothesis and provide insights into LMD's behaviors.

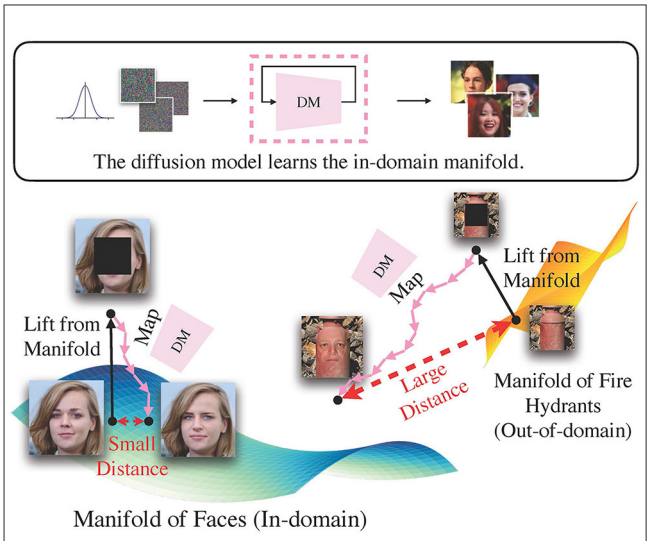


FIGURE 1
The intuition behind LMD. In essence, LMD leverages a diffusion model as a mapping toward the in-domain manifold. It applies a mask to the image to lift it from its original manifold, and uses the diffusion model to guide it toward the in-domain manifold. If an image is in-domain, it would generally have smaller distance between the original and mapped locations than out-of-domain images.

2 Materials and methods

2.1 Preliminaries

Problem formulation. Formally, we define the unsupervised out-of-distribution (OOD) task as follows: We aim to build a detector to identify data points $\mathbf{x}$ that deviate from a distribution of interest $\mathcal{D}$. The detector should be built using only unlabeled data $\mathbf{x_1}, \cdots, \mathbf{x_n}$ sampled from $\mathcal{D}$. It should assign an OOD score $s(\mathbf{x})$ that positively correlates with the likelihood of $\mathbf{x}$ *not* belonging to $\mathcal{D}$.

Diffusion models. In this section, we present a brief summary of the concepts behind the diffusion model (DM). It is a class of generative models that can learn complex distributions. It involves a forward process of diffusion and a backward process of denoising. Diffusion corrupts the original data with noise, while denoising—performed by a learned neural network—progressively reduces noise from the corrupted image. There are various formulations of diffusion models, such as score-based generative models (Song and Ermon, 2019) and stochastic differential equations (Song et al., 2020). A comprehensive review can be found in Yang et al. (2022).

LMD is agnostic to the different DM variants. Here, we describe one prominent variant: the Denoising Diffusion Probabilistic Models (DDPMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020). DDPM's diffusion process begins with a data sample x_0, and injects Gaussian noise at every subsequent step $t = 1, 2, \cdots, T$ following Equation (1)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_t, \beta_t\mathbf{I}) \tag{1}$$

where β_t adheres to a predetermined variance schedule. The denoising process has a prior distribution $x_T \sim \mathcal{N}(0, 1)$, and

Input: test image x, in-domain diffusion model θ_{in}
Output: OOD score of test image x
for $i = 1$ **to** r **do**
  $M_i \leftarrow$ Get_Mask(i)
  $x'_i \leftarrow$ Inpaint(x, M_i, θ_{in})
  $d_i \leftarrow$ Distance(x, x'_i)
end for
return Aggregate($d_1, \ldots, d_r$)

Algorithm 1. Lift, Map, Detect (LMD).

formulates the process following Equation (2)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2}$$

where both $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are parametrized by a neural network θ.

2.2 Lift, Map, Detect

Lift, Map, Detect (LMD) is inspired by the observation that a diffusion model maps images toward the manifold it is trained on. Concretely, it leverages a diffusion model trained over unlabeled in-domain data. Given a test image, LMD applies corruption techniques to lift it from its original manifold, and utilizes the diffusion model to map it toward the in-domain manifold on which the model is trained. As depicted in Figure 1, if the image is indeed in-domain, the model can map it back to its manifold close to its original location. Conversely, if the image belongs to a different manifold, then the diffusion model would redirect it toward the in-domain manifold, moving it further away from its original location. Hence, out-of-domain images often have larger distance between the original and mapped images than in-domain images, and LMD identifies images with large distance as OOD. Figure 2 presents the general framework of LMD in Figure 2, and Algorithm 1 provides a succinct representation of the LMD algorithm. Subsequent sections explain each component of LMD in detail.

2.2.1 Lifting and mapping images

LMD lifts an image by masking parts of it, and maps it by inpainting over the masked area. For convenience, we also refer the lifted and mapped images as masked and reconstructed images, respectively. Masking provides a straightward way of controlling the extent to which an image is lifted, as larger masked area generally corresponds to larger deviation from the manifold. Furthermore, recent studies have shown that vanilla diffusion models can perform inpainting without the need for retraining, regardless of the size or shape of the masked regions. This highlights masking and inpainting as an intuitive strategy. Algorithm 2 describes the high-level process of inpainting with diffusion models. Additionally, we observe that an alternative way of lifting and mapping an image is to just add noise to it and then denoise with the diffusion model. We compare this instantiation with masking and inpainting in Table 4.

LMD operates based on the assumption that in-domain images have smaller reconstruction distance than out-of-domain images.

Input: original image x_{orig}, binary mask M where 0 indicates region to be inpainted, diffusion model θ
Output: inpainted image x_{inp}
for $t = T$ **to** 1 **do**
  if $t == T$ **then**
    $x_{inp} \leftarrow$ sample from noise distribution
  end if
  $x'_{orig} \leftarrow$ diffuse($x_{orig}; \theta$) to step $t-1$
  $x_{inp} \leftarrow$ denoise($x_{inp}; \theta$) to step $t-1$
  $x_{inp} \leftarrow x'_{orig} \cdot M + x_{inp} \cdot (1 - M)$
end for
return x_{inp}

Algorithm 2. Inpaint.

In practice, the validity of this assumption depends on two factors. First of all, inpainting with a diffusion model is stochastic. This occasionally leads to unfaithful in-domain reconstructions or faithful out-of-domain reconstructions. Consequently, a single reconstruction distance provides a noisy signal for identifying OOD images. To mitigate the randomness, we perform **multiple reconstructions** for each image, and use the median reconstruction distance as the OOD score. Our experiments in Section 3.4.3 show that this can significantly improve the detection performance.

Another factor to consider is the amount of information removed from an image. In the extreme case where the whole image is masked out, the reconstruction would be a random image from the in-domain manifold. This could lead to large reconstruction distance for both in-domain and out-of-domain images, especially when the in-domain distribution is diverse. Conversely, if only one pixel is removed from an image, then both in-domain and out-of-domain reconstructions would be highly faithful. Therefore, a mask should ideally provide sufficient clues for the diffusion model to map a lifted in-domain image close to its original location, while creating enough space to produce dissimilar out-of-domain reconstructions.

In this regard, we propose to use the **alternating checkerboard $N \times N$** mask (Figure 3). For simplicity, we assume that images are square-shaped with size $L \times L$; extension to rectangular-shaped images is straightforward. The checkerboard mask divides the image into an $N \times N$ grid of patches, where each patch has size $\frac{L}{N} \times \frac{L}{N}$. It masks out every other patch in a checkerboard-like fashion, covering 50% of an image in total. During multiple reconstructions, the masked and unmasked patches are flipped at each reconstruction attempt. This ensures that salient characteristics of an out-of-domain images are covered at some attempts. We default to $N = 8$. Experiments with different values of N can be found in Table 2.

2.2.2 Measuring reconstruction distance

We use the Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) metric to measure the distance between the original and reconstructed images. LPIPS utilizes calibrated intermediate activations of a pretrained neural network as features, and measures the normalized ℓ_2 distance between

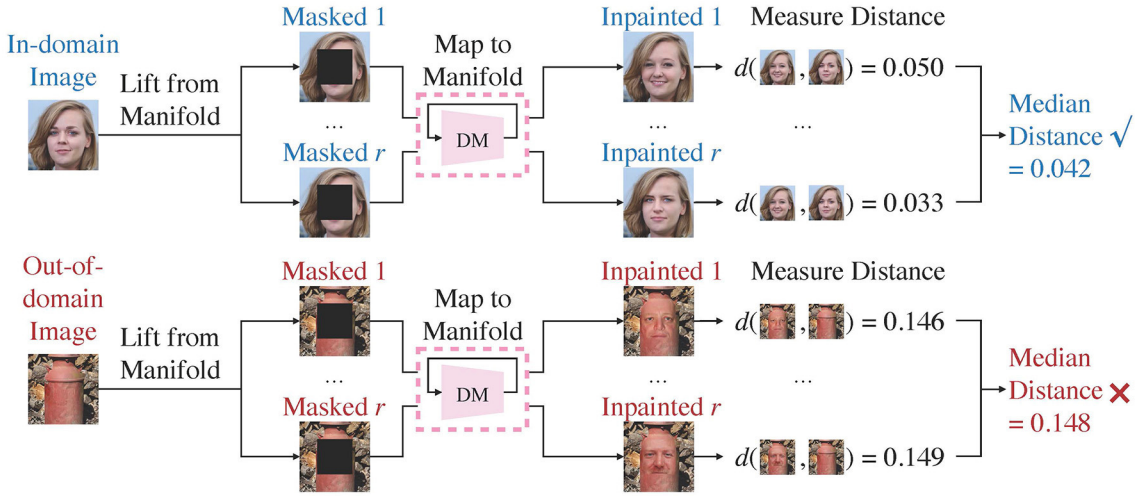


FIGURE 2
Overview of the LMD process. LMD utilizes a diffusion model trained over the in-domain manifold. It repeatedly lifts an image from its manifold by masking, and maps it toward the diffusion model's training manifold by inpainting. It measures the median distance between the original and the mapped images, and considers images with larger distance as out-of-domain.

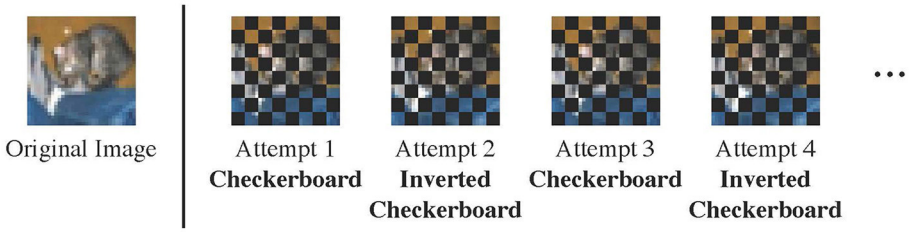


FIGURE 3
The alternating checkerboard mask. We flip the masked and unmasked regions at each reconstruction attempt. The example in the figure is 8 × 8.

the features of two images. This yields a value between 0 and 1, where lower value indicates higher similarity. We employ the version with AlexNet (Krizhevsky et al., 2012) backbone pretrained on ImageNet.[1] LPIPS has been observed to align with human perception of image similarity (Zhang et al., 2018), and has been applied in research on a wide range of tasks (Karras et al., 2019; Alaluf et al., 2021; Meng et al., 2021) and image modalities (Gong et al., 2021; Lugmayr et al., 2022; Toda et al., 2022). Experiments with alternative metric choices in Table 3.

3 Results

3.1 Experiment settings

We benchmark LMD against existing unsupervised OOD detection methods on widely used datasets. We provide fine-grained analysis and visualizations of the reconstructed images to better understand LMD's performance. Additionally, we perform ablation studies to analyze the individual components of LMD.

3.1.1 Baselines

We compare LMD with seven existing baselines, covering three mainstream classes of methods: likelihood-based, reconstruction-based and feature-based. For likelihood-based methods, we consider Likelihood **(Likelihood)** (Bishop, 1994), Input Complexity **(IC)** (Serrà et al., 2019) and Likelihood Regret **(LR)** (Xiao et al., 2020). We obtain the likelihood from the diffusion model using Song et al. (2020)'s approach.[2] We adapt the official GitHub repository of Likelihood Regret[3] for both Likelihood Regret and Input Complexity. For Input Complexity, we leverage the likelihood from the diffusion model to ensure fairness in comparison; we have experimented with both the PNG compressor and the JPEG compressor, and we report the results from the PNG compressor due to its superior performance. For reconstruction-based methods, we consider Reconstruction with Autoencoder and Mean Squared Error loss **(AE-MSE)**, AutoMahalanobis **(AE-MH)** (Denouden et al., 2018) and AnoGAN **(AnoGAN)** (Schlegl et al., 2017). For feature-based method, we consider Pretrained Feature Extractor + Mahalanobis Distance

1 We use the implementation of https://github.com/richzhang/PerceptualSimilarity.

2 https://github.com/yang-song/score_sde_pytorch

3 https://github.com/XavierXiao/Likelihood-Regret

(Pretrained) (Xiao et al., 2021). We use our own implementation as we did not find any existing implementation to our best efforts.

3.1.2 Evaluation

We evaluate the performance of LMD and the baselines using the area under Receiver Operating Characteristic curve (ROC-AUC), following the practice of existing works (Hendrycks and Gimpel, 2016; Ren et al., 2019; Xiao et al., 2021). OOD detection methods commonly produce numeric OOD scores, and apply a decision threshold to classify data as in-domain or out-of-domain. The ROC curve plots the true positive rate against the false positive rate at various decision thresholds, and ROC-AUC measures the area under the curve. ROC-AUC ranges between 0 and 1, with higher values indicating better performance. A detector achieves ROC-AUC > 0.5 when it in general assigns higher OOD scores to out-of-domain images than in-domain images. Conversely, it yields ROC-AUC < 0.5 when it in general assigns higher OOD scores for in-domain images.

3.1.3 Datasets

For quantitative evaluations, we consider pairwise combinations of CIFAR10 (Krizhevsky, 2009), CIFAR100 (Krizhevsky, 2009) and SVHN (Netzer et al., 2011), and pairwise combinations of MNIST (LeCun et al., 2010), KMNIST (Clanuwat et al., 2018), and FashionMNIST (Xiao et al., 2017), as the in-domain and out-of-domain datasets. This yields 12 pairs in total. For qualitative evaluations, we further present visualizations on two pairs of in-domain vs. out-of-domain datasets with higher image resolutions: CelebA-HQ (Karras et al., 2017) vs. ImageNet (Russakovsky et al., 2015), and LSUN bedroom (Yu et al., 2015) vs. LSUN classroom (Yu et al., 2015). We standardize these images to 256×256.

3.1.4 Implementation details of LMD

We build LMD on top of Song et al. (2020)'s implementation. For datasets in Table 3, we use DDPM++ models with SubVP SDE. We take Song et al. (2020)'s pretrained CIFAR10 checkpoint, and train from scratch for the other datasets. We use alternating checkerboard 8×8 mask (Figure 3), reconstruction distance metric LPIPS and 10 reconstructions per image for LMD.

For the higher resolution datasets, we use NCSN++ models with VE SDE. We take Song et al. (2020)'s pretrained FFHQ (Karras et al., 2019) checkpoint for CelebA-HQ vs. ImageNet. This is to avoid model memorization concerns given that the CelebA-HQ checkpoint is pretrained over the whole dataset. We use Song et al. (2020)'s pretrained LSUN bedroom checkpoint for LSUN bedroom vs. LSUN classroom. For these datasets, we consider a checkerboard 4×4 mask, a checkerboard 8×8 mask and a square-centered mask, with *one* reconstruction per image. We additionally report the ROC-AUC from our default configuration of alternating 8×8 checkerboard and 10 reconstructions per image as a reference. We use LPIPS as the distance metric.

3.2 Quantitative results and analysis

We present the OOD detection performance of LMD and the baselines on 12 dataset pairs in Table 1. LMD attains the highest ROC-AUC on five pairs, while demonstrating consistent and strong performance on others. Specifically, on CIFAR100 vs. SVHN, it attains 10% higher ROC-AUC than the best baseline performance. LMD also attains the highest average ROC-AUC of 0.907, which is 9% higher than the best average performance among the baselines. We visualize examples of the in-domain and out-of-domain reconstructions of LMD in Figure 4. In general, in-domain reconstructions resemble their original images, while out-of-domain reconstructions are fragmented and noisy.

We further conduct fine-grained analysis to understand LMD's performance. We observe that each dataset in Table 1 consists of images from multiple distinct semantic categories, forming a diverse data distribution. For example, CIFAR10 comprises 10 different objects or animals, and SVHN comprises 10 digits. We seek to understand whether LMD performs similarly across different semantic categories, or if certain categories are more challenging for LMD than the others. Specifically, we group the images by their ground truth classes, and examine the distinguishability of the OOD scores for each pair of classes of the in-domain vs. out-of-domain datasets. We present the results for CIFAR10 vs. SVHN and SVHN vs. CIFAR10 in Figure 5. On CIFAR10 vs. SVHN, all pairs of classes are highly distinguishable, with ROC-AUC ranging from 0.97 to 1. This is unsurprising given that LMD attains strong performance of ROC-AUC 0.992 on this pair. On SVHN vs. CIFAR10, pairwise performance shows visible variation, with ROC-AUC ranging from 0.84 to 0.97. Specifically, the ROC-AUC is relatively low when the in-domain class is "3" or "5," and when the out-of-domain class is "deer" or "frog." This suggests that the reason behind LMD's satisfactory but suboptimal performance on SVHN vs. CIFAR10 is primarily attributed to the relative difficulty in distinguishing between some of the semantic categories.

3.3 Qualitative studies on higher resolution images

We show qualitative results on images with resolution 256×256 for two in-domain/out-of-domain pairs: CelebA-HQ vs. ImageNet (Figure 6) and LSUN bedroom vs. LSUN classroom (Figure 7). The ROC-AUCs in the images correspond to LMD's performance with only *one* reconstruction attempt. As a reference, under our default configuration of alternating checkerboard 8×8 mask and 10 reconstruction attempts, CelebA-HQ vs. ImageNet has a ROC-AUC of **0.993**, and LSUN bedroom vs. LSUN classroom has a ROC-AUC of **0.927**.

For CeleA-HQ vs. ImageNet, LMD performs competitively under all three mask choices, and achieves ROC-AUC ranging from 0.991 to 1 even without multiple reconstructions. Given the highly structured nature of human faces, the in-domain reconstructions under all three masks are accurate. For the out-of-domain images, reconstructions under the checkerboard masks contain local

TABLE 1 ROC-AUC of LMD and the baselines.

ID	OOD	Likelihood	IC	LR	Pretrained	AE-MSE	AE-MH	AnoGAN	LMD
CIFAR10	CIFAR100	0.520	0.568	0.546	**0.806**	0.510	0.488	0.518	0.607
	SVHN	0.180	0.870	0.904	0.888	0.025	0.073	0.120	**0.992**
CIFAR100	CIFAR10	0.495	0.468	0.484	0.543	0.509	0.486	0.510	**0.568**
	SVHN	0.193	0.792	0.896	0.776	0.027	0.122	0.131	**0.985**
SVHN	CIFAR10	0.974	0.973	0.805	**0.999**	0.981	0.966	0.967	0.914
	CIFAR100	0.970	0.976	0.821	**0.999**	0.980	0.966	0.962	0.876
MNIST	KMNIST	0.948	0.903	0.999	0.887	0.999	**1.000**	0.933	0.984
	FashionMNIST	0.997	**1.000**	0.999	0.999	**1.000**	**1.000**	0.992	0.999
KMNIST	MNIST	0.152	0.951	0.431	0.582	0.102	0.217	0.317	**0.978**
	FashionMNIST	0.833	**0.999**	0.557	0.993	0.896	0.868	0.701	0.993
FashionMNIST	MNIST	0.172	0.912	0.971	0.647	0.804	0.969	0.835	**0.992**
	KMNIST	0.542	0.584	0.994	0.730	0.976	**0.996**	0.912	0.990
Average		0.581	0.833	0.783	0.821	0.651	0.679	0.658	**0.907**

Higher value is better. We use the default configuration of alternating checkerboard 8 × 8, LPIPS metric and 10 reconstructions per image for all experiments. LMD consistently demonstrates strong performance and attains the highest average ROC-AUC. The bold values mean the best performance, i.e., the highest ROC-AUC, among the evaluated methods in each setting, where a setting is either an in-domain vs. out-of-domain dataset pair or the average across all dataset pairs.

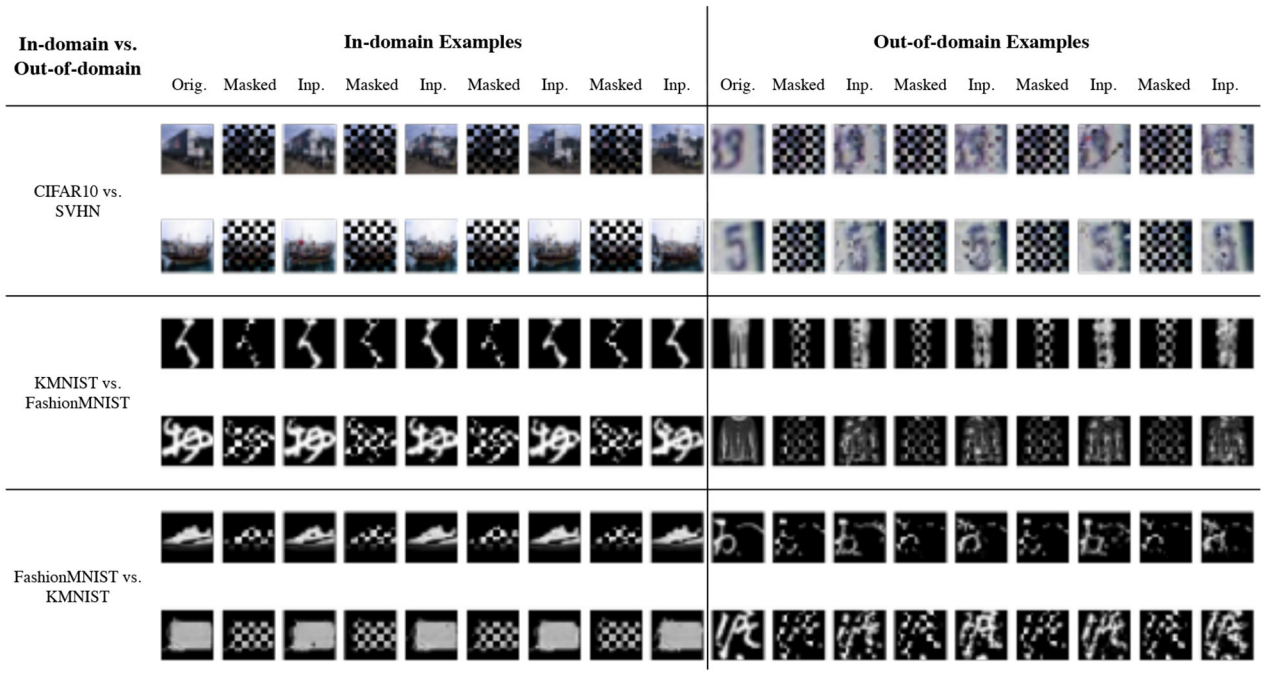


FIGURE 4
Example reconstructions from three pairs of dataset. "Orig." is the original image and "Inp." is the inpainted image. Generally, the in-domain reconstructions are faithful while the out-of-domain reconstructions are noisy and dissimilar.

distortions, while reconstructions under the center mask tend to hallucinate faces. As a result, in this case, the in-domain and out-of-domain reconstructions become more discernible when employing larger patches in masking.

For LSUN bedroom vs. LSUN classroom, the checkerboard 8×8 mask attains strong results, while the checkerboard 4 × 4 mask and the center-squared mask demonstrate suboptimal performance. This is because bedroom images exhibit greater variation and contain more intricate details. Consequently, when large patches are masked, the diffusion model may fill in plausible yet different content, resulting in significant reconstruction discrepancies for in-domain images. In fact, even with the checkerboard 8 × 8 mask, the diffusion model may hallucinate or alter elements in the bedroom inpaintings. Moreover, the complex and diverse nature

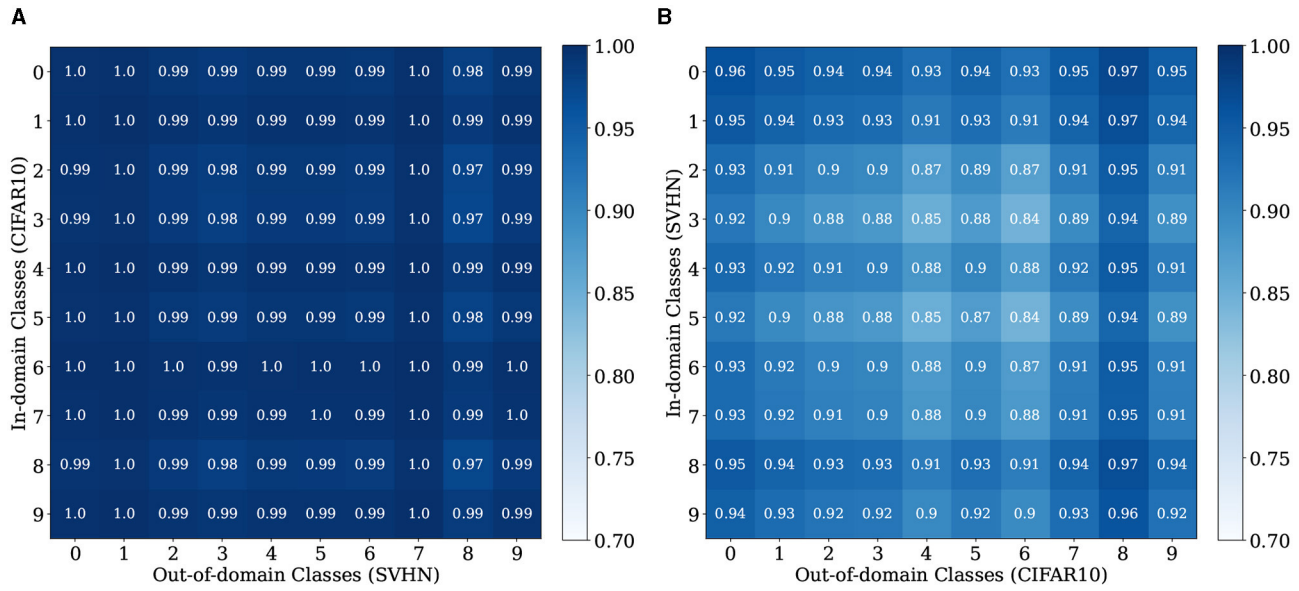


FIGURE 5
Per-Class ROC-AUC for CIFAR10 vs. SVHN and SVHN vs. CIFAR10. The classes for CIFAR10 are: 1, airplane; 2, automobile; 3, bird; 4, cat; 5, deer; 6, dog; 7, frog; 8, horse; 9, ship; and 10, truck. **(A)** CIFAR10 vs. SVHN. **(B)** SVHN vs. CIFAR10.

of bedroom images poses substantial challenges for the diffusion model to accurately learn the in-domain distribution; samples and inpaintings from the LSUN bedroom model generally have lower quality than those from the CelebA-HQ model.

Results from these two dataset pairs collectively demonstrate that LMD could scale to higher resolution images with richer details. They also highlight the checkerboard 8 × 8 mask as a versatile default choice, as it is effective for both structured and diverse in-domain distributions. For further discussions on mask choices, please refer to Section 3.4.1.

3.4 Ablation studies

3.4.1 Mask choice

Table 2 presents the performance of LMD under alternative mask choices. Besides our default mask, we consider alternating checkerboard 4 × 4, alternating checkerboard 16 × 16, a fixed 8 × 8 checkerboard for which we do not perform the flipping operation, a square-centered mask, and a random patch mask following (Xie et al., 2022).[4] Figure 8 visualizes the mask patterns. We experiment on three dataset pairs: CIFAR10 vs. CIFAR100, CIFAR10 vs. SVHN and MNIST vs. KMNIST. For all the mask choices, we perform 10 reconstructions per image and use LPIPS as the reconstruction distance metric.

Our default mask choice of alternating checkerboard 8 × 8 shows consistent and strong performance. Alternating checkerboard 16 × 16 mask, fixed checkerboard 8 × 8 mask and the random patch mask are competitive but underperform the default choice. Nevertheless, alternating checkerboard mask is recommended over fixed checkerboard mask or random patch mask, as it ensures that all parts of the image are covered in some of the reconstruction attempts. Alternating checkerboard 4 × 4 and square-centered masks show suboptimal performance on MNIST vs. KMNIST. This is because they mask out too much information from the images, and therefore lead to unfaithful reconstructions for both in-domain and out-of-domain images.

3.4.2 Reconstruction distance metric

We study the effect of using alternative metrics for measuring the reconstruction distance. We consider two popular metrics, Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) (Wang et al., 2003), both of which have been widely used for image comparison (Zhang et al., 2019; Bhat et al., 2021; Saharia et al., 2022). We further observe that Xiao et al. (2021) uses features from a ResNet-50 pretrained with SimCLRv2 (Chen et al., 2020) on ImageNet, and achieves superior performance on CIFAR10 vs. CIFAR100. Thus, we also consider a SimCLRv2-based metric, in which we calculate the cosine distance between the SimCLRv2 features of the original and reconstructed images.

We present the performance of LMD under different distance metrics in Table 3. MSE and SSIM demonstrate poor performance when SVHN is the out-of-domain dataset. Our default choice LPIPS demonstrates strong and consistent performance, and attains the highest average ROC-AUC. SimCLRv2 is competitive but underperforms LPIPS. This suggests that deep feature based metrics are in general effective, and LPIPS is suitable as a default choice.

4 https://github.com/microsoft/SimMIM

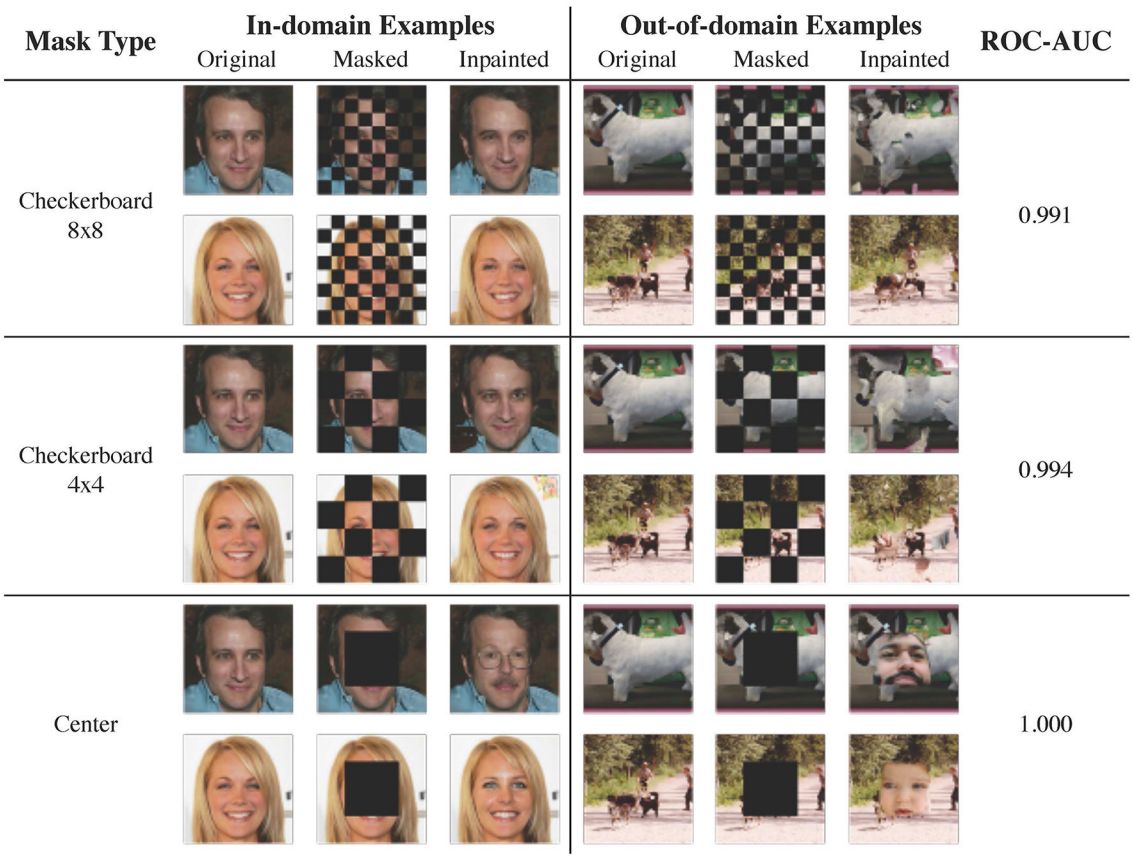


FIGURE 6
Examples of image reconstruction from CelebA-HQ (in-domain) and ImageNet (out-of-domain). For out-of-domain reconstructions, the checkerboard masks result in local inconsistencies, while the center mask hallucinates faces. In this case, employing larger masked patches slightly improves the performance.

3.4.3 Number of reconstructions per image

We examine LMD's performance under different number of reconstructions per image. Figure 9 plots the ROC-AUC against the number of reconstructions per image for MNIST vs. KMNIST and KMNIST vs. MNIST. LMD's performance improves as the number of reconstructions increases, regardless of the choice of distance metric. The improvement is especially obvious for the first 5 attempts, and gradually plateaus as the number of attempts approaches 10. This suggests that it is generally sufficient to perform 10 attempts per image.

3.4.4 Alternative instantiation of lifting and mapping

We observe that another intuitive way of lifting and mapping images with a diffusion model is to lift by diffusion to an intermediate step t in the noise schedule, and denoising back to the image distribution. We refer to this alternative instantiation as diffusion/denoising, and compare it with our default instantiation of masking/inpainting. Given that the image distribution is at $t = 0$ and the noise distribution is at $t = T$, the larger t we diffuse to, the further away we lift an image from the manifold. We consider different lifting distances with $t = 250$, $t = 500$, and $t = 750$, where the full schedule has $T = 1000$. We use our default alternating checkerboard 8×8 mask for masking/inpainting. We use 10 reconstructions per image and the LPIPS metric for both diffusion/denoising and masking/inpainting.

We present the performance in Table 4. Diffusion/denoising with $t = 250$ and $t = 750$ demonstrate suboptimal performance on several pairs, indicating that the lifting distance is too small or too large for the in-domain and out-of-domain to be distinguishable. $t = 500$ is competitive but underperforms masking/inpainting. This suggests that while LMD is robust to alternative choices of lifting and mapping, masking/inpainting is the recommended instantiation.

3.4.5 Alternative choices for the inpainting model

We perform qualitative evaluation on using other classes of inpainting models in the LMD framework. We consider **Masked Autoencoder (MAE)** (He et al., 2022) trained on CIFAR10,[5] and **LaMa** (Suvorov et al., 2022),[6] a GAN-based inpainting model, trained on CelebA-HQ. We perform *one* reconstruction per image, as both MAE and LaMa are deterministic.

5 https://github.com/IcarusWizard/MAE
6 https://github.com/advimman/lama

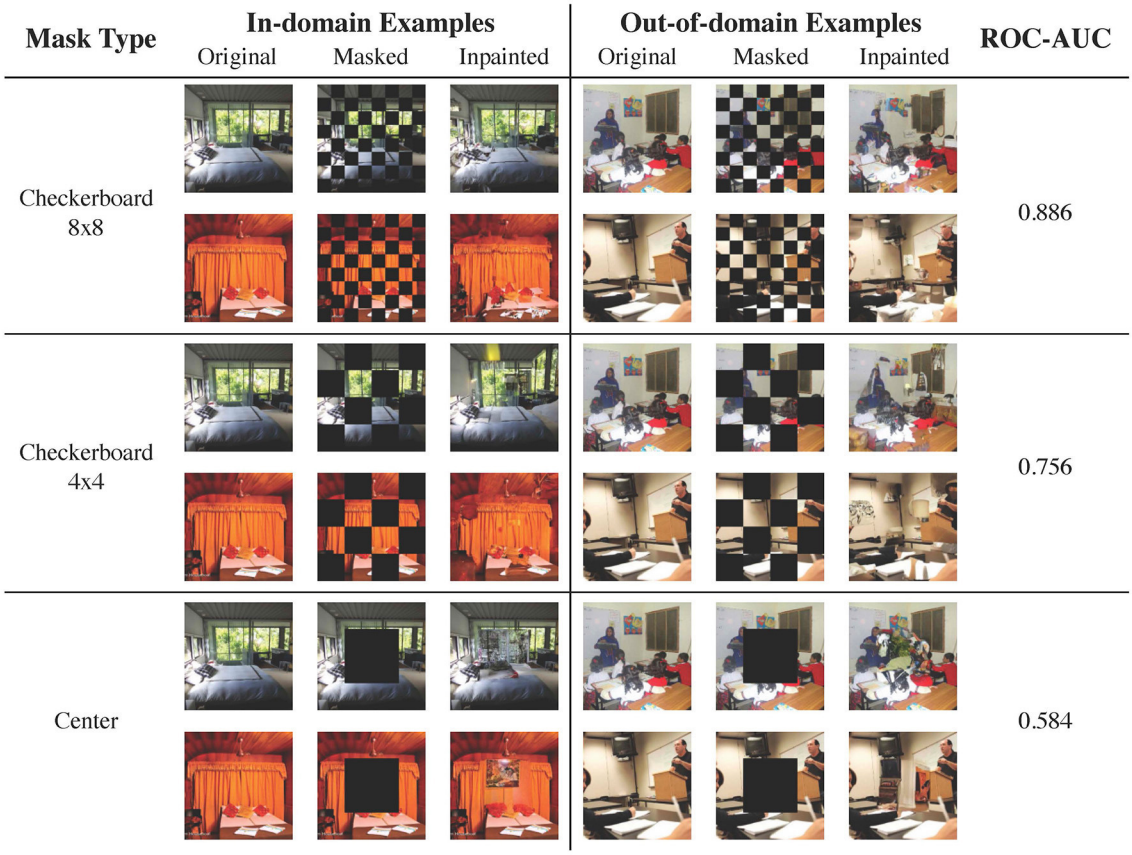


FIGURE 7
Reconstruction examples from LSUN bedroom (in-domain) and LSUN classroom (out-of-domain). As bedroom images are diverse and contain richer details, a mask with smaller patches is preferrable.

TABLE 2 **Performance of ROC-AUC on three dataset pairs with different mask types.**

Mask type	CIFAR10 vs. CIFAR100	CIFAR10 vs. SVHN	MNIST vs. KMNIST
Alternating checkerboard 4 × 4	0.594	0.987	0.923
Alternating checkerboard 8 × 8	**0.607**	**0.992**	0.984
Alternating checkerboard 16 × 16	0.597	0.981	**0.997**
Fixed checkerboard 8 × 8	0.601	0.990	0.974
Center	0.570	0.978	0.479
Random patch	0.591	0.990	0.912

The alternating checkerboard 8 × 8 shows strong and consistent results. The bold values mean the best performance, i.e., the highest ROC-AUC, among the evaluated methods in each setting, where a setting is either an in-domain vs. out-of-domain dataset pair or the average across all dataset pairs.

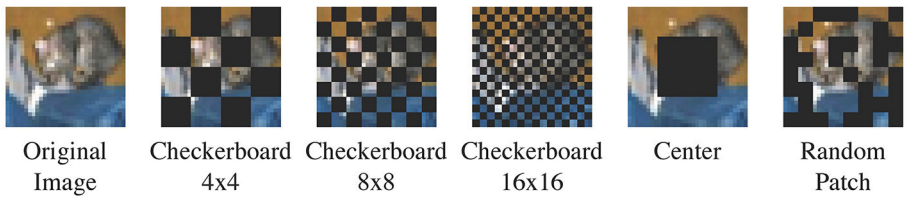


FIGURE 8
Masks used in the mask ablation. The random patch mask in the figure is just one example; a different pattern is sampled each time.

TABLE 3 ROC-AUC performance under different reconstruction distance metrics.

ID	OOD	MSE	SSIM	LPIPS	SimCLRv2
CIFAR10	CIFAR100	0.548	0.624	0.607	0.713
	SVHN	0.155	0.329	0.992	0.970
CIFAR100	CIFAR10	0.549	0.551	0.568	0.523
	SVHN	0.157	0.258	0.985	0.924
SVHN	CIFAR10	0.987	0.998	0.914	0.933
	CIFAR100	0.979	0.995	0.876	0.928
MNIST	KMNIST	0.998	0.997	0.984	0.983
	FashionMNIST	0.995	0.999	0.999	0.999
KMNIST	MNIST	0.835	0.922	0.978	0.920
	FashionMNIST	0.802	0.979	0.993	0.995
FashionMNIST	MNIST	0.993	0.960	0.992	0.961
	KMNIST	0.998	0.988	0.990	0.977
Average		0.750	0.800	**0.907**	0.902

LPIPS demonstrates consistent and robust results, while other metrics exhibit performance fluctuations. The bold values mean the best performance, i.e., the highest ROC-AUC, among the evaluated methods in each setting, where a setting is either an in-domain vs. out-of-domain dataset pair or the average across all dataset pairs.

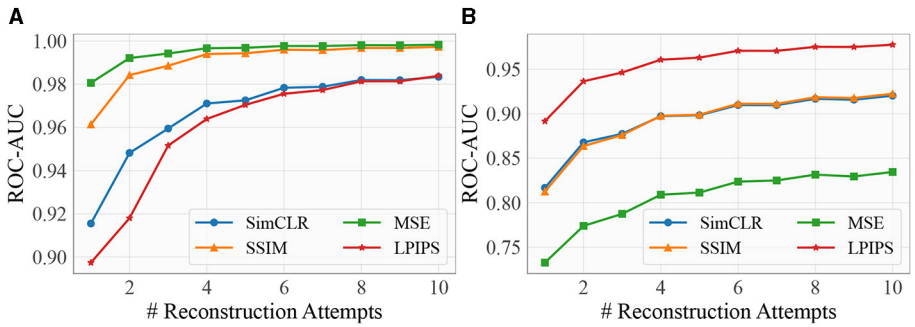


FIGURE 9
ROC-AUC vs. the number of reconstruction attempts. More reconstruction attempts enhances the OOD detection performance, irrespective of the distance metric. **(A)** MNIST vs. KMNIST. **(B)** KMNIST vs. MNIST.

Both models demonstrate lower performance than the diffusion model in various scenarios. Figure 10 shows LaMa's performance on CelebA-HQ vs. ImageNet. LaMa attains reasonable results, but it underperforms diffusion models. LaMa hallucinates faces with the center mask, but unlike the diffusion model, the color and texture of the hallucinated faces are very consistent with the surroundings.

Figure 11 shows MAE's performance on CIFAR10 vs. SVHN. Both in-domain and out-of-domain reconstructions are accurate when the individual masked patch sizes are small, while both deviate from the originals when the patch sizes are large. Performance-wise, inpainting with MAE only attains ROC-AUC 0.065 for checkerboard 8×8 mask, 0.178 for checkerboard 4×4 mask and 0.403 for center mask.

The suboptimal performance of alternative inpainting models can be attributed to their ability to leverage various sources of information—from not only its understanding of the training distribution, but also color or texture of unmasked parts of an image. Models like LaMa and MAE employ specialized loss functions and large masked ratios during training, and thus excel at inferring missing regions from known ones regardless of semantics. Consequently, these models are more prone to producing reasonable out-of-domain inpaintings, especially with simpler out-of-domain images. In contrast, a vanilla diffusion model is not specifically trained for inferring missing regions from the surroundings. It primarily relies on its understanding of the training distribution to perform inpainting, and thus attains robust performance.

4 Discussion

4.1 LMD's relationship with existing works

In the unsupervised setting, existing works generally follow one of the three paradigms: likelihood-based, reconstruction-based and feature-based. LMD is a reconstruction-based approach. Typically, reconstruction-based methods involve

TABLE 4 ROC-AUC performance of using diffusion/denoising vs. masking/inpainting.

ID	OOD	Denoising ($t = 250$)	Denoising ($t = 500$)	Denoising ($t = 750$)	Inpainting
CIFAR10	CIFAR100	0.583	0.600	0.589	**0.607**
	SVHN	0.967	0.976	0.954	**0.992**
CIFAR100	CIFAR10	**0.568**	0.524	0.436	**0.568**
	SVHN	0.949	0.957	0.904	**0.985**
SVHN	CIFAR10	0.861	**0.966**	0.957	0.914
	CIFAR100	0.847	0.949	**0.957**	0.876
MNIST	KMNIST	0.956	**0.993**	0.715	0.984
	FashionMNIST	0.998	0.998	0.927	**0.999**
KMNIST	MNIST	0.645	0.972	0.721	**0.978**
	FashionMNIST	**0.998**	0.994	0.943	0.993
FashionMNIST	MNIST	0.428	0.941	0.876	**0.992**
	KMNIST	0.567	0.943	0.862	**0.990**
Average		0.781	0.901	0.820	**0.907**

Diffusion/denoising with $t = 500$ achieves reasonable performance but underperforms diffusion/inpainting. The bold values mean the best performance, i.e., the highest ROC-AUC, among the evaluated methods in each setting, where a setting is either an in-domain vs. out-of-domain dataset pair or the average across all dataset pairs.

training a model using in-domain samples, and assessing the reconstruction quality of a test data point under the model. Prior works commonly use autoencoders (Sakurada and Yairi, 2014; Xia et al., 2015; Zhou and Paffenroth, 2017; Zong et al., 2018) or GANs (Schlegl et al., 2017; Li et al., 2018). One concurrent work (Graham et al., 2022) utilizes diffusion models, and considers image reconstructions under varying numbers of diffusion and denoising steps. This contrasts with LMD, which repeatedly performs masking and inpainting with fixed number of steps. These two approaches are orthogonal and complementary.

The likelihood-based paradigm has been extensively explored, with early contributions dating back to Bishop (1994). The core idea is to approximate the in-domain distribution with a generative model that has likelihood computation capability (Salimans et al., 2017; Kingma and Dhariwal, 2018). Intuitively, the model should assign higher likelihood to in-domain data than out-of-domain data, but various studies have observed that such assumption often does not hold (Choi et al., 2018; Nalisnick et al., 2018; Kirichenko et al., 2020). One line of work addresses this issue under a typicality test framework (Ren et al., 2019; Serrà et al., 2019; Xiao et al., 2020). Essentially, they view likelihood as a model statistic rather than a literal measure of how likely a data point is in-domain. They examine the extent to which the model statistic of a test data point deviates from the typical distribution of model statistics for in-domain data. Notably, this is complementary to LMD, as the reconstruction distance can also be viewed as a model statistic. Other likelihood-based approaches include adjusting the likelihood by background likelihood (Ren et al., 2019), image complexity (Serrà et al., 2019) or the likelihood under optimal model configurations (Xiao et al., 2020), or improving the generative model architectures (Maaløe et al., 2019; Kirichenko et al., 2020).

The feature-based paradigm usually involves extracting lower-dimensional features from the data from unsupervised sources, such as autoencoders (Denouden et al., 2018), generative models (Ahmadian and Lindsten, 2021), self-supervised training (Hendrycks et al., 2019; Bergman and Hoshen, 2020; Tack et al., 2020; Sehwag et al., 2021) or pretrained feature extractors (Xiao et al., 2021). They then perform detection in lower-dimensional space, typically with simple techniques like fitting one-class Support Vector Machines or Gaussian Mixture Models.

4.2 Limitation and future work

One limitation of LMD is the speed. Vanilla diffusion models have a time-consuming denoising process that involves a large number of sampling steps. Therefore, similar to other diffusion-based approaches for various tasks (Meng et al., 2021; Lugmayr et al., 2022; Saharia et al., 2022), LMD is currently not well-suited for real-time OOD detection. Several recent works have proposed methods to accelerate the sampling process of pre-trained diffusion models through noise rescaling (Nichol and Dhariwal, 2021), sampler optimization (Watson et al., 2022), or numerical methods (Liu et al., 2022; Wizadwongsa and Suwajanakorn, 2023). One future direction is to harness these methods to expedite LMD's detection.

Another potential extension is to utilize more advanced methods for aggregating reconstruction distances from multiple reconstructions, or even under different masks or distance metrics. As briefly discussed in Section 4.1, this can involve integrating typicality test approaches such as multiple hypothesis testing or learning density models (Nalisnick et al., 2019; Morningstar et al., 2021; Bergamin et al., 2022).

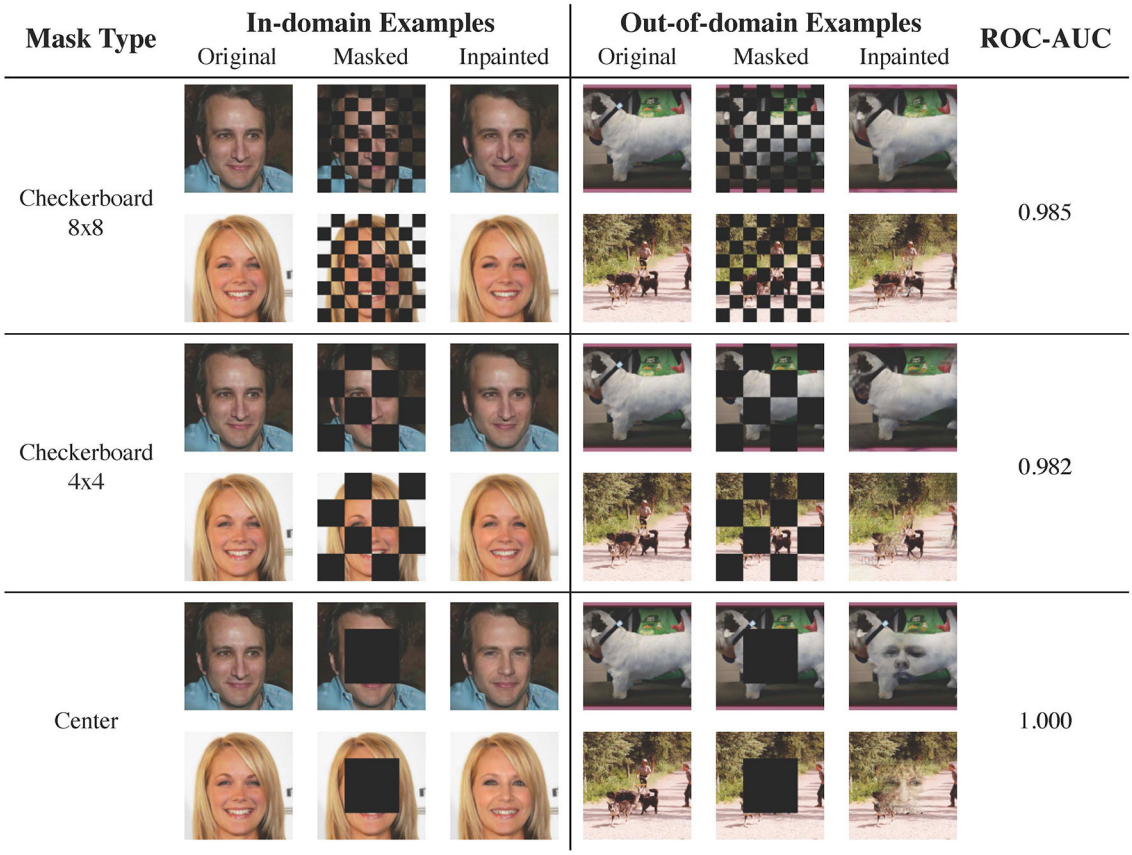


FIGURE 10
Reconstruction examples from CelebA-HQ (in-domain) and ImageNet (out-of-domain) using LaMa, a GAN-based inpainting model. Unlike the diffusion model, LaMa produces less visible artifacts. Even though it also introduces face-like artifacts with the center mask, the faces have the colors and textures of the surrounding unmasked regions.

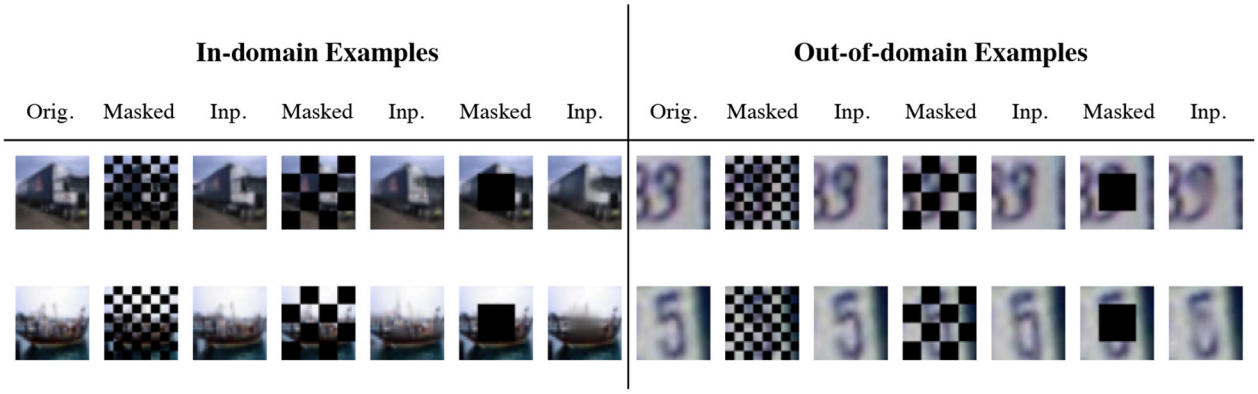


FIGURE 11
Reconstruction examples from CIFAR10 (in-domain) and SVHN (out-of-domain) using MAE. Differentiating between in-domain and out-of-domain inpaintings are hard, because reconstructing SVHN from only known regions is relatively simple, and because MAE is trained to have strong capability of inference from known regions.

5 Conclusion

We propose a novel method, *Lift, Map, Detect* (LMD), for unsupervised out-of-distribution detection. LMD leverages the diffusion model's strong ability in mapping images onto its training manifold, and detects images with large distance between the original and mapped images as OOD. Our extensive experiments and analysis show that LMD achives strong performance for

various image distributions with different characteristics. Some future directions of improvement include accelerating LMD's speed and leveraging advanced aggregation for reconstruction distance.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: CIFAR10, CIFAR100, SVHN, MNIST, KMNIST, and FashionMNIST: can be accessed through https://pytorch.org/vision/stable/datasets.html. CelebA-HQ: https://github.com/tkarras/progressive_growing_of_gans. ImageNet: https://www.image-net.org/. LSUN bedroom and LSUN classroom: https://github.com/fyu/lsun.

Ethics statement

Written informed consent was not obtained from the individual(s) for the publication of any potentially identifiable images or data included in this article, because these human face images are either from the public datasets CelebA-HQ and FFHQ, which are widely used in the machine learning and computer vision communities, or synthetic faces created by generative models.

Author contributions

ZL and JZ contributed to the design of the research, performed the experiments, and wrote the manuscript. KW is the PhD supervisor of ZL and JZ, he conceptualized and directed the research, and revised the manuscript. All authors approved the submitted version.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. This research was supported by grants from DARPA AIE program, Geometries of Learning (HR00112290078), the Natural Sciences and Engineering Research Council of Canada (NSERC) (567916), the National Science Foundation NSF (IIS-2107161, III1526012, IIS-1149882, and IIS-1724282), and the Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875).

Acknowledgments

We would like to thank Yufan Wang for helping with literature search and initial setups of some of the baselines.

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.

Publisher's note

References

Ahmadian, A., and Lindsten, F. (2021). "Likelihood-free out-of-distribution detection with invertible generative models," in *IJCAI*, 2119–2125. doi: 10.24963/ijcai.2021/292

Alaluf, Y., Patashnik, O., and Cohen-Or, D. (2021). "Restyle: a residual-based stylegan encoder via iterative refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6711–6720. doi: 10.1109/ICCV48922.2021.00664

Bergamin, F., Mattei, P.-A., Havtorn, J. D., Senetaire, H., Schmutz, H., Maaløe, L., et al. (2022). "Model-agnostic out-of-distribution detection using combined statistical tests," in *International Conference on Artificial Intelligence and Statistics* (PMLR), 10753-10776.

Bergman, L., and Hoshen, Y. (2020). Classification-based anomaly detection for general data. *arXiv preprint arXiv:2005.02359*.

Bhat, S. F., Alhashim, I., and Wonka, P. (2021). "Adabins: depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4009–4018.

Bishop, C. M. (1994). Novelty detection and neural network validation. *IEE Proc. Vision Image Sig. Proc*. 141, 217–222. doi: 10.1049/ip-vis:19941330

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. (2020). "Big self-supervised models are strong semi-supervised learners," in *Advances in Neural Information Processing Systems*, 22243–22255.

Choi, H., Jang, E., and Alemi, A. A. (2018). Waic, but why? Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical Japanese literature. *arXiv preprint arXiv:1812.01718*.

Denouden, T., Salay, R., Czarnecki, K., Abdelzad, V., Phan, B., and Vernekar, S. (2018). Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. *arXiv preprint arXiv:1812.02765*.

Gong, Y., Liao, P., Zhang, X., Zhang, L., Chen, G., Zhu, K., et al. (2021). Enlighten-gan for super resolution reconstruction in mid-resolution remote sensing images. *Rem. Sens*. 13:1104. doi: 10.3390/rs13061104

Graham, M. S., Pinaya, W. H., Tudosiu, P.-D., Nachev, P., Ourselin, S., and Cardoso, M. J. (2022). Denoising diffusion models for out-of-distribution detection. *arXiv preprint arXiv:2211.07740*. doi: 10.1109/CVPRW59228.2023.00296

Hamet, P., and Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism* 69, S36–S40. doi: 10.1016/j.metabol.2017.01.011

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16000–16009. doi: 10.1109/CVPR52688.2022.01553

Hendrycks, D., and Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Hendrycks, D., Mazeika, M., and Dietterich, T. (2018). Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*.

Hendrycks, D., Mazeika, M., Kadavath, S., and Song, D. (2019). "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems* 32.

Ho, J., Jain, A., and Abbeel, P. (2020). "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 6840–6851.

Huang, R., Geng, A., and Li, Y. (2021). "On the importance of gradients for detecting distributional shifts in the wild," in *Advances in Neural Information Processing Systems*, 677–689.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *CoRR, abs/1710.10196*.

Karras, T., Laine, S., and Aila, T. (2019). "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4401–4410. doi: 10.1109/CVPR.2019.00453

Kingma, D. P., and Dhariwal, P. (2018). "Glow: generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems* 31.

Kirichenko, P., Izmailov, P., and Wilson, A. G. (2020). "Why normalizing flows fail to detect out-of-distribution data," in *Advances in Neural Information Processing Systems*, 20578–20589.

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. Technical report.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* 25.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs*. Available online at: http://yann.lecun.com/exdb/mnist (accessed July 08, 2023).

Lee, K., Lee, K., Lee, H., and Shin, J. (2018). "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems* 31.

Li, D., Chen, D., Goh, J., and Ng, S.- K. (2018). Anomaly detection with generative adversarial networks for multivariate time series. *arXiv preprint arXiv:1809.04758*.

Liang, S., Li, Y., and Srikant, R. (2017). Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

Liu, L., Ren, Y., Lin, Z., and Zhao, Z. (2022). Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. (2022). "Repaint: inpainting using denoising diffusion probabilistic models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11461–11471. doi: 10.1109/CVPR52688.2022.01117

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. (2019). "Biva: a very deep hierarchy of latent variables for generative modeling," in *Advances in Neural Information Processing Systems* 32.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., et al. (2021). "Sdedit: guided image synthesis and editing with stochastic differential equations," in *International Conference on Learning Representations*.

Morningstar, W., Ham, C., Gallagher, A., Lakshminarayanan, B., Alemi, A., and Dillon, J. (2021). "Density of states estimation for out of distribution detection," in *International Conference on Artificial Intelligence and Statistics* (PMLR), 3232–3240.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. (2018). Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*.

Nalisnick, E. T., Matsukawa, A., Teh, Y. W., and Lakshminarayanan, B. (2019). Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*.

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*, 7.

Nichol, A. Q., and Dhariwal, P. (2021). "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning* (PMLR), 8162–8171.

Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., et al. (2019). "Likelihood ratios for out-of-distribution detection," in *Advances in Neural Information Processing Systems* 32.

Rigano, C. (2019). Using artificial intelligence to address criminal justice needs. *Natl. Inst. Justice J.* 280, 1–10.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R., et al. (2019). Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*.

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

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2022). Image super-resolution via iterative refinement. *IEEE Trans. Patt. Analy. Mach. Intell.* 45, 4713–4726. doi: 10.1109/TPAMI.2022.3204461

Sakurada, M., and Yairi, T. (2014). "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 4–11. doi: 10.1145/2689746.2689747

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging* (Springer), 146–157. doi: 10.1007/978-3-319-59050-9_12

Sehwag, V., Chiang, M., and Mittal, P. (2021). Ssd: A unified framework for self-supervised outlier detection. *arXiv preprint arXiv:2103.12051*.

Serrà, J., Álvarez, D., Gómez, V., Slizovskaia, O., Nú nez, J. F., and Luque, J. (2019). Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning* (PMLR), 2256–2265.

Song, Y., and Ermon, S. (2019). "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems* 32.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., et al. (2022). "Resolution-robust large mask inpainting with fourier convolutions," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2149–2159. doi: 10.1109/WACV51458.2022.00323

Tack, J., Mo, S., Jeong, J., and Shin, J. (2020). "CSI: novelty detection via contrastive learning on distributionally shifted instances," in *Advances in Neural Information Processing Systems* 11839–11852.

Toda, R., Teramoto, A., Kondo, M., Imaizumi, K., Saito, K., and Fujita, H. (2022). Lung cancer ct image generation from a free-form sketch using style-based pix2pix for data augmentation. *Sci. Rep.* 12:12867. doi: 10.1038/s41598-022-16861-5

Wang, H., Li, Z., Feng, L., and Zhang, W. (2022). "Vim: out-of-distribution with virtual-logit matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4921–4930. doi: 10.1109/CVPR52688.2022.00487

Wang, Z., Simoncelli, E. P., and Bovik, A. C. (2003). "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers* (IEEE), 1398–1402.

Watson, D., Chan, W., Ho, J., and Norouzi, M. (2022). "Learning fast samplers for diffusion models by differentiating through sample quality," in *International Conference on Learning Representations*.

Wizadwongsa, S., and Suwajanakorn, S. (2023). Accelerating guided diffusion sampling with splitting numerical methods. *arXiv preprint arXiv:2301.11558*.

Xia, Y., Cao, X., Wen, F., Hua, G., and Sun, J. (2015). "Learning discriminative reconstructions for unsupervised outlier removal," in *Proceedings of the IEEE International Conference on Computer Vision*, 1511–1519. doi: 10.1109/ICCV.2015.177

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR, abs/1708.07747*.

Xiao, Z., Yan, Q., and Amit, Y. (2020). "Likelihood regret: an out-of-distribution detection score for variational auto-encoder," in *Advances in Neural Information Processing Systems*, 20685–20696.

Xiao, Z., Yan, Q., and Amit, Y. (2021). Do we really need to learn representations from in-domain data for outlier detection? *arXiv preprint arXiv:2105.09270*.

Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., et al. (2022). "Simmim: a simple framework for masked image modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9653–9663. doi: 10.1109/CVPR52688.2022.00943

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., et al. (2022). Diffusion models: a comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*.

Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. (2015). Lsun: construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Zhang, K. A., Cuesta-Infante, A., Xu, L., and Veeramachaneni, K. (2019). Steganogan: high capacity image steganography with gans. *arXiv preprint arXiv:1901.03892*.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 586–595. doi: 10.1109/CVPR.2018.00068

Zhou, C., and Paffenroth, R. C. (2017). "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 665–674. doi: 10.1145/3097983.3098052

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., et al. (2018). "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*.

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 22 May 2024
DOI 10.3389/fcomp.2024.1274779

OPEN ACCESS

EDITED BY
Yi-Zhe Song,
University of Surrey, United Kingdom

REVIEWED BY
Aneeshan Sain,
University of Surrey, United Kingdom
Ruoyi Du,
Beijing University of Posts and Telecommunications (BUPT), China

*CORRESPONDENCE
Ankita Shukla
ankitas@unr.edu

RECEIVED 08 August 2023
ACCEPTED 29 February 2024
PUBLISHED 22 May 2024

CITATION
Shukla A, Dadhich R, Singh R, Rayas A, Saidi P, Dasarathy G, Berisha V and Turaga P (2024) Orthogonality and graph divergence losses promote disentanglement in generative models. *Front. Comput. Sci.* 6:1274779. doi: 10.3389/fcomp.2024.1274779

Orthogonality and graph divergence losses promote disentanglement in generative models

Ankita Shukla[1]*, Rishi Dadhich[2], Rajhans Singh[2,3], Anirudh Rayas[3], Pouria Saidi[3], Gautam Dasarathy[3], Visar Berisha[3] and Pavan Turaga[2,3,4]

[1]Department of Computer Science and Engineering, University of Nevada, Reno, NV, United States, [2]Geometric Media Lab, Arizona State University, Tempe, AZ, United States, [3]School of Electrical, Computer, Energy Engineering, Arizona State University, Tempe, AZ, United States, [4]School of Arts, Media, and Engineering, Arizona State University, Tempe, AZ, United States

Over the last decade, deep generative models have evolved to generate realistic and sharp images. The success of these models is often attributed to an extremely large number of trainable parameters and an abundance of training data, with limited or no understanding of the underlying data manifold. In this article, we explore the possibility of learning a deep generative model that is structured to better capture the underlying manifold's geometry, to effectively improve image generation while providing implicit controlled generation by design. Our approach structures the latent space into multiple disjoint representations capturing different attribute manifolds. The global representations are guided by a disentangling loss for effective attribute representation learning and a differential manifold divergence loss to learn an effective implicit generative model. Experimental results on a 3D shapes dataset demonstrate the model's ability to disentangle attributes without direct supervision and its controllable generative capabilities. These findings underscore the potential of structuring deep generative models to enhance image generation and attribute control without direct supervision with ground truth attributes signaling progress toward more sophisticated deep generative models.

KEYWORDS

generative models, auto-encoders, graph divergence, manifolds, geometry

1 Introduction

Data-driven deep learning techniques have resulted in numerous advances, but several findings have demonstrated the brittleness of such models in different end tasks (Nguyen et al., 2015; Pontin, 2018). Many reasons have been hypothesized for such empirical behavior, chief among which is the realization that there is a need to leverage known physical laws such as the physics of image formation, the interaction of light with surfaces, and disentangling the effects of intrinsic object-related shape from photometric variation into deep learning frameworks (Bronstein et al., 2017). Several studies show that even simple unaccounted for shifts in data can lead to large losses in performance. Furthermore, from information theoretic perspectives (Achille and Soatto, 2018), the concepts of invariance and task performance are considered at odds with each other (e.g., discrimination). Information theoretic metrics for invariance seek to reduce the dimension of representations (Achille and Soatto, 2018), whereas metrics for a specific end-task such as classification seem to benefit from larger representation dimension. Thus, it seems that one cannot achieve true invariance while maintaining high end-task performance nor

can one achieve high end-task performance while guaranteeing invariance. Information theoretic analysis suggests the need for a middle ground, where a geometric treatment of feature spaces and loss functions can potentially allow deep representations to find practical tradeoffs between task performance and invariance. When applied to generative models, invariance often refers to achieving a clean disentanglement of control variables–that correspond specifically to physical factors, such as pose, lighting, and shape–where modifying the control variable can be done in isolation, without affecting other variables.

In vision literature, much prior knowledge exists about how light interacts with surface geometry and reflectance properties, the workings of projective geometry, and how temporal dynamics can be used to explain observed dynamic scenes. These physical laws and properties constrain the set of feasible or valid observations from image sensors. Images are often constrained to lie in low-dimensional subsets (of large Euclidean spaces), more formally referred to as image manifolds. While numerous efforts have sought to characterize these image manifolds, both empirically and theoretically, throughout the last two decades (Turaga and Srivastava, 2015; Shao et al., 2018), how they are integrated into controllable and disentangled generative architectures is still an open question. In prior study (Shukla et al., 2019), we have shown that several of these constraint sets can be relaxed to subspace-type or sphere-type constraints. Different such constraints can be accommodated by constraining the features from the latent space to have orthogonality properties, as a proxy for physical factor disentanglement.

In the context of generative models, while there exist many different classes of architectures, a common theme is to learn the underlying distribution of the dataset. Once trained, it can be used to sample novel data points from the underlying distribution. There are diverse types of generative models, including but not limited to generative adversarial networks (GANs; Goodfellow et al., 2014) and variational autoencoders (VAEs; Kingma and Welling, 2013), each with pros and cons. These generative models generally have a low-dimensional latent space modeled using a Gaussian or uniform distribution, and they map these low-dimensional points to complex high-dimensional data points, matching the distribution of the training dataset. Generative models are versatile and used in various applications such as text-to-image models (Zhang et al., 2023), image-to-image translation (Zhu et al., 2017), domain adaptation (Hoffman et al., 2018), image editing (Zhu et al., 2020), and inverse problems (Asim et al., 2020). Furthermore, these generative models can offer explainable and controllable representation, leading to disentangled representations, a key area of interest. How do we encode the geometric nature of the output space in the loss function of generative model is still an open question. In this article, we make some concerted advances toward that via manifold-divergence loss functions and latent-space orthogonality properties.

One major challenge in existing approaches is the trade off between their ability to disentangle different attributes and their ability to generate novel samples. Most existing studies are based on VAEs and GANs that encourage factorization of the latent space. Methods based on VAE add a regularizer to the loss function to encourage disentanglement in the encoder distribution. Owing to their ability to capture explainable and controllable representation, they have been used in applications in computer vision, recommender systems, graph learning (Ma et al., 2019; Wang et al., 2020), and various downstream tasks. VAEs are auto-encoder models that map low-dimensional representation to images with a goal of image reconstruction. As, the network only focuses on reconstruction and mapping to it to Gaussian prior, it has weak disentanglement. Thereafter, β-VAE proposed to add a hyperparameter β that provides a tradeoff between regularization and reconstruction. However, the generated images have low reconstruction quality. With the success of geometric constraints in different learning scenarios, they have been recently explored in the context of unsupervised disentangled representation learning.

In this study, we draw from several prior threads of studies, several of which we have pursued, including orthogonality constraints on latent spaces, chart-autoencoder-inspired architectures, and graph divergence measures as differentiable loss functions. We develop a controllable generative architecture that integrates the following components: (a) a generative architecture motivated by chart-autoencoders to promote separation of latent space in a set of disjoint latent spaces, (b) an orthonormality constraint across latent spaces implemented as a proxy for statistical independence to promote effective disentanglement, (c) a differentiable graph theoretic divergence measure that serves as an approximation to manifold-to-manifold divergence, as a measure of discrepancy between the training-set and the generated set. The contributions of this article are as follows:

- We propose a set of simple yet effective loss functions for disentangled representation learning that combine the benefits of orthogonality constraints in the latent space to promote factor disentanglement, with a differentiable graph divergence loss on the output to promote a manifold structure in the output space.
- We develop an architecture that consists of encoding latent spaces as attribute spaces that can be trained with the aforementioned loss functions. This has the advantage of providing image manipulation controls by navigating individual attribute spaces.
- We show experimental results on the challenging 3D shapes datasets, showing disentanglement of several meaningful attributes, and their potential in generative modeling tasks.

2 Background

2.1 Notation

We use lowercase letters for scalars, bold lowercase letters for vectors, and bold uppercase letters for matrices. We use $\mathcal{G}(\mathbf{X}) = (\mathbf{X}, E)$ to define the complete directed graph over the vertex set $\mathbf{X}$ with edges E, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is a set of points in $\mathbb{R}^d$. For any edge, $e \in E$ with adjacent vertices i and j, we denote the weight of the edge by $d(e) = d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$, where $\|.\|$ denotes the Euclidean norm. Following the notation from Djolonga and Krause (2017), we assume there is a function π that assigns a label to each vertex, i.e., $\pi : X \to \{1, 2\}$. We use $\mathbf{x} \sim p$ to indicate that a random vector $\mathbf{x}$ is drawn from a distribution p.

2.2 Orthonormality in disentangled latent spaces

Orthogonality in latent spaces is motivated as a proxy for physical independence of variables. Specifically, our high-level approach has been to promote the learning of disentangled representations to account for physical variables such as rotation, illumination, and shapes as elements of groups such as the special orthogonal group, and Grassmannians.

Meanwhile, the illumination cone model is a pivotal concept in computer vision, especially for tasks like facial recognition. This model conceptualizes all potential images of an object under varying lighting conditions as existing within a high-dimensional space. Assuming Lambertian reflectance, and convex object-shapes, one can show that the image space is a convex-cone in image space (Georghiades et al., 2001). A relaxation of this model leads to identifying cones as linear subspaces, which are seen as points on a Grassmannian manifold $\mathcal{G}_{n,k}$ (n = image-size, k = lighting dimensions, typically considered equal to number of linearly independent normals on the object shape). Under certain conditions of variance on the Grassmannian being low, a distribution of points on the Grassmannian induces a distribution on a high-dimensional sphere, whose dimension depends on n and k (Chakraborty and Vemuri, 2015), which we have leveraged in prior study to impose Grassmannian constraints in latent spaces (Lohit and Turaga, 2017). Similarly, 3D pose is frequently represented as an element of the special orthogonal group $SO(3)$. For analytical purposes, it is convenient to think of rotations represented by quaternions, which are elements of the 3-sphere S^3 embedded in $\mathbb{R}^4$, with the additional constraint of antipodal equivalence. This makes rotations to be identified as points on a real-projective space $\mathbb{RP}^3$. Real-projective spaces are just a special case of the Grassmannian–in this case, of 1D subspaces in $\mathbb{R}^4$. From a distribution on quaternions, we can induce a distribution in a higher dimensional hyperspherical manifold.

Our previous approaches indicate that imposing these product-of-sphere constraints via a simple orthonormality condition improves model explainability, reduces calibration error, and provides robustness to a variety of image degradation and feature-pruning conditions (Choi et al., 2020). In this study, we show that it improves the learning of disentangled representations as well.

Let $\mathbf{z}_k$ represents the latent space representation corresponding to the k^{th} attribute. Now, for a disentangling network, the combined orthogonal loss function is given by (1)

$$\begin{aligned} &\text{argmin}_{\theta,\phi} \quad \mathcal{L}_{dis}(\theta,\phi) + \|\mathbf{Z}\mathbf{Z}^\top - \mathbf{I}\|_2 \\ &\text{where,} \quad \mathbf{Z} = [\mathbf{z}^1, \mathbf{z}^2, \ldots \mathbf{z}^k], \quad \text{with} \\ &\mathbf{z}^k \text{subpart of the latent embedding:} \quad \mathbf{z} = e(\mathbf{X}). \end{aligned} \tag{1}$$

2.3 Graph test statistics

In the context of controllable generative models, the main goal is to train a generative model capable of transforming latent space representations to samples generated by an unknown target distribution. To ensure that the generated samples are drawn according to a desired target distribution, it becomes essential to measure the "closeness" between the latent space distribution and the unknown target distribution. As traditional divergence measures require knowledge of the underlying distributions, they are not suitable for this task. In this article, we consider the k-NN test statistic, a multivariate graph test statistics for computational efficiency (Djolonga and Krause, 2017) that exhibit the desired property of being distribution-free while acting as a good surrogate for the divergence measure between distributions. Given that the k-NN test statistic is inherently non-differentiable, a smoothing process is introduced to approximate it with continuously differentiable functions.

Let us consider the latent space representations, denoted by $z \sim Q_0$, generated according to a distribution Q_0. The generative model then produces samples $f_\theta(z) \sim Q$, where f_θ is a differentiable function parameterized by θ. The primary objective is to optimize θ to produce samples that closely resemble the unknown target distribution P. We now outline the procedure we employ to compute this statistic as mentioned in Djolonga and Krause (2017). First, we gather data samples from two distributions, denoted as $X_0 \sim P$ and $X_1 \sim Q$, which is aggregated to form a joint dataset $\mathbf{X} = X_0 \cup X_1$. Then, we construct a complete graph $\mathcal{G}(\mathbf{X})$ on $\mathbf{X}$, a k-NN neighborhood denoted by $\mathcal{U}^*$ is constructed by connecting each point $x \in \mathbf{X}$ to its k-nearest neighbors (in Euclidean distance). In order to distinguish between the two sets of data, we define a group membership function, represented as a map $\pi^* : \mathbf{X} \to \{0, 1\}$, which assigns the value 0 to elements in X_0 and the value 1 to elements in X_1. Finally, the k-NN test statistic, denoted as T_{π^*}, is computed by evaluating the number of edges in $\mathcal{U}^*$ connecting points in X_0 to points in X_1, more formally for every edge $e \in \mathcal{U}^*$ with adjacent vectors i and j, we denote by $\mathbb{I}_{\pi^*}(e)$ to mean $\mathbb{I}\{\pi^*(i) \neq \pi^*(j)\}$, where $\mathbb{I}$ is the indicator function. The k-NN test statistic is then given by $T_{\pi^*}(\mathcal{U}^*) = \sum_{e \in \mathcal{U}^*} \mathbb{I}_{\pi^*}(e)$. Under the null hypothesis where the two distributions are equal, it results in a larger test statistic.

As our objective was to design a generative model capable of producing according to a target distribution P, we seek to identify the optimal parameter θ by maximizing the expected test statistic $\mathbb{E}_{X_0 \sim P, Z \sim Q_0}[T_{\pi^*}(X_0, f_\theta(Z))]$. However, as T_{π^*} is not differentiable, we use the differentiable k-NN test (Djolonga and Krause, 2017) by relaxing it to expectations in natural probabilistic models by designing a probability distribution over a subset of the edges $\mathcal{U}$ to focus on feasible configurations (Djolonga and Krause, 2017). To this end, the neighborhood $\mathcal{U}$ is drawn according to the Gibbs measure with the temperature parameter λ. Subsequently, the graph test statistic can be replaced by its expectation giving rise to the smoothed statistic (Djolonga and Krause, 2017):

$$T_{\pi^\star}\left(\mathcal{U}^\star\right) \to T^\lambda_{\pi^\star} := \mathbb{E}_{\mathcal{U} \sim P(.|d,\lambda)}\left[T_{\pi^\star}(\mathcal{U})\right] = \sum_{e \in \mathcal{U}} \mathbb{I}_{\pi^\star}(e) \mu\left(d/\lambda\right)_e , \tag{2}$$

where $\mu\left(d/\lambda\right)_e$ denotes the marginal probability of the edge e. In this study, we employ the smoothed k-NN test with $k = 1$, as it provides the most computationally efficient differentiable test. In this case, it has been shown that the smoothed graph test in Equation (2) redcues to the following

$$T^\lambda_{\pi^\star}(X_0, X_1) = \sum_{i,j} \mathbb{I}\{\pi^*(i) \neq \pi^*(j)\} \frac{e^{-\|x_i - x_j\|/\lambda}}{\sum_{k \neq i} e^{-\|x_i - x_k\|/\lambda}} . \tag{3}$$

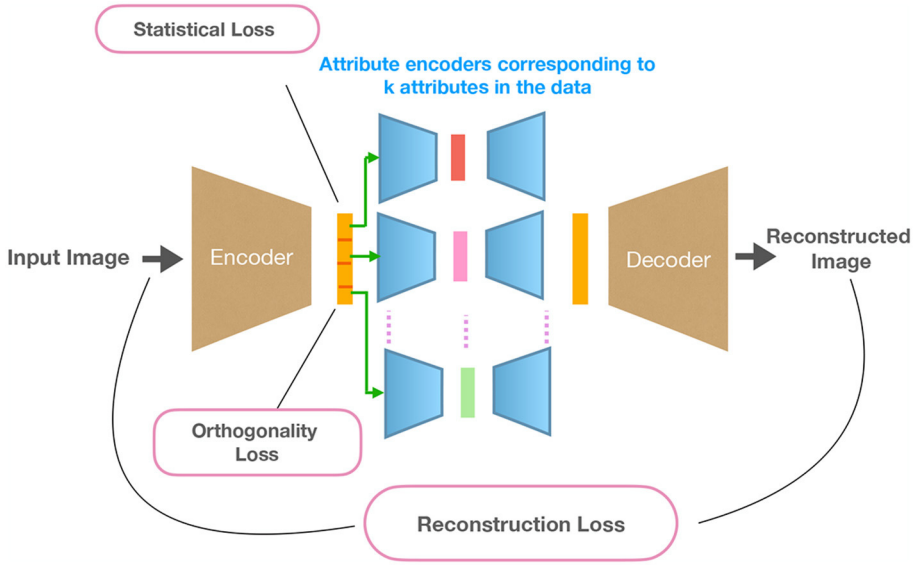


FIGURE 1
Overview of the proposed approach DAE-OG. Images are transformed with an encoder to a low-dimensional latent space. This representation is partitioned into *k* disjoint subparts corresponding to *k* attributes and transformed by *k* attribute auto-encoders. This is followed by a decoder that reconstructs the image from the concatenated attribute auto-encoder representations.

Although this test defined by Equation (3) can be used directly, lower values of $T^{\lambda}_{\pi^\star}$ does not guarantee lower *p*-values; toward this, Djolonga and Krause (2017) proposed an alternative test statistic based on the notion of a smooth *p*-value defined as

$$t^{\lambda}_{\pi^\star} = \frac{T^{\lambda}_{\pi^\star} - \mathbb{E}_{\pi\sim H_0}\left[T^{\lambda}_{\pi}\right]}{\sqrt{\mathbb{V}_{\pi\sim H_0}\left[T^{\lambda}_{\pi}\right]}}. \tag{4}$$

We use this notion of the *t*-statistic in Equation (4) to define our graph divergence loss as follows,

$$\mathcal{L}^{\lambda}_{\text{stat}}(X_0, X_1) = -t^{\lambda}_{\pi^\star}(X_0, X_1) = -\frac{T^{\lambda}_{\pi^\star} - \mathbb{E}_{\pi\sim H_0}\left[T^{\lambda}_{\pi}\right]}{\sqrt{\mathbb{V}_{\pi\sim H_0}\left[T^{\lambda}_{\pi}\right]}}. \tag{5}$$

3 Proposed framework

In this section, we provide details of the proposed disentangled generative model. An overview of the framework is shown in Figure 1. We use autoencoder as the backbone of our model and improve disentangling performance and reconstruction quality through proposed constraints and divergence loss. Specifically, we introduce an orthogonality loss to promote disentangled representation and a manifold divergence loss to learn the underlying data distribution. These losses improve the disentanglement and generative performance of the model, discussed in detail in the following sections.

Our approach first embeds the training images into a low-dimensional representation followed by mapping disjoint parts of these representations to low-dimensional latent vectors with attribute auto-encoders, aiming to encode different attributes in the data. This is followed by a decoder/generator network that transforms representations from attribute auto-encoders to an image.

3.1 Encoder network

Specifically, as the first step, images are transformed by the encoder $e(\theta)$ to the latent representation $\mathbf{z}_e \in \mathbb{R}^{n_{out}}$. The latent representation is partitioned into k equally sized subsets, where k denotes the number of distinct attributes in the data. The partitioned representations are transformed using k different attribute auto-encoders.

3.2 Relevance of attribute auto-encoders

Specifically, we utilize attribute encoder networks that transform a disjoint subset of latent representations further into a lower dimension space. This allows the network to learn, from disjoint latent dimensions, relevant and informative factors of variations that control different aspects of the image. The choice of the number of such attribute auto-encoders is empirically based on observed factors of variations in the data. This is also based on the assumption that the images are created by different factors that can vary independently of each other, which is often the case in practical situations. Each of the attribute auto-encoder transforms the subset of latent representation to an integral latent representation of dimension n_k for the k^{th} attribute. We choose n_k as the number of unique variations in an attribute.

3.3 Decoder network

The decoder network is responsible for reconstructing an image of the same size as the input using the embeddings from the attribute encoders. The embeddings are stacked in-order and fed to the decoder network for reconstruction. For the sake of simplicity,

we refer to the combined parameters of attribute encoders and the decoder network as ϕ and the mapping function as g.

3.4 Loss Functions

We now present details of our loss functions that enable disentangled representation learning in an auto-encoder framework. Our loss function consists of three parts that are discussed below:

3.4.1 Reconstruction loss

Given an autoencoder, reconstruction loss measures the ability of the network to reconstruct an image from the input image when transformed into a low-dimensional space.

3.4.2 Enforcing orthogonality on latent embeddings

We impose the orthogonality constraint on the initial latent space. To achieve disentangled representation, we enforce orthogonality constraint on representations of every image. The encoder transforms the image to n_{out} dimensions. In order to ensure stability during network training, the subset dimensions are normalized to unity, as given by Equation (1).

3.4.3 Enforcing $\mathcal{L}_{stat}$ on latent embeddings

Given an unknown target distribution, the main objective was to learn an implicit generative model where one can sample from without the ability to evaluate the distribution. This can be achieved by minimizing a divergence loss that measures the difference between the target distribution and a transformation on the latent space and can be captured by enforcing the $\mathcal{L}_{stat}$ on the initial latent embeddings, as given by Equation (5).

The total loss function is given as follows:

$$\begin{aligned}\mathcal{L}(\theta,\phi) =& \text{argmin}_{\theta,\phi} \quad \|\mathbf{X}-\mathbf{X}_{recon}\|_2+\alpha_1\|\mathbf{Z}\mathbf{Z}^\top-\mathbf{I}\|_2 \\ &+\alpha_2\mathcal{L}_{stat}(\mathbf{X},g(\mathcal{N})) \qquad (6)\\ \text{where,}\quad \mathbf{Z}\in\mathbb{R}^{n_{out}} &= [\hat{\mathbf{z}}^1,\hat{\mathbf{z}}^2,\ldots\hat{\mathbf{z}}^k],\quad \mathbf{X}_{recon}=g(e(\mathbf{X},\theta),\phi)\\ \text{and}\quad \hat{\mathbf{z}}^i &= \mathbf{z}^i/\|\mathbf{z}^i\|_2 \quad \text{for} \quad i=1,2,..k\end{aligned}$$

here $\mathbf{z}^k$ represents the latent space representation corresponding to the k^{th} encoder and α_1 and α_2 are hyperparameters for the weights corresponding to the two loss functions. Here, we use θ to denote the encoder e parameters, and ϕ denotes the combined parameters of the attribute auto-encoders and the decoder for simplicity.

3.4.4 Generation

In order to generate new samples, we sample from a Gaussian prior with zero mean and unit standard deviation. The normal distribution is defined in the n_{out} dimensional space.

4 Experimental setup and results

In this section, we present details about the experiments to evaluate the effectiveness of our approach. Our approach is termed **D**isentangled **A**ttribute **E**ncoder with **O**rthogonality and **G**raph divergence—**DAE-OG**. We compare our approach **DAE-OG** with the model without orthogonality constraint, referred to as **DAE-G**.

4.1 Setup

4.1.1 Dataset description

Our approach is evaluated on the 3D shapes dataset Burgess and Kim (2018). This dataset consists of 3D shapes, procedurally generated from six ground truth independent latent factors. These factors are floor color, wall color, object color, scale, shape, and orientation. For our experiments, we re-sample the dataset to have five variations for two different objects with fixed floor hue.

4.1.2 Model

We implement the initial encoder with convolution layers followed by a fully connected layer. Each of the attribute encoders consists of FC and ELU layers. The decoder is constructed in the same way as the encoder. For our experiments, we report results with three and five attribute auto-encoders. We use the same network architecture across all our experiments–an overview of the architecture is shown in Table 1.

4.1.3 Training details

We adopt an annealing strategy for network training with the loss function given in Equation (6). The models are trained for 1,000 epochs, with an initial learning rate of $3e-4$. The value of hyperparameters α_1 and α_2 are selected as $\alpha_1=\alpha_2=0.1$ and $\alpha_1=\alpha_2=0.001$ for 3 and 5 attribute spaces, respectively. We set n_{out} to 96 and 105 for 3 and 5 attribute encoders, respectively. In case of 3 attribute encoder network, the n_k corresponding to three attribute encoders are 6, 9, and 5. In addition, in case of 5 attribute encoder network, the n_k corresponding to the five attribute encoders are 15, 8 10, 10, and 2. For the graph divergence loss, we use $k=1$ in k-nn test and $\lambda=0.1$ for all our experiments.

4.2 Results

We compare the advantages of our model from both qualitative and quantitative aspects, across many criteria including reconstruction error, image quality, disentanglement measures, and FID scores.

4.2.1 Reconstruction fidelity

We first evaluate the reconstruction fidelity of the model both quantitatively and qualitatively. Few example images as well as corresponding reconstructed images are shown in Figure 2 for 3 and 5 partitions of the latent space. We also report MSE in

TABLE 1 Details of the network architectures used to design the generative model used across different experiments.

Encoder: images →	Conv(3, 16, 4, 4) + ELU	**Decoder:**	FC (n_{out}, 128) + ELU
	Conv(16, 32, 2, 2) + ELU		FC (128, 2048) + ELU
	FC (2048, 128) + ELU		DConv(32,16,2,2) + ELU
	FC (2048, n_{out}),		DConv(16, 3, 4, 4) + Sigmoid() → **Images**
Attribute auto-encoders			
Encoder	FC (n_{in},128) + ELU	**Decoder**	FC (k_n,32) + ELU
	FC (128,64) + ELU		FC (32, 64) + ELU
	FC (64, 32) + ELU		FC (64,128) + ELU
	FC (32,n_k)		FC (128, n_{out})

Here, n_k denotes the internal dimension of the attribute autoencoder, chosen to be the number of unique labels in an attribute.

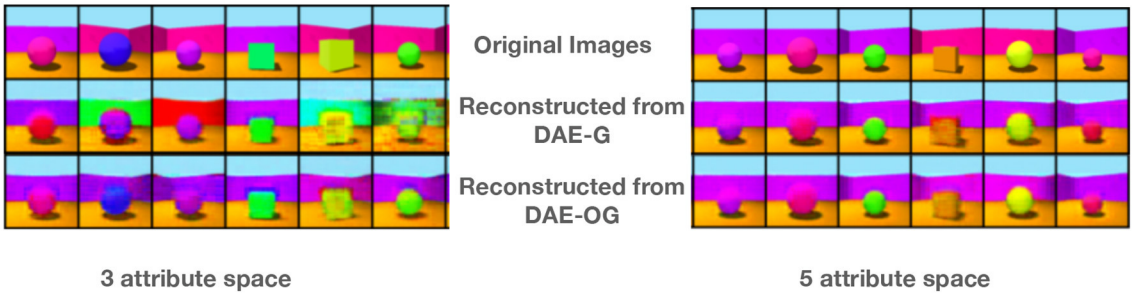


FIGURE 2
Some example images and the corresponding reconstructions from our proposed model DAE-OG and its improvement over DAE-G for three and five attribute spaces. DAE-OG results in better visual quality.

TABLE 2 Comparison of MSE, PSNR, and FID of DAE-G and DAE-OG models for three and five attribute encoder models.

Method	Three attribute encoder			Five attribute encoder		
	MSE ↓	PSNR ↑	FID ↓	MSE ↓	PSNR ↑	FID ↓
DAE-G	0.054	14.25	154.53	0.016	19.48	146.49
DAE-OG	**0.019**	**17.84**	**153.04**	**0.014**	**19.97**	**145.80**

Bold means better performance.

Table 2. We observe that our approach consistently results in better reconstruction quality both quantitatively and qualitatively.

4.2.2 Image generation

Images are generated by sampling from a normal distribution in the initial encoder latent space. Example images are shown in Figure 3 for 3 and 5 attribute encoders. We observe that DAE-OG generates images that are visibly consistently better than DAE-G.

4.2.3 Latent space interpolation

Latent space manipulation is important for assessing the performance of disentangling abilities of models in terms of capturing independent and semantically meaningful factors of variations. We show results by interpolating between two images in the latent space of the initial encoder. The results show that traversal in the latent space leads to smoother interpolation in the image space as well. Specifically, owing to the orthogonality constraint, we obtain better results as shown in the Figures 4–7.

4.2.4 Disentanglement and FID scores

A large number of disentanglement scores have been proposed over the last several years that measure different aspects of disentanglement. We use a few of them in this study to evaluate the quality of disentanglement achieved owing to the contribution of orthogonality constraint. The results are shown in Table 3. We observe that the results with orthogonality constraints are consistently better than their counterpart.

4.2.5 Effect of orthogonality

As with the method DAE-OG, we observe smoother transition within an attribute space. We note that imposing the orthogonality loss term promotes disentanglement as seen in the low-dimensional

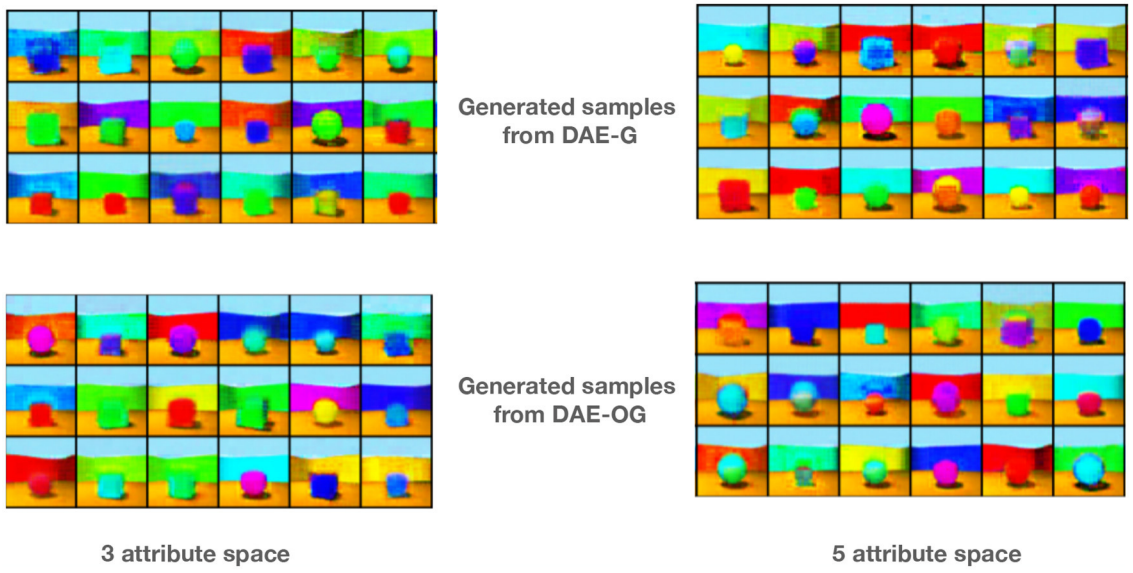


FIGURE 3
Some example images generated from our proposed model with three and five attribute auto-encoders by using implicit sampling for DAE-G and DAE-OG models.

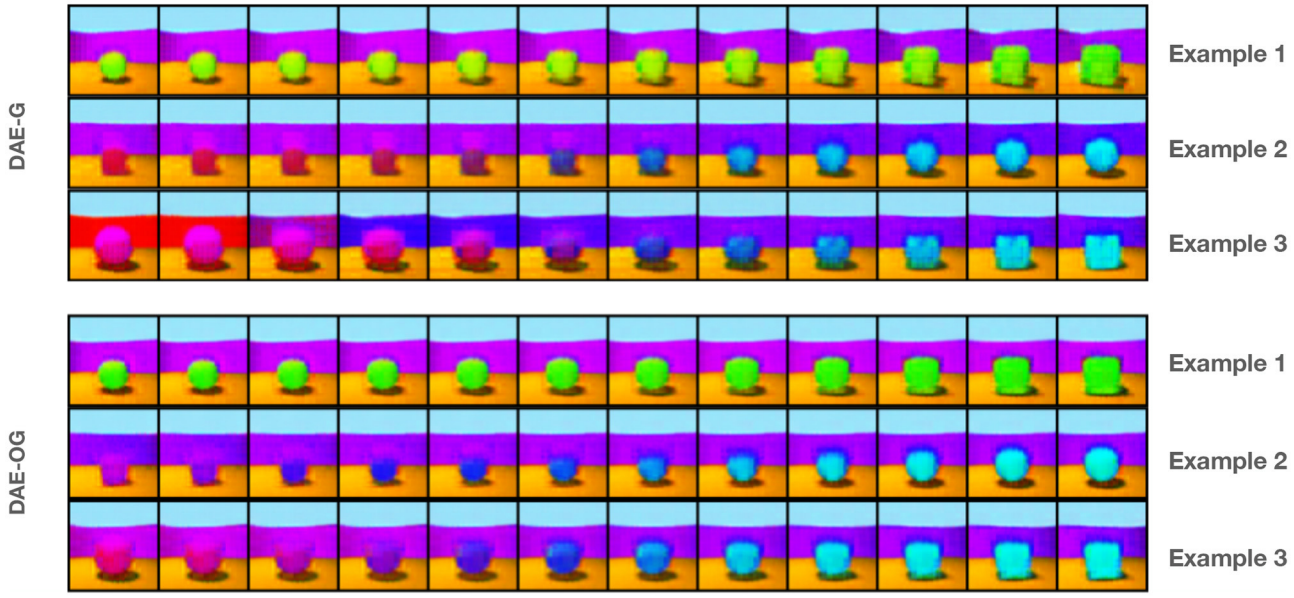


FIGURE 4
Three example interpolation results in the latent space of three attribute encoders on DAE-G and DAE-OG (bottom row). DAE-OG achieves a smoother interpolation.

visualization of the latent space in Figure 8, done via t-SNE (van der Maaten and Hinton, 2008).

5 Discussion

The experimental datasets chosen here in our study use the 3D shapes dataset, which has simple objects and scenes; still has sufficient complexity owing to shape change, view change, and background changes. We do observe meaningful disentanglement of variables in this case. We do anticipate that scaling to more complex datasets is feasible and could form directions for future study. Due to the special nature of disentanglement tasks, where one needs to further provide some notion of meaning to variable disentangled, common datasets used in literature to assess disentangling models usually include datasets, which show some natural transitions. These include (a) KITTI-masks–which contain binary masks of pedestrians (Klindt et al., 2021), (b) the Natural

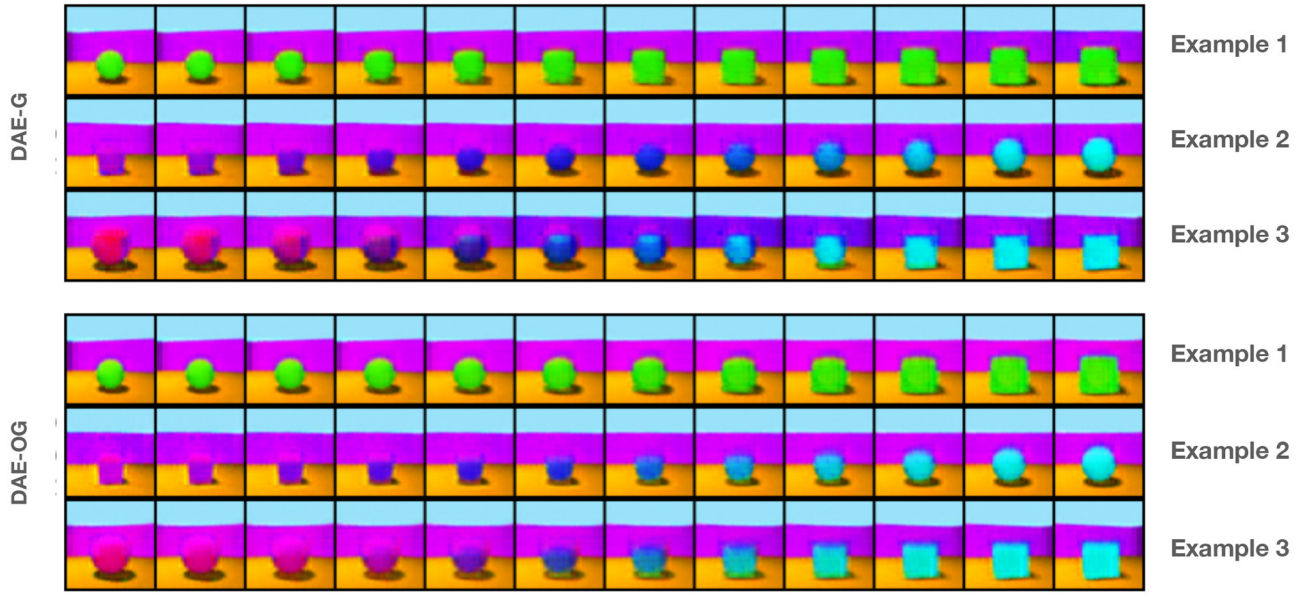


FIGURE 5
Three example interpolation results in the latent space of five attribute encoder. The first and the last image in each row denotes the start and end of the interpolation.

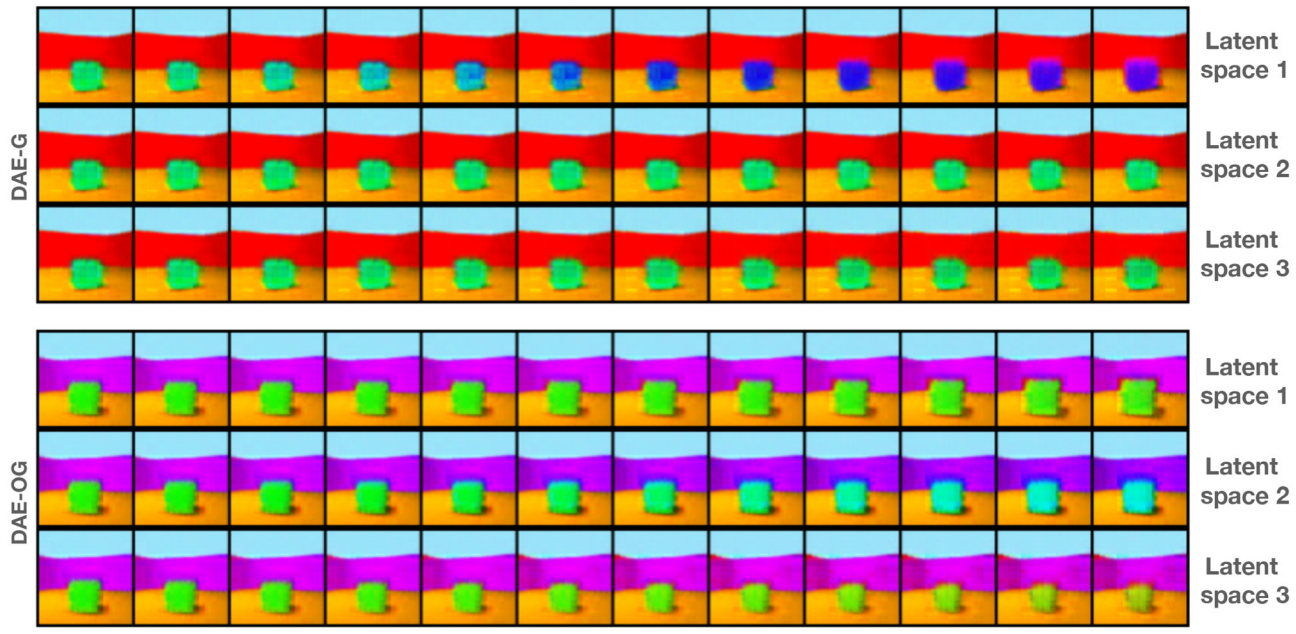


FIGURE 6
Interpolation along individual attribute space by keeping the other two attribute spaces frozen for a three attribute encoder-based model. For DAE-OG, in 1st attribute space, we see variation in the size; in 2nd attribute space, we see variation in the color of the object; and in 3rd attribute space, we see variation in shape.

Sprites dataset (Matthey et al., 2017), which consists of pairs of rendered sprite images with generative factors from the YouTube-VIS challenge, which we have experimented with before (Shukla et al., 2019), and (c) the 3DIdent dataset (Zimmermann et al., 2021), which contains objects rendered in 3D under differing lighting and viewing conditions.

The dataset we have chosen (Kim and Mnih, 2018) is another standard benchmark in this area, and is most similar to 3DIdent, however with simpler objects. With more complex objects as in 3DIdent, some of the observed visual variation will be due to self-shadowing and cast-shadowing, which would be an interesting avenue to explore the impact on disentanglement performance.

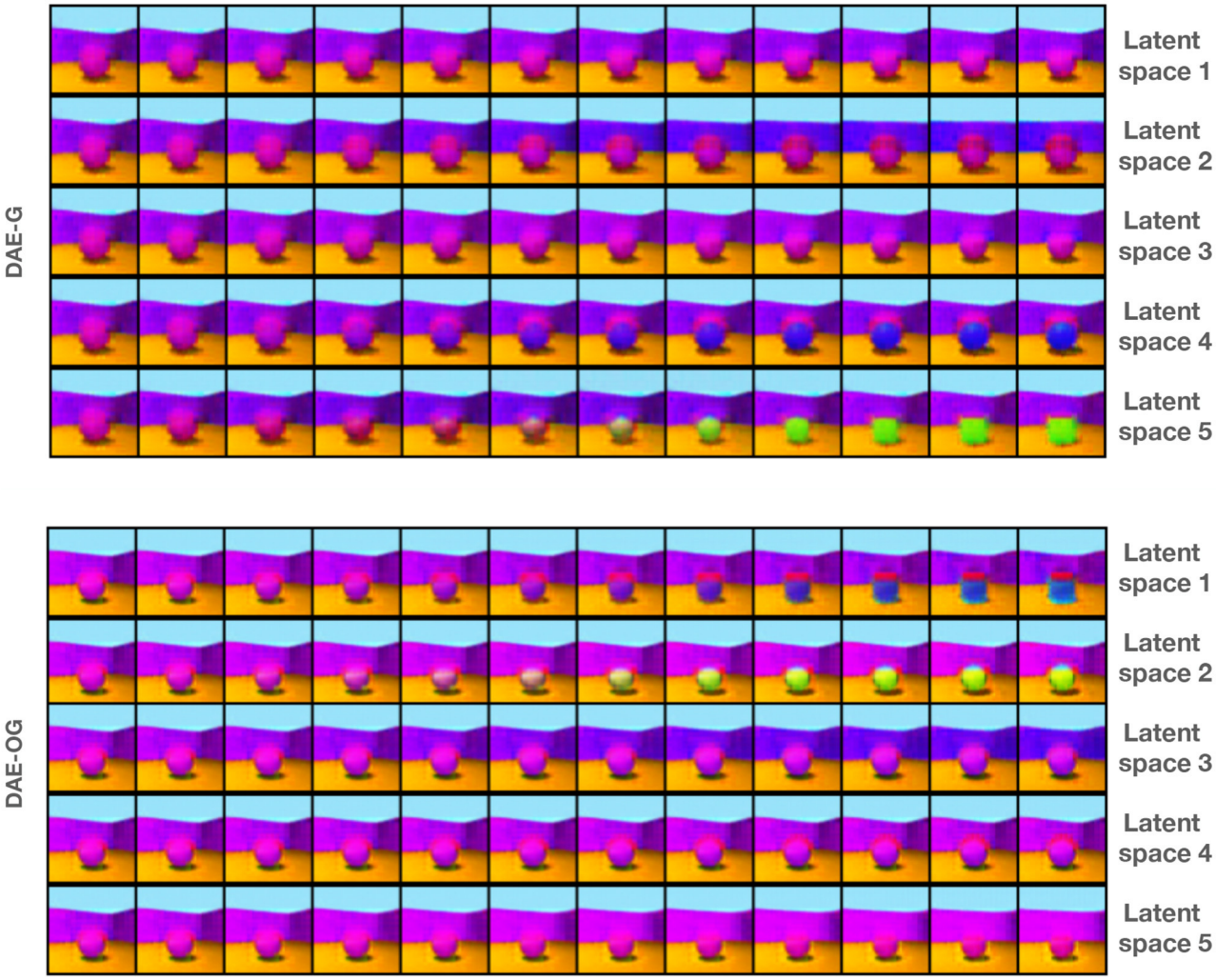


FIGURE 7
Interpolation along individual attribute space by keeping the other two attribute representations frozen for five attribute encoder-based model for DAE-G and DAE-OG. DAE-OG results in smoother interpolation in each space, with only one attribute change in a given space.

TABLE 3 **Disentanglement metric score.**

Method	Three attribute encoder				Five attribute encoder			
	mig	dcimig	mig_sup	jemmig	mig	dcimig	mig_sup	jemmig
DAE-G	0.0234	0.2832	0.1709	0.2271	0.0333	0.3172	0.1948	0.2273
DAE-OG	0.02760	0.2580	0.1589	0.2083	0.0278	0.3366	0.1885	0.2248

DAE-OG consistently outperforms DAE-G across the various disentanglement metrics.

6 Conclusion

In this article, we presented an approach to learning disentangled representations in a generative framework. In addition to disentanglement, our approach enables diverse image generation and manipulation. We find that orthogonality in the latent space encourages disentanglement with a graph divergence loss that transforms the latent space. Our results support the hypothesis that inductive biases are crucial for learning disentangled representations. In future, we would like to explore the possibility of incorporating known attribute-specific constraints to further improve the interpretability of the disentangled representations.

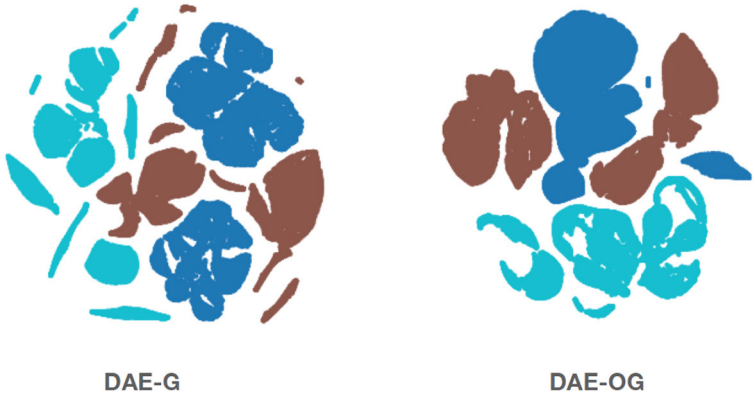


FIGURE 8
Two-dimensional visualization of the latent space representations showing the effect of orthogonality on the latent space representation for three attribute latent spaces. Attributes are color-coded. We note that DAE-OG achieves more compact and smoother transitions within an attribute space.

Author's note

This study was carried out when AS was at Geometric Media Lab, Arizona State University.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: https://github.com/deepmind/3d-shapes.

Author contributions

AS: Conceptualization, Investigation, Methodology, Writing - original draft, Writing - review & editing. RD: Investigation, Writing - original draft. RS: Investigation, Writing - original draft. AR: Writing - original draft, Writing - review & editing. PS: Investigation, Writing - original draft, Writing - review & editing. GD: Writing - original draft, Writing - review & editing. VB: Investigation, Writing - original draft, Writing - review & editing. PT: Conceptualization, Investigation, Writing - original draft, Writing - review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This study was supported by Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112290073.

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.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Achille, A., and Soatto, S. (2018). Emergence of invariance and disentanglement in deep representations. *J. Machine Learn. Res.* 19, 1947–1980. doi: 10.1109/ITA.2018.8503149

Asim, M., Daniels, M., Leong, O., Ahmed, A., and Hand, P. (2020). "Invertible generative models for inverse problems: mitigating representation error and dataset bias," in *International Conference on Machine Learning* (PMLR), 399–409.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Sign. Process. Mag.* 34, 18–42. doi: 10.1109/MSP.2017.2693418

Burgess, C., and Kim, H. (2018). 3D Shapes Dataset. Available online at: https://github.com/deepmind/3dshapes-dataset/

Chakraborty, R., and Vemuri, B. C. (2015). "Recursive fréchet mean computation on the grassmannian and its applications to computer vision," in *IEEE International Conference on Computer Vision, ICCV 2015*, 4229–4237.

Choi, H., Som, A., and Turaga, P. K. (2020). Role of orthogonality constraints in improving properties of deep networks for image classification. *CoRR* abs/2009.10762. Available online at: https://arxiv.org/abs/2009.10762

Djolonga, J., and Krause, A. (2017). "Learning implicit generative models using differentiable graph tests," in *Advances in Approximate Bayesian Inference NIPS Workshop*. Available online at: http://www.approximateinference.org/2017/accepted/DjolongaKrause2017.pdf

Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 643–660. doi: 10.1109/34.927464

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, eds. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, 2672–2680.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., et al. (2018). "CYCADA: cycle-consistent adversarial domain adaptation," in *International Conference on Machine Learning* (PMLR), 1989–1998.

Kim, H., and Mnih, A. (2018). "Disentangling by factorising," *Proceedings of the 35th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research*, eds. J. Dy and A. Krause, vol. 80 (PMLR), 2649–2658. Available online at: https://proceedings.mlr.press/v80/kim18b.html

Kingma, D. P., and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. doi: 10.48550/arXiv.1312.6114

Klindt, D. A., Schott, L., Sharma, Y., Ustyuzhaninov, I., Brendel, W., Bethge, M., et al. (2021). "Towards nonlinear disentanglement in natural data with temporal sparse coding," *International Conference on Learning Representations*. Available online at: https://openreview.net/forum?id=EbIDjBynYJ8

Lohit, S., and Turaga, P. K. (2017). "Learning invariant riemannian geometric representations using deep nets," in *IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017* (IEEE Computer Society), 1329–1338.

Ma, J., Cui, P., Kuang, K., Wang, X., and Zhu, W. (2019). "Disentangled graph convolutional networks," in *International Conference on Machine Learning* (PMLR), 4212–4221.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. (2017). *dsprites: Disentanglement Testing Sprites Dataset*. Available online at: https://github.com/deepmind/dsprites-dataset/

Nguyen, A. M., Yosinski, J., and Clune, J. (2015). "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 427–436.

Pontin, J. (2018). *Greedy, Brittle, Opaque, and Shallow: The Downsides to Deep Learning*. Wired Magazine.

Shao, H., Kumar, A., and Fletcher, P. T. (2018). "The riemannian geometry of deep generative models," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW), 4288.

Shukla, A., Bhagat, S., Uppal, S., Anand, S., and Turaga, P. K. (2019). "PrOSe: product of orthogonal spheres parameterization for disentangled representation learning," in *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019* (BMVA Press), 88. Available online at: https://bmvc2019.org/wp-content/uploads/papers/1056-paper.pdf

Turaga, P. K., and Srivastava, A. (2015). *Riemannian Computing in Computer Vision, 1st Edn*. Berlin: Springer Publishing Company, Incorporated.

van der Maaten, L., and Hinton, G. (2008). Visualizing data using t-SNE. *J. Machine Learn. Res.* 9, 2579–2605. Available online at: http://jmlr.org/papers/v9/vandermaaten08a.html

Wang, X., Jin, H., Zhang, A., He, X., Xu, T., and Chua, T.-S. (2020). "Disentangled graph collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1001–1010.

Zhang, C., Zhang, C., Zhang, M., and Kweon, I. S. (2023). Text-to-image diffusion model in generative AI: a survey. *arXiv preprint arXiv:2303.07909*. doi: 10.48550/arXiv.2303.07909

Zhu, J., Shen, Y., Zhao, D., and Zhou, B. (2020). "In-domain gan inversion for real image editing," in *European Conference on Computer Vision* (Berlin: Springer), 592–608.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232.

Zimmermann, R. S., Sharma, Y., Schneider, S., Bethge, M., and Brendel, W. (2021). "Contrastive learning inverts the data generating process," in *Proceedings of the 38th International Conference on Machine Learning*, eds. M. Meila and T. Zhang (PMLR), 12979–12990.

frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 26 August 2024
DOI 10.3389/fcomp.2024.1260604

OPEN ACCESS

EDITED BY
Maria Rodriguez Martinez,
Yale University, United States

REVIEWED BY
Zhanglin Cheng,
Chinese Academy of Sciences (CAS), China
Ankita Shukla,
University of Nevada, Reno, United States

*CORRESPONDENCE
Yinzhu Jin
✉ yj3cz@virginia.edu

RECEIVED 18 July 2023
ACCEPTED 12 July 2024
PUBLISHED 26 August 2024

CITATION
Jin Y, McDaniel R, Tatro NJ, Catanzaro MJ, Smith AD, Bendich P, Dwyer MB and Fletcher PT (2024) Implications of data topology for deep generative models. *Front. Comput. Sci.* 6:1260604. doi: 10.3389/fcomp.2024.1260604

Implications of data topology for deep generative models

Yinzhu Jin[1]*, Rory McDaniel[1], N. Joseph Tatro[2], Michael J. Catanzaro[3], Abraham D. Smith[3,4], Paul Bendich[3,5], Matthew B. Dwyer[1] and P. Thomas Fletcher[1,6]

[1]Department of Computer Science, University of Virginia, Charlottesville, VA, United States, [2]STR, Vision and Image Understanding Group, Woburn, MA, United States, [3]Geometric Data Analytics, Inc., Durham, NC, United States, [4]Math, Stats, and CS Department, University of Wisconsin-Stout, Menomonie, WI, United States, [5]Department of Mathematics, Duke University, Durham, NC, United States, [6]Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, United States

Many deep generative models, such as variational autoencoders (VAEs) and generative adversarial networks (GANs), learn an immersion mapping from a standard normal distribution in a low-dimensional latent space into a higher-dimensional data space. As such, these mappings are only capable of producing simple data topologies, i.e., those equivalent to an immersion of Euclidean space. In this work, we demonstrate the limitations of such latent space generative models when trained on data distributions with non-trivial topologies. We do this by training these models on synthetic image datasets with known topologies (spheres, torii, etc.). We then show how this results in failures of both data generation as well as data interpolation. Next, we compare this behavior to two classes of deep generative models that in principle allow for more complex data topologies. First, we look at chart autoencoders (CAEs), which construct a smooth data manifold from multiple latent space chart mappings. Second, we explore score-based models, e.g., denoising diffusion probabilistic models, which estimate gradients of the data distribution without resorting to an explicit mapping to a latent space. Our results show that these models do demonstrate improved ability over latent space models in modeling data distributions with complex topologies, however, challenges still remain.

KEYWORDS

data topology, generative model, variational autoencoder (VAE), diffusion probabilistic models (DDPM), topological data analysis

1 Introduction

Recent advances in deep generative models (DGMs) have resulted in the unprecedented ability of these models to produce realistic data, including imagery, text, and audio. While qualitative evaluation of generated data makes it clear that DGMs are improving at a rapid pace, quantifying how well a model produces samples similar to the original data distribution on which it was trained is a challenging task and an area of active research. Inherent to this problem is that generative models are fundamentally meant to produce data that would be judged to be realistic to a human observer, and quantifying human perception—of images, language, or audio—is a difficult task.

A common approach to evaluating a generative model is to compute an empirical distributional distance between a sample from the data distribution and a sample generated by the model. For example, in computer

vision, the Fréchet inception distance (FID) (Heusel et al., 2017) is a popular choice for such a distance metric. The FID approximates both the data distribution and the generated image distribution as multivariate normal distributions on the outputs of an Inception v3 model trained on ImageNet. The Fréchet distance between the resulting multivariate normal distributions is then computable in closed-form. More recently, precision and recall (Sajjadi et al., 2018) were proposed to separately evaluate how close generated samples are to the data distribution (precision) and how well they cover the data distribution (recall).

The manifold hypothesis of machine learning informally states that data distributions naturally lie near lower-dimensional manifolds embedded in the higher-dimensional Euclidean space formed by their raw representations. One class of DGMs, including variational autoencoders (VAEs) (Kingma and Welling, 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014), attempt to model the data manifold explicitly. They do this by generating data by mapping points from a prior distribution in a lower-dimensional latent space into the data representation space. This has led researchers to investigate the manifold properties of such DGMs and use manifold methods to evaluate their quality. Shao et al. (2018) develop algorithms for computing geodesic curves and parallel translation of VAEs. They observed that while VAEs were able to capture the curvature of synthetic data manifolds when trained on real image data, the manifolds generated by VAEs were nearly flat. Arvanitidis et al. (2018) propose that deterministic generators lead to a distortion of the data manifold in the latent space that fails to capture the intrinsic curvature of the data. They propose a stochastic Riemannian metric to correct for this and show that this results in improved variance estimates. Chen et al. (2018) demonstrate that Riemannian geodesics in the latent space of a DGM give better interpolations and visual inspection of generated data. Shukla et al. (2018) show that disentangled dimensions of the latent space of a VAE demonstrate higher curvature.

While these works have investigated the differential and metric geometry of DGMs, less is known about the topological properties of DGMs. Theoretically, models that generate data from a continuous mapping of a Gaussian prior distribution into Euclidean space, such as VAEs and GANs, are not able to faithfully reproduce data with non-trivial topology (e.g., spheres, tori, or other spaces with "holes"). In practice, these models may be able to perform fairly well in approximating non-trivial data topologies by shifting density away from holes. The chart autoencoder (CAE) model by Schonsheck et al. (2019) extends the topological abilities of VAEs/GANs by modeling a manifold topology with multiple overlapping charts. On the other hand, DDPMs and their relatives have no topological constraints in theory. However, the topological abilities of these various DGMs have not been empirically tested or compared. This paper empirically tests the ability of generative models to handle data arising from distributions with underlying topology, and is, to the best of our knowledge, the first systematic study in this direction. There have been papers that use topological techniques, such as Manifold Topology Divergence (Barannikov et al., 2021) or Geometry Score (Khrulkov and Oseledets, 2018), to quantify the quality of data produced by generative models. More broadly, there has been extensive recent work (Hensel et al., 2021) at the interface of TDA and DL/ML. These range from methods (e.g., Chen et al., 2019; Solomon et al., 2021; Nigmetov and Morozov, 2022) that integrate TDA-based loss functions into DL algorithms, to bespoke DNN architectures (Carrière et al., 2020) that incorporate layers that process persistence diagrams, to works (e.g., Naitzat et al., 2020; Wheeler et al., 2021) that use TDA to analyze the structure of data as it moves through DNN layers.

This paper is organized as follows. In Section 2 we review the methods used in this paper, namely, the DGMs and metrics for evaluating their quality, including persistent homology. In Section 3 we present our experiments comparing the ability of three DGMs—VAE, CAE, and DDPM—to learn to generate data with known non-trivial topologies. To do this, we use two synthetic image datasets with a torus and sphere topology, respectively, and a real dataset of conformations of cyclooctane, which is known to have topology equivalent to a Klein bottle intersecting with a 2-sphere (Martin et al., 2010). Note that this test is even more difficult from a topological perspective, as the cyclooctane conformations form a topology that is non-manifold, but rather a more complicated stratified space (in this case, the intersection of two manifolds). Finally, in Section 4 we discuss conclusions from these experiments and future directions.

2 Background and methods

In this section, we first review the three deep generative models (VAEs, CAEs, and DDPM) that we evaluated for their ability to learn data distributions with non-trivial topology. Next, we describe the evaluation metrics used for our study, both related and unrelated to the topological structure.

2.1 Deep generative models

Various structures for deep generative models have been proposed over time. Some of the popular models are normalizing flows (Rezende and Mohamed, 2015), variational autoencoders (Kingma and Welling, 2014), generative adversarial networks (Goodfellow et al., 2014), deep energy-based model (Du and Mordatch, 2019), and the recent denoising diffusion models (Ho et al., 2020). Each type of generator has different variations. Yet, topology is rarely considered in the design. Here we choose three models to discuss.

2.1.1 Variational autoencoders

A variational autoencoder (VAE) is a type of encoder-decoder generative model proposed by Kingma and Welling (2014). Unlike the traditional autoencoder (Hinton and Salakhutdinov, 2006), a VAE models the probability distribution of the latent representation, z, of each data point instead of a deterministic latent representation. A VAE models the marginal log-likelihood of the ith data point, $x^{(i)}$, as Equation 1:

$$\log p_\theta\left(x^{(i)}\right) = D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| \, p_\theta\left(z \mid x^{(i)}\right)\right) + \mathcal{L}\left(\theta, \phi; x^{(i)}\right), \tag{1}$$

where θ is a vector of the parameters for the generative model, and ϕ is a vector of the parameters for the variational approximation.

The objective is to maximize the evidence lower bound (ELBO), which is derived to be Equation 2:

$$\mathcal{L}\left(\theta, \phi; x^{(i)}\right) = -D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi}\left[\log p_\theta\left(x^{(i)} \mid z\right)\right], \quad (2)$$

Usually the prior $p_\theta(z)$ is set to be an isotropic Gaussian, $\mathcal{N}(\mathbf{0}, \mathrm{I})$. The encoder also models the $q_\phi(z \mid x^{(i)})$ as a Gaussian distribution $\mathcal{N}(\mu^{(i)}, (\sigma^{(i)})^2\mathrm{I})$. Therefore, the first term is easy to compute with predicted mean and variance of $q_\phi(z \mid x^{(i)})$. On the other hand, the equation for the second term of the lower bound depends on what probability distribution we assume in the data space. For example, using an isotropic Gaussian distribution leads to the mean squared error loss, and using a Bernoulli distribution corresponds to minimizing the binary cross entropy loss.

As discussed above, VAEs usually assume a Gaussian distribution in the latent space. Although this might be a reasonable assumption for many data with trivial topology, it might cause problems when this is not the case. Even in a simple case where the data has an $\mathbb{S}^1$ topology which is a loop, the neural network could struggle to learn a mapping from two different topological spaces. Although one might argue the Gaussian can be deformed enough so that it resembles a loop in practice, we still need experiments to investigate this issue. Similarly, generative adversarial networks (Goodfellow et al., 2014) also use a Gaussian prior distribution in the latent space, and therefore might as well have problems learning data with non-trivial topology.

2.1.2 Chart autoencoders

In many applications of the VAE, its learned latent space is often treated as a linear space. For instance, generating interpolations between two points of a given dataset is often performed by generating the linear path between the embeddings of these points in latent space. This operation implicitly assumes that the geodesics between points correspond to linear paths in latent space. Yet, we know there exist manifolds, such as the sphere S^2, which are not homeomorphic to a single linear space. It follows that the latent space learned by a VAE trained on such a manifold is not geometrically faithful. That is the latent space either contains a point that decodes to a point off the manifold, or the space cannot capture all geodesic paths. To this end, recent architectures have been introduced to rectify this problem. We consider one such architecture, the chart autoencoder (CAE) (Schonsheck et al., 2019).

Chart autoencoders are a generative model architecture motivated by the concept of an atlas in differential geometry. In comparison to the VAE, we learn a set of k encoders and decoders parameterized by $\{\phi_i\}_{i=1}^k$ and $\{\theta_i\}_{i=1}^k$ respectively. Each corresponding encoder and decoder is affiliated with a *latent chart*, Z_i. Thus, the latent space of the CAE is composed of a set of linear latent spaces. The CAE output is determined by a chart prediction network, P. In the original work, P maps x from the input space X to $p \in \mathbb{R}^k$, where p represents the vector of log probabilities of the chart membership of x. In training, the output of the CAE is taken to be the sum of the outputs from the k decoders weighted by the chart prediction vector, p. During evaluation, the output is taken to be that of the decoder corresponding to the likeliest chart via p. In this work, we update the chart prediction network to map from the direct sum of the latent embeddings, z_i, instead of x. This change was made to allow generations from the latent space without reference to any network input. Intuitively, this chart prediction network is analogous to the chart transition function affiliated with a geometric atlas. Indeed, the CAE is capable of transitioning between the outputs of different latent charts when a linear interpolation is performed in latent space.

2.1.3 Denoising diffusion probabilistic models

In contrast to the previous two models, the denoising diffusion probabilistic model (DDPM) proposed by Ho et al. (2020) does not have a low-dimensional latent space. It is based on the diffusion probabilistic model by Sohl-Dickstein et al. (2015), which learns to reverse a diffusion process in which Gaussian noise is gradually added to the original image, x_0, for T timesteps until we get a sampled image x_T that is nearly pure noise. We call the diffusion process that adds noise the forward process, which is a Markov chain. The reverse process is also defined to be a Markov chain as follows Equation 3:

$$\begin{aligned} p_\theta(x_{0:T}) &:= p(x_T) \textstyle\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \\ p_\theta(x_{t-1} \mid x_t) &:= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \end{aligned} \quad (3)$$

where β_t's define the variance schedule and θ is the parameter vector of the model that learns the reverse process. During training, we can optimize the lower bound of the log-likelihood.

In the DDPM, β_t is fixed and therefore the first term of the loss can be ignored. $\Sigma_\theta(x_t, t)$ is also fixed for each time step t. Then DDPM reparameterizes x_t with the added noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$, and $\mu_\theta(x_t, t)$ with $\epsilon_\theta(x_t)$, which means the model is now trained to predict the noise ϵ. Their experiments also show that omitting the different weights dependent on t does not compromise the final performance, which results in the final loss (Equation 4):

$$\mathcal{L}_{simple}(\theta) := \mathbb{E}_{t, x_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, t)\right\|^2\right], \quad (4)$$

with $\bar{\alpha}_t$'s being expressions of β_t's. It is also worth noting that Song et al. (2021) derived the same model from the view of a score based model, which learns the gradient of the log probability density in the data space.

We can see that the DDPM does not assume any topology on the original data distribution. The sampling only depends on the fact that the diffused data distribution is Gaussian, which is achieved by using a prefixed time variance schedule. Therefore, theoretically, it should be able to learn the data of any topological structure. Yet, this needs to be examined through experiments. Similarly, energy based models (Du and Mordatch, 2019) also do not assume any topology on the data distribution, and during sampling start from Gaussian distribution and then travel to high probability regions of the data space. Thus, we could expect a similar ability in learning distributions of non-trivial topology.

2.2 Quantitative metrics for evaluating DGMs

Given the purpose of DGMs is to generate samples that are as realistic as possible for a human, the straightforward evaluation method would be the judgments by human eyes. However, there have been attempts to quantitatively measure their performances.

2.2.1 Wasserstein distance

We propose to evaluate how well a generative model learns a data probability distribution using a sample approximation to the L_2 Wasserstein 2-distance. By definition this should be Equation 5:

$$W_2(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \left(\mathbb{E}_{(x,y)\sim\gamma}\|x-y\|^2\right)^{1/2}, \tag{5}$$

where μ and ν are probability measures of the ground truth data and the generative model, respectively, $\Gamma(\mu,\nu)$ is the set of any joint distribution of x and y such that $\int \gamma(x,y)dy = \mu(x)$ and $\int \gamma(x,y)dx = \nu(y)$. We implement the empirical version as Equation 6:

$$W_2(\mu,\nu) = \inf_{\pi}\left(\frac{1}{n}\sum_{i=1}^{n}\|X_i - Y_{\pi(i)}\|^2\right)^{1/2}, \tag{6}$$

where $X_1, X_2, ..., X_n$ are random samples from μ, and $Y_1, Y_2, ..., Y_n$ from ν, and π is any permutation of $1, 2, ..., n$. Since the datasets are simulated, we can easily sample from the ground truth data distribution μ. The best π is obtained using the Jonker–Volgenant algorithm (Jonker and Volgenant, 1988) implemented by SciPy (Virtanen et al., 2020).

There are some existing works (e.g., Genevay et al., 2018), that train generative models from the viewpoint of optimal transport, and therefore include the Wasserstein distance in the training loss. However, to the best of our knowledge, we are the first to employ Wasserstein distance to evaluate how well generators learn the overall data distribution. The high time complexity ($O(n^3)$) of the Jonker–Volgenant algorithm forbids us from using too large sample sizes to represent ground truth and learned distributions. Therefore, one concern is whether the set of samples can adequately cover the whole distribution. However, in our experiments, we use data from known low-dimensional distributions that can be reasonably covered with relatively fewer samples.

2.2.2 Fréchet inception distance

Fréchet Inception Distance (FID) is computed by computing the Wasserstein distance on two probability distributions obtained by feeding a set of ground truth examples and a set of fake examples to an embedding function. The embedding function generally used is Inception v3 trained on ImageNet with the final layer truncated, yielding a 2048-dimensional vector for each sample. A normal distribution is fit in this space for each of the ground truth and fake sets, which are then the direct inputs for the Wasserstein distance. While FID has been shown to usually align with human judgement (Heusel et al., 2017), it has a number of shortcomings (Chong and Forsyth, 2020; Parmar et al., 2022). Despite its shortcomings, FID has established itself as the de facto standard metric for judging the quality of generative images (Borji, 2022).

2.2.3 Density, coverage

A line of work has defined metrics that separate failure modes by using multi-valued metrics. For example, a metric might focus on *fidelity* which captures the degree to which a generated image resembles those in a dataset, whereas another might focus on *diversity* which captures the degree to which a sample reflects the variation in generative factors that gives rise to a dataset.

The earliest work, Precision and Recall (Sajjadi et al., 2018), introduces two metrics that successfully separate dropping and adding modes (recall) from image quality (precision), but have some shortcomings including not being robust to outliers and requiring more significant tuning to be accurate.

Density and Coverage (Naeem et al., 2020) address these limitations by, still in an embedding space, defining a manifold for a set of ground truth examples and measuring how often generated points land in it. For their reported results, they use the 4,096-dimensional layer of a truncated VGG16 trained on ImageNet as the embedding space. They then form the real manifold as k-nearest neighbor balls for each real point. Density is then a cumulative measure of how many real neighborhood balls the generated points land in, normalized for the number of points. Intuitively, this value is greater than 1 when many generated samples occur in a few real modes and less than 1 when the generated samples are too diverse or don't fall in real modes. The other half of the metric, Coverage, is then the percentage of real neighborhood balls that have a generated point within them. Intuitively, this is 1 when all modes of the original data are covered, and less than 1 otherwise.

2.2.4 Topological data analysis: persistent homology

Here we give some brief intuition about the information carried by the *persistent homology* of a point cloud. Readers interested in a fuller and more rigorous discussion are pointed to textbooks such as Edelsbrunner and Harer (2010) or Oudot (2017).

Suppose that $X = \{x_1, \ldots, x_n\}$ is a point cloud in some Euclidean space. For example, let X be the collection of points on the left of Figure 1. The *persistence diagram* $D_k(X)$ is a compact summary of some of the k-dimensional multi-scale shape information carried by X. We now give some more details about what this means.

For each threshold value $r \geq 0$, let $X_r = \bigcup_{i=1}^{n} B_r(x_i)$. Note that whenever $r < s$, we have $X_r \subset X_s$, and as r moves from 0 to ∞, the union of balls around the points in X grows from the points themselves to the entire Euclidean space. During this process, various shape changes occur. In our working example, as r increases, the number of connected components, which began as $|X|$, rapidly becomes 3 as clusters form and then subsequently decreases as those clusters merge. The r values at which these mergers happen are recorded as *death* values and stored in the

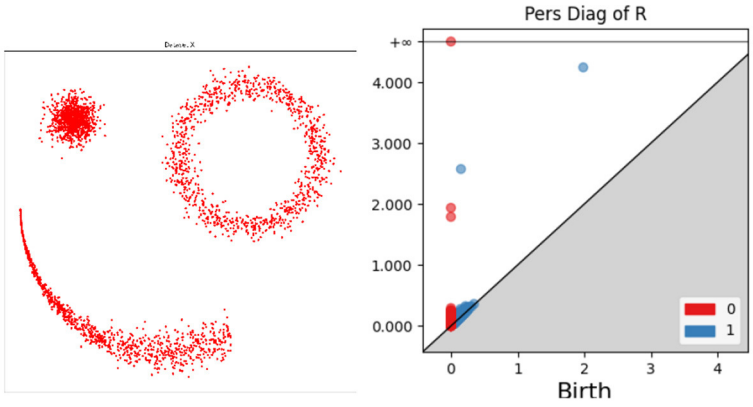


FIGURE 1
Illustration of persistence diagrams **(right)** for the Rips homology filtration on a point cloud **(left)**. Persistence is shown in dimensions zero (red) and one (blue).

zero-dimensional persistence diagram $D_0(X)$; see the red dots on the right side of Figure 1. The higher-level connectivity of the union of balls also changes as r increases. In our working example, an annulus forms in the upper right of X at a very small value of r, and a ring appears connecting the three clusters at a larger value of r. In technical terms, these features are called one-dimensional homology classes (Edelsbrunner and Harer, 2010) and have rigorous algebraic definitions. The r values at which they first appear are called *birth value*. Each homology class eventually fills in as r increases; for example, the annulus at the upper right fills in at the apparent radius of the feature. These *death values* of the one-dimensional features are paired with the birth values that created the feature, and they are plotted in the one-dimensional persistence diagram $D_1(X)$; see the blue dots on the right side of Figure 1.

Thus, each persistence diagram $D_k(X)$ consists of a (multi-) set of dots in the plane, with each dot recording the birth and death value of a k-dimensional homological feature. Intuitively 0 and 1 dimensional features represent connected components and loops/holes, respectively. Not shown in this example are two-dimensional features, which represent voids, and still higher-dimensional features. The *persistence* of a feature is the vertical distance of its dot to the major diagonal $y = x$ in the persistence diagram. Higher-persistence features are generally thought of as genuine representatives of the underlying space, while lower-dimensional features are more likely to be caused by sampling noise. This intuition can be formalized in inference theorems (e.g., Cohen-Steiner et al., 2007; Fasy et al., 2014).

Persistence diagrams of point clouds are computed by transforming the growing union of balls into combinatorial objects called filtered simplicial complexes. Without going into the technical details here, we note that many software packages for doing this exist (Otter et al., 2017 gives a nice overview), and that the experiments in this paper use giotto-tda (Tauzin et al., 2021).

3 Experiments

3.1 Datasets

We conduct experiments on two synthetic image datasets and one real dataset. Samples of each dataset are shown in Figure 2.

The "torus" ellipse image dataset contains 10,000 grayscale images of white ellipses on black backgrounds. Each image is of size 32×32 and contains one ellipse. The images are downsampled from 64×64 images so the edges of ellipses are blurred. The ellipse can rotate around itself 0 to π. And because the ellipse is 180−degree rotation symmetric, it renders the topology of $\mathbb{S}^1$. The center point position of the ellipse rotates 0 to 2π around the center of the image with a radius of 7 pixels, which independently renders another $\mathbb{S}^1$ topology. In combination this results in $\mathbb{S}^1 \times \mathbb{S}^1$ topology, i.e. a torus topology.

The rotating jar image dataset is generated using POV-Ray by Persistence of Vision Pty. Ltd. (2004). There are 10,000 RGB colored samples of size 64×64. Each image contains one rotating jar in the fixed center position. The object has random three dimensional orientations and it has rotational symmetry with respect to the axis that connects the lid knob and the center point of the bottom. Therefore, the image is defined given the orientation of the lid knob. This indicates that the data has a S^2 topology.

The cyclooctane dataset consists of 6,040 points in $\mathbb{R}^{24}$, corresponding to conformations of the cyclooctane molecule (C_8H_{16}) (Martin et al., 2010). A *conformation* is a configuration of atoms in a molecule up to rotation and translation of the molecule. Physical chemistry constraints for cyclooctane imply the positions of the 16 hydrogen atoms are determined by the positions of the 8 carbon atoms in each conformation (Hendrickson, 1967; Martin et al., 2010). Each point in the dataset consists of the 8 spatial coordinates of the carbon atoms flattened into a single vector, as in $[(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_8, y_8, z_8)]$ becomes $(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_8, y_8, z_8) \in \mathbb{R}^{24}$.

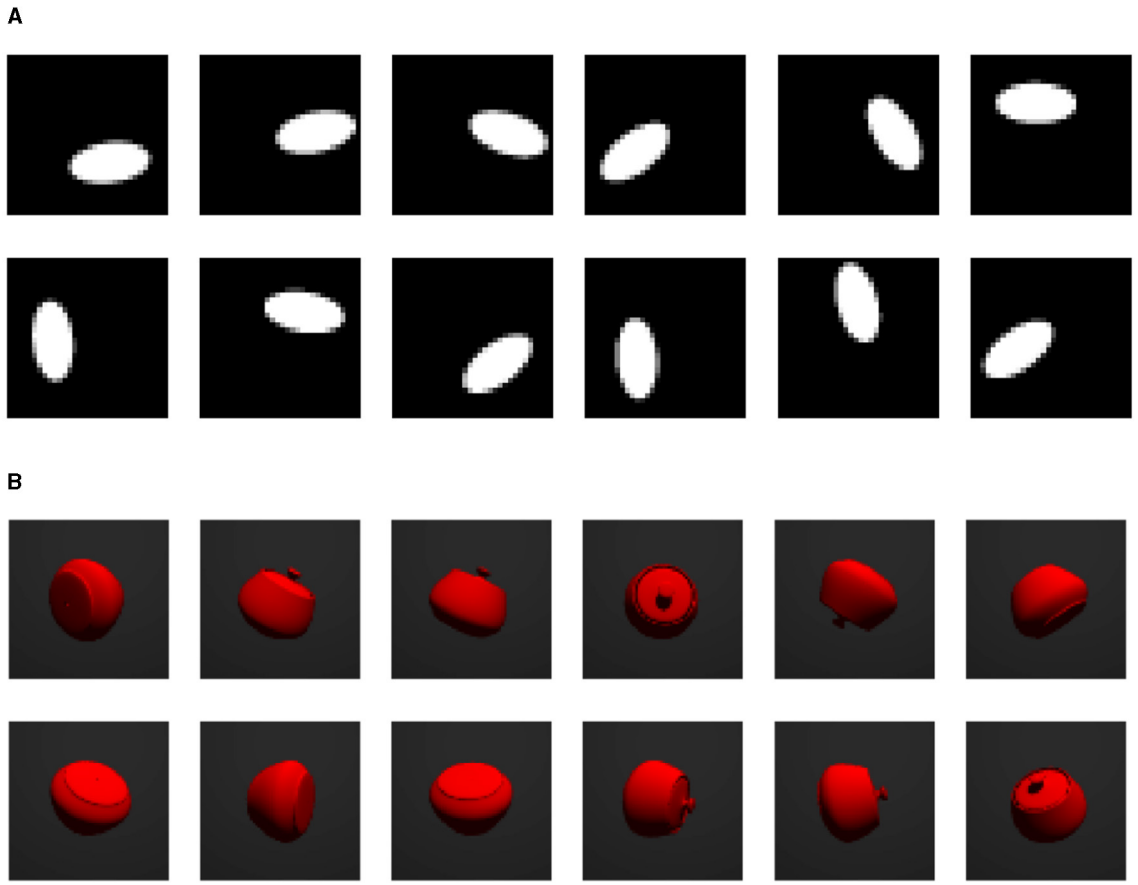


FIGURE 2
Samples from each dataset. **(A)** "Torus" ellipse image dataset. **(B)** Rotating jar image dataset.

3.2 Training setups

Here we introduce our training setups of different generative models. We adopted relatively simple architectures that are capable of generating reasonably good quality samples. VAE and DDPM used for the same dataset are designed to have a similar number of parameters, so that we know the performance difference is not because of different parameter numbers. Training hyper-parameters, including learning rates, epochs, weight values for VAE loss terms, and total time steps for DDPM, are determined using Bayesian search (Falkner et al., 2018) over a set of different options. Therefore, the hyper-parameters for each model are different but they are chosen to maximize the performance. Every model is trained using Adam optimizer (Kingma and Ba, 2015). For more details see Supplementary material.

3.3 Qualitative evaluation

First, we evaluate each generative model qualitatively by observing randomly generated samples and interpolations between two data points.

Samples from generators trained on the "torus" dataset are shown in Figure 3. We can see that DDPM produces high quality samples that are almost indistinguishable from ground truth images by human eyes. The ellipses have clear edges and are always in the same correct shape. In contrast, VAE sometimes generates clearly invalid images. The ellipse shapes are completely lost in some cases. Figure 4 shows samples from DGM trained on the rotating jar. Both VAE and DDPM generally produce credible images. However, we can see that VAE occasionally fails and generates misshapen jars. These results could be due to VAE not learning the correct topology of the dataset and possibly sampling on the "holes" of the torus or the sphere. We will explore this further in the following subsections.

We also performed interpolation between two data points using different generators, visualized in Figures 5, 6. For the VAE, we linearly interpolate the latent space, which results in invalid images in the middle (5−th image for the "torus" and 3−rd image for the jar). We assume this happens because when we linearly interpolate between two points, we travel across the void of the latent distribution, where the VAE decoder cannot map to valid data points. For the DDPM, two end images are diffused for several time steps ($t = 250$ for the "torus" and $t = 350$ for the jar) and then linearly interpolated. Next, we apply the usual denoising steps until reversing back to $t = 0$ to get clean images. Due to the stochasticity in both the forward and reverse processes, the endpoints will be different from the original images to a certain degree, depending on the diffusing time

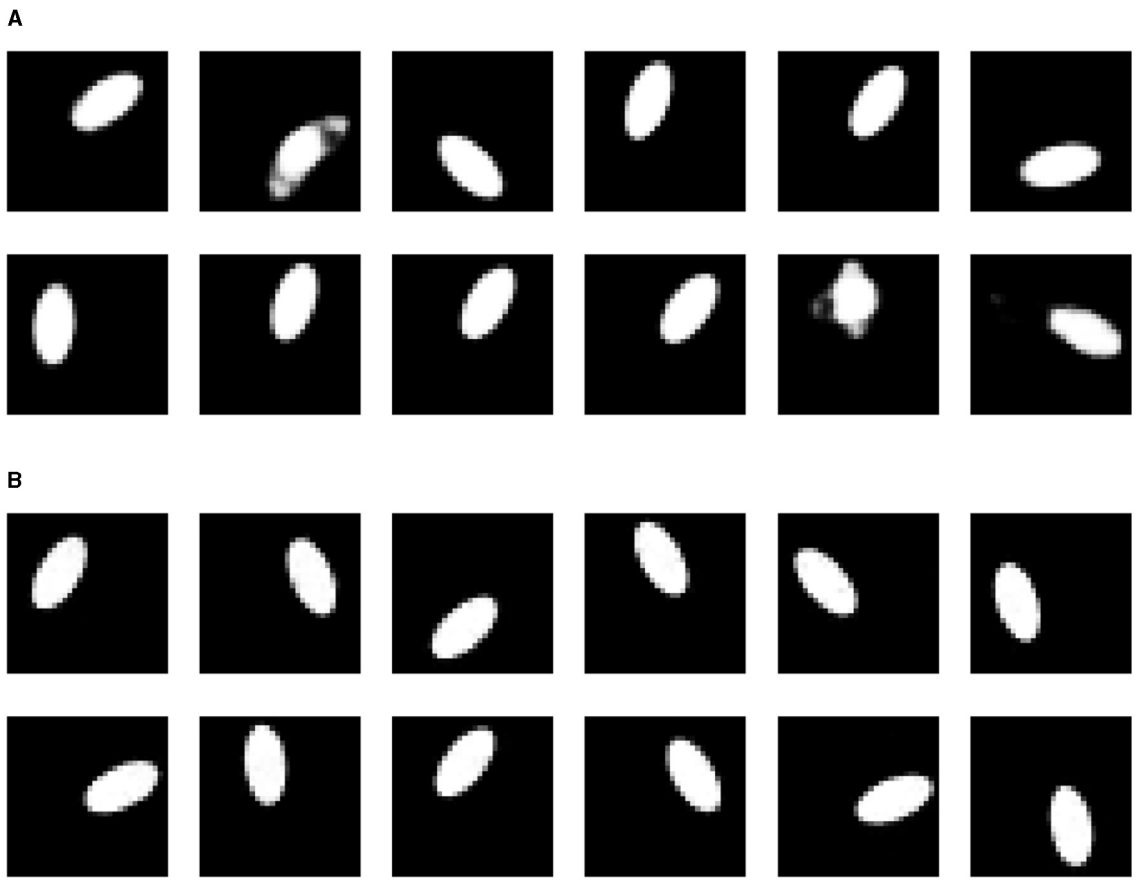


FIGURE 3
Random samples from different generators on "torus" ellipse image dataset. **(A)** Variational autoencoder. **(B)** Denoising diffusion probabilistic model.

steps. We can also see that although the generated images look valid but do not provide a reasonably continuous interpolation. This can be considered as a shortcoming of DDPM not having a latent space.

In Figure 7, we see generated samples of cyclooctane under our different architectures. To visualize the conformations of cyclooctane, we embed the $\mathbb{R}^{24}$ representations in $\mathbb{R}^3$ using Isomap. This embedding is locally isometric and has been used in literature such as Martin et al. (2010). The original embedding of the dataset is visible on the left. Notice the geometry of this manifold involves a Klein bottle enveloped by a sphere. We find that the vanilla VAE struggles to generate conformations associated with the Klein bottle. This is not ideal as these conformations are associated with specific conformational states that do not correspond to any points on the outer sphere. Matching our intuition, the CAE is able to better cover the manifold of cyclooctane, where the embedded color represents chart membership. Still, we find the outer shell of the sphere is sparsely covered. Perhaps counterintuitively, the DDPM model visually best samples the data manifold. It is clear that the samples cover both the Klein bottle and the outer sphere with reasonable density.

3.4 Quantitative performance metrics

For each of the three datasets and each DGM, we computed the L_2 Wasserstein metric between a sample set from the ground truth data distribution and a sample set generated by the DGM models. Because of the computational complexity of the Jonker-Volgenant algorithm, we were limited to computing with sample sizes of 3,000 data in both ground truth and DGM. To ensure that the metric values were stable at the given sample size, we repeated the metric calculation 10 times, each time with an independently drawn sample from both the ground truth and the DGM. For the cyclooctane dataset, since we only have 6,040 samples in the ground truth data, we randomly draw 3,000 samples each time without replacement. Results are shown in Table 1. Since the sample size used to approximate the Wasserstein distance is limited, to rule out the effects of random sampling, we also performed t-tests on Wasserstein distance observations and the resulting p-values are listed in Table 2. It is clear that the Wasserstein distances for different models are significantly different, except for between the VAE and CAE trained on cyclooctane, which have very similar results. We can see that on the image datasets, VAE has a consistently smaller distance to the ground truth data distribution

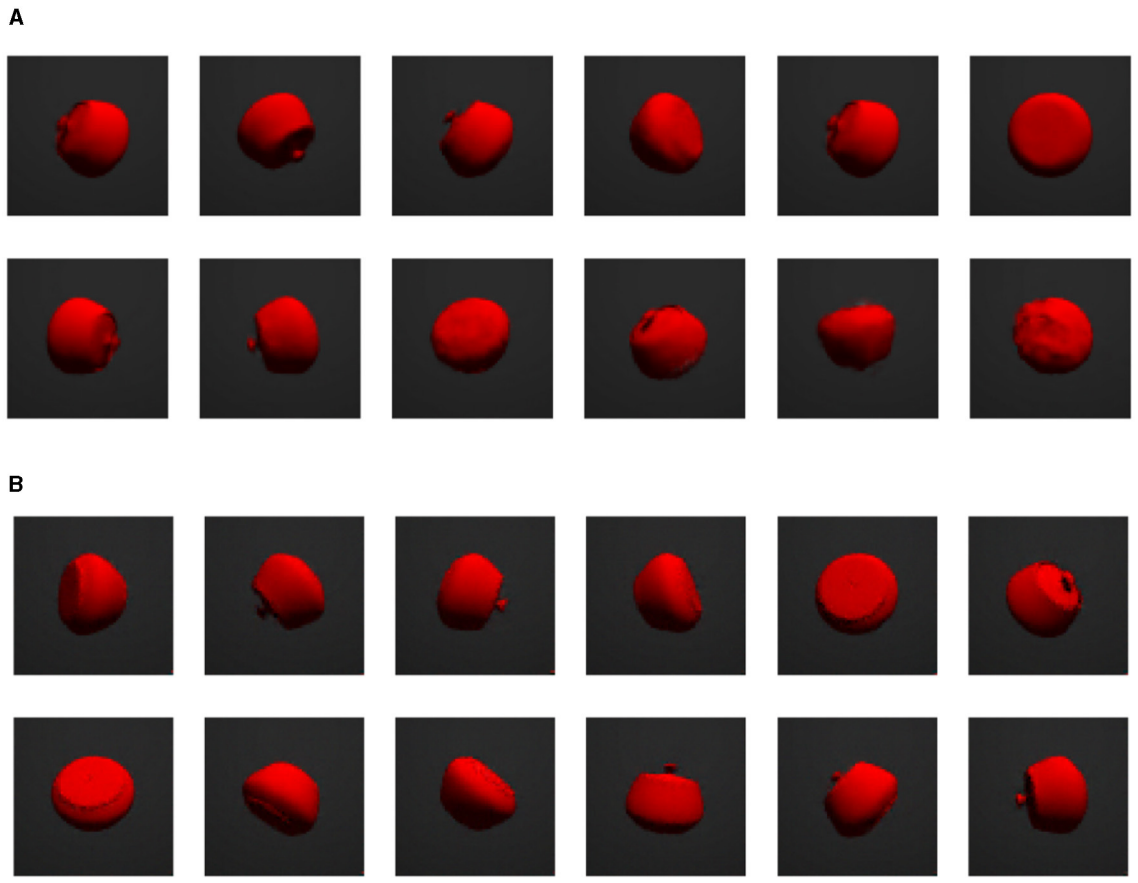


FIGURE 4
Random samples from different generators on rotating jar image dataset. **(A)** Variational autoencoder. **(B)** Denoising diffusion probabilistic model.

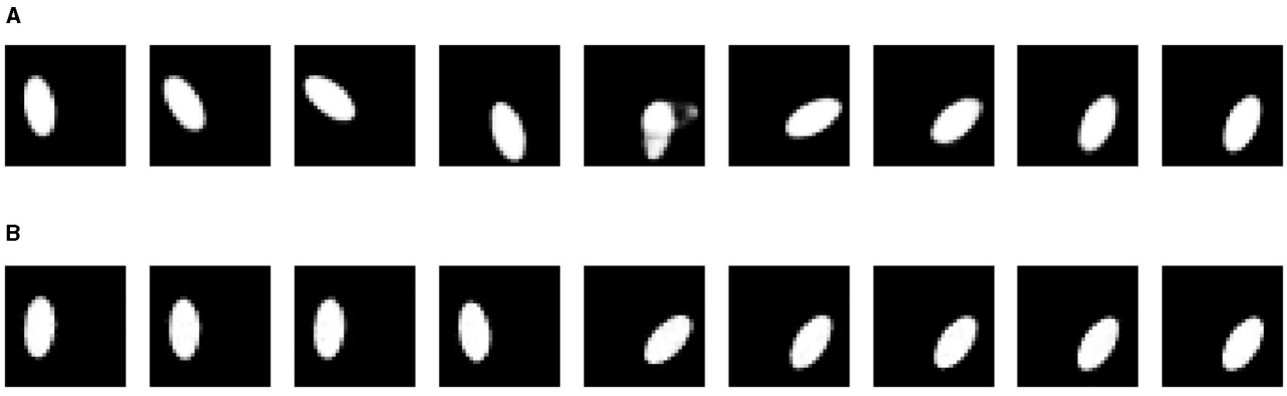


FIGURE 5
Interpolations from different generators on "torus" ellipse image dataset. **(A)** Variational autoencoder. **(B)** Denoising diffusion probabilistic model.

than DDPM, despite what appears to be worse image quality to human eyes. The result is different for the cyclooctane dataset, with DDPM having a significantly smaller distance while CAE has a similar result to VAE. This should indicate in some datasets VAE is learning the overall distribution better than DDPM. It could be the case that in terms of L_2 distance, although DDPM samples are more precise, or more close to the ground truth distribution, VAE samples cover the whole data distribution better. And we can also see that the probability based metric alone does not sufficiently represent the real world performance of models.

For the cyclooctane dataset, we calculated the bond lengths of generated samples and compared them to the bond lengths of the true cyclooctane data. Bond lengths for the true data are tightly distributed about the mean value of 1.52 Å with a

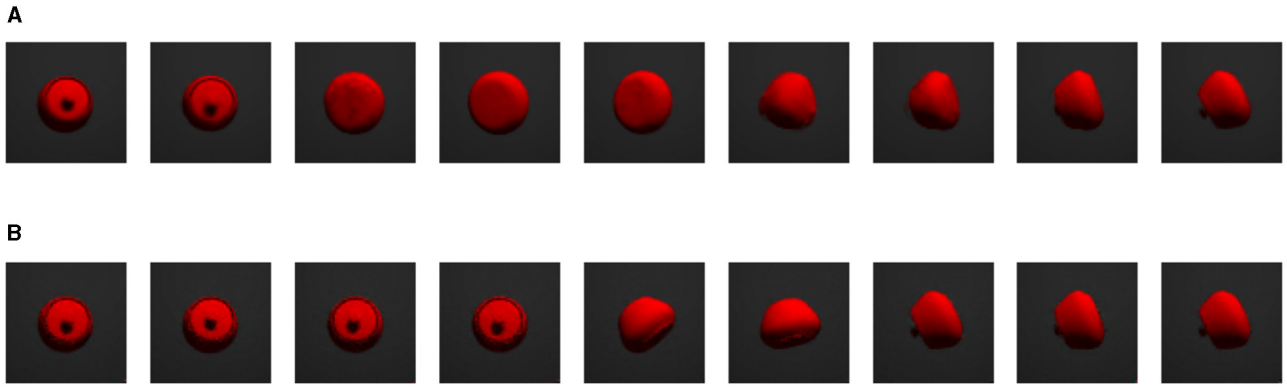


FIGURE 6
Interpolations from different generators on rotating jar image dataset. **(A)** Variational autoencoder. **(B)** Denoising diffusion probabilistic model.

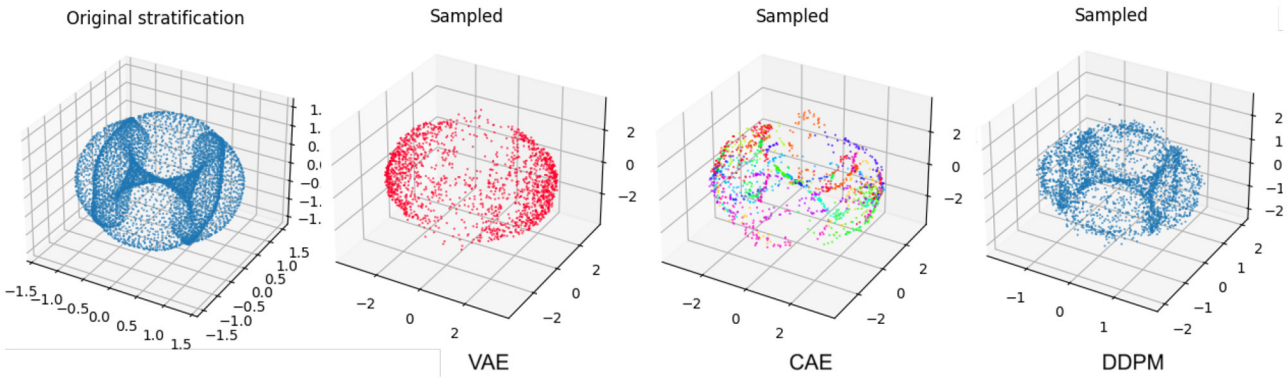


FIGURE 7
A comparison of the cyclooctane conformations generated by sampling the various diffusion models. On the left, we display the Isomap embedding of the original cyclooctane data. For the CAE embeddings, the different colors denote the corresponding chart. Notice that the vanilla VAE struggles to generate the inner Klein bottle of the Isomap embedding. Counterintuitively, the DDPM generations most resemble the original data manifold even though its latent space is high dimensional.

TABLE 1 L_2 Wasserstein distance.

	"Torus" ellipse	Rotating jar	Cyclooctane
Ground truth	2.01 (±0.15)	2.05 (±0.06)	0.215 (±0.009)
VAE	2.26 (±0.05)	2.17 (±0.05)	0.860 (±0.004)
DDPM	2.65 (±0.19)	3.20 (±0.10)	0.389 (±0.011)
CAE	-	-	0.860 (±0.010)

Reporting mean and standard deviation over 10 independent runs, each time sampling $n = 3{,}000$ images from both the ground truth data distribution and generators.

TABLE 2 p-value of Wasserstein distance observations.

	"Torus" ellipse	Rotating jar	Cyclooctane
Ground truth & VAE	$9.28e-5$	$1.26e-4$	$7.47e-32$
Ground truth & DDPM	$1.30e-7$	$4.04e-17$	$8.70e-19$
Ground truth & CAE	-	-	$2.04e-29$
VAE & DDPM	$6.41e-6$	$1.35e-16$	$4.76e-28$
VAE & CAE	-	-	1
DDPM & CAE	-	-	$3.50e-26$

standard deviation of $4.09e-05$ Å. Figure 8 shows the distribution of each sample set's bond lengths. We can see that although the sample bond lengths of all the generative models are much more dispersed than the ground truth values, DDPM has a relatively better distribution. The expected errors of each distribution to the mean ground truth value are also calculated. This error is 0.165 Å for VAE, 0.155 Å for CAE and 0.04 Å for DDPM.

We also computed deep-learning-based metrics - FID, density, and coverage, for our two image datasets. The sample sizes of 50,000 are used for both ground-truth data distribution and the DGM learned distribution. The deep learning model used for embedding is VGG16 "IMAGENET1K_V1". The FID results are shown in

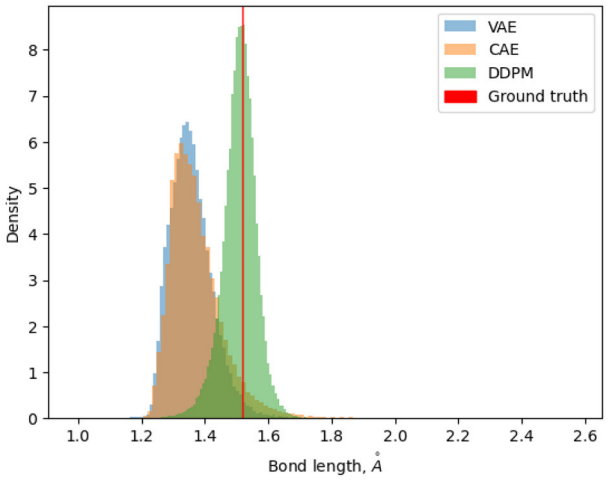


FIGURE 8
Density histogram of bond lengths of samples generated by different models, compared to the ground truth bond lengths.

TABLE 3 Torchmetrics implementation of FID using 50,000 samples. Lower is better.

	"Torus" ellipse	Rotating jar
VAE	16.77	77.72
DDPM	15.00	74.90

TABLE 4 Density / Coverage.

	"Torus" ellipse	Rotating jar
VAE	0.895 / 0.878	0.403 / 0.605
DDPM	0.903 / 0.951	0.943 / 0.903

Reference implementation from Naeem et al. (2020) with $k = 5$ and torchvision pretrained VGG16 "IMAGENET1K_V1" as the embedding. 50, 000 samples. Density is positively valued, with a value of 1 being ideal; values greater than 1 represent generated data occurring near common modes in the real data more often, and values less than 1 represent generated data occurring less often near real data. Coverage is in the range [0, 1] with 1 being optimal; it represents the percentage of real points that are covered by a generated point.

Table 3, and density and coverage results are shown in Table 4. Unlike in the case of Wasserstein distance, DDPM constantly has better metrics values than VAE. This could indicate that the deep learning model used to embed images does capture image features in a way that matches better with human visual experiences. The much larger sample size might also influence the results.

3.5 Topological properties

We also report the persistent homology of ground truth data and samples from generators. Giotto-tda (Tauzin et al., 2021) is used to obtain the results. As introduced in Section 2.2.4, the results show when a topological feature was born and died. Zero-dimensional features are connected components, one-dimensional features are loops, and two-dimensional features are voids (e.g., spheres). The further a point on the persistence diagram is from the diagonal line of "birth = death," the longer they persist across a range of scales, that is, distance thresholds determining when points are connected. These points that stand out beyond the diagonal are more likely to indicate a topological structure.

As we can see in Figure 9A, the "torus" dataset has two significant one-dimensional loops [approximately $(2.5, 8)$ and $(2.5, 13)$] and one two-dimensional sphere [approximately $(6, 8)$] because of its torus topology. The VAE captures this topological structure poorly (Figure 9B), and only significantly captures one one-dimensional loop structure. Although there are many other points relatively far above the diagonal line, there are no points that stand out from the others clearly. On the other hand, DDPM preserves this structure very well (Figure 9C), and we can clearly identify two one-dimensional loops [the points located at approximately $(3, 8)$ and $(3.5, 13)$] and one two-dimensional sphere [located at approximately $(7, 8)$].

This result gives insight into the fact that the VAE sometimes generates invalid samples despite its smaller Wasserstein distance. More intuitively, we show the PCA visualization of the data and the generator samples in Figure 10. We can clearly see that VAE wrongly generates samples in the middle of the torus and violates the original data topology, but DDPM does not. The results for the jar dataset are displayed in Figure 11. As we discussed above, the data has a spherical topology, which is indicated by a significant dimension 2 point in the persistence diagram in Figure 11A [located at approximately $(5.5, 7.5)$]. This structure is clearly better preserved by DDPM [approximately $(5.5, 7.5)$]. Whereas in the persistence diagram of the VAE model, the two-dimensional structure is much less significant, and also an incorrect one-dimensional loop appears [located at approximately $(3, 5.5)$].

4 Discussion and conclusion

In this paper, we investigated the ability of DGMs to model data distributions with non-trivial topologies. We hypothesized that VAEs would struggle to faithfully model non-Euclidean topologies because they generate data by continuously transforming a Gaussian random vector from a lower-dimensional, Euclidean latent space. This hypothesis was supported by our experiments on datasets with known topology. Our results comparing persistence diagrams of generated VAE samples vs. the ground truth persistence diagram show that a VAE does not faithfully recover the correct topology in the case of the torus ($\mathbb{T}^2$) or the sphere (S^2). We further hypothesize that a similar failure to capture topology would hold for other models based on a Euclidean latent space, e.g., GANs, although this would need to be verified with further experiments.

Conversely, we hypothesized that DDPM and related score-based models, which theoretically have no constraint on their topology and learn the distribution in the original data dimension, would more effectively capture non-trivial data topologies. This turned out to be the case in our image experiments, where the DDPM persistence diagrams showed that they generated samples with much better matches to the ground-truth data topology. Furthermore, the ability of DDPMs to adapt to the topology of the data may explain their improved

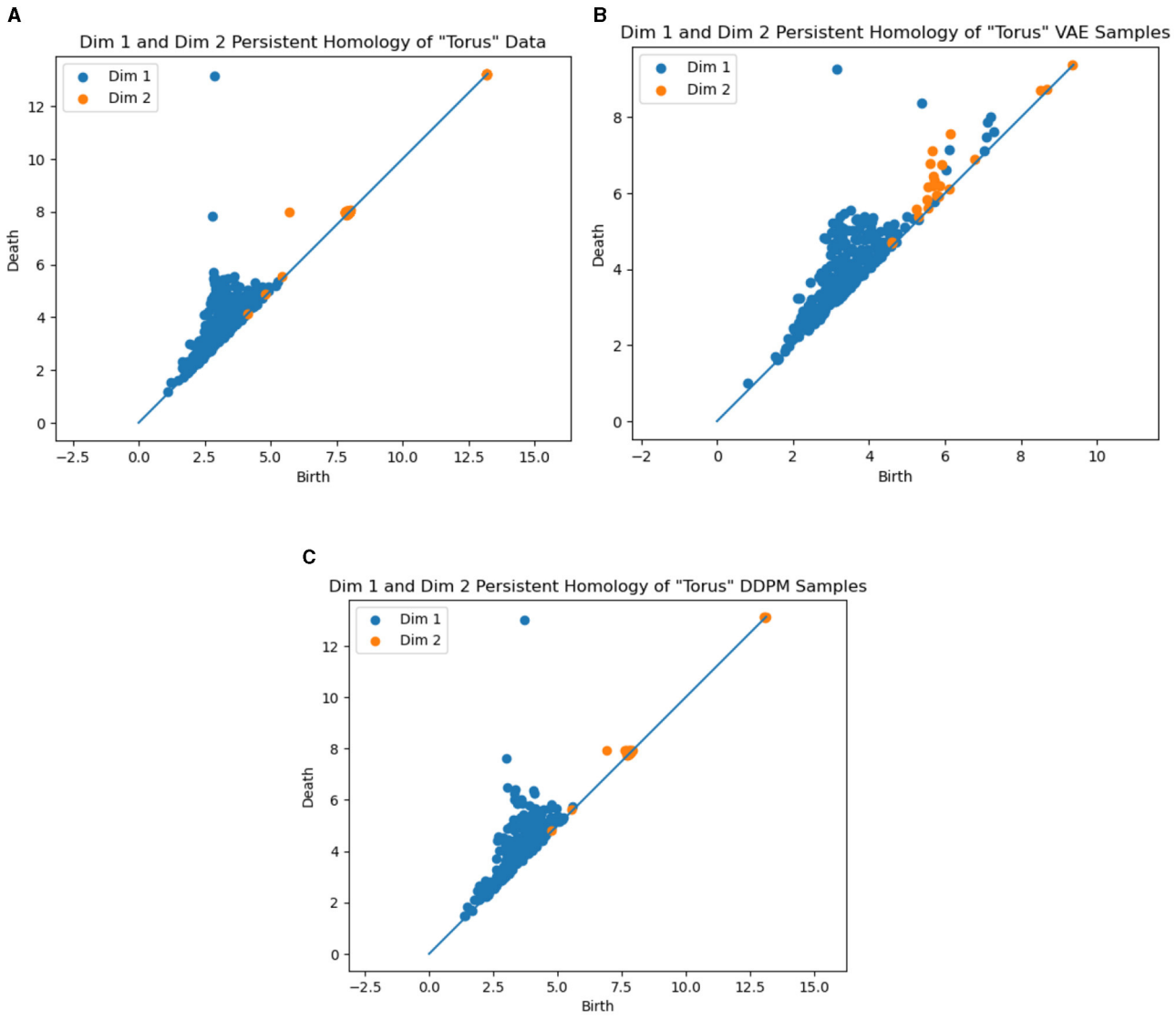


FIGURE 9
Persistent homology of ground truth data and generator samples on the "torus" dataset. **(A)** Ground truth data. **(B)** Samples from VAE. **(C)** Samples from DDPM.

performance in generating realistic data samples, as they can avoid sampling in "holes" of the data distribution. However, one downside to DDPMs is that they do not parameterize the data distribution with a low-dimensional latent space. This makes moving along the data manifold, such as in the case of interpolation, more difficult with DDPMs. The CAE model tries in a sense to bridge this gap by providing a low-dimensional latent space, while at the same time also providing more topological flexibility. Our cyclooctane results show qualitatively and quantitatively that the CAE performs well on a complex data topology.

One unexpected result is the disagreement between the L_2 Wasserstein metric and the other quantitative metrics (FID, density, and coverage). It may be the case the restriction on the sample size for the Wasserstein metric limits its approximation accuracy. Or it may be the case that the exact matching of points between the two samples is prone to outliers or other artifacts in the samples. Or it may simply be that "perceptual distances" mimicked by the VGG16 network are substantially different enough from L_2 distances to cause reverse conclusions in the two classes of metrics. This discrepancy, and the more general question of how to best measure the distribution quality of a DGM, are directions ripe for future research.

In conclusion, our main novel contribution is the first test of the abilities of generative models to handle different data topologies. Our empirical findings highlight the limitations of a simplistic data topology assumption. The main takeaways are as follows:

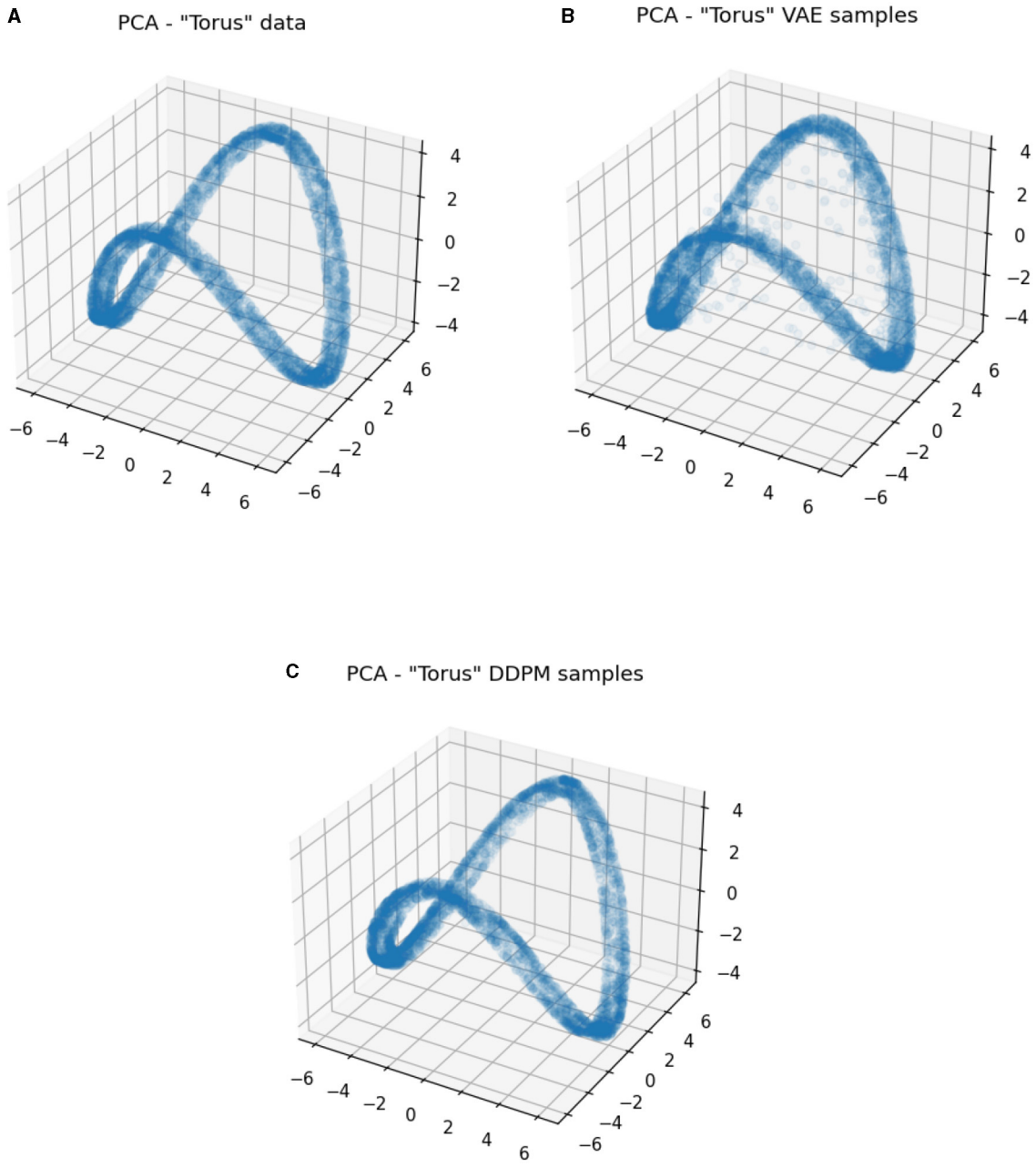


FIGURE 10
PCA visualizations of ground truth data and generator samples on the "torus" dataset. **(A)** Ground truth data. **(B)** Samples from VAE. **(C)** Samples from DDPM.

- Generative models that assume data can be continuously mapped from a Euclidean latent space, e.g., VAEs, have limited ability to capture more complex topologies present in data.
- Conversely, DDPMs operate in the full-dimensional data space and without assumptions about the data topology. This results in DDPMs being better able to capture non-trivial topologies in data.
- However, the absence of straightforward Euclidean latent spaces in DDPM presents obstacles, particularly in tasks such as interpolations.
- Finally, our research underscores that distribution-based evaluation metrics sometimes fail to provide a comprehensive assessment of a generative model's ability to accurately capture the underlying data topology.

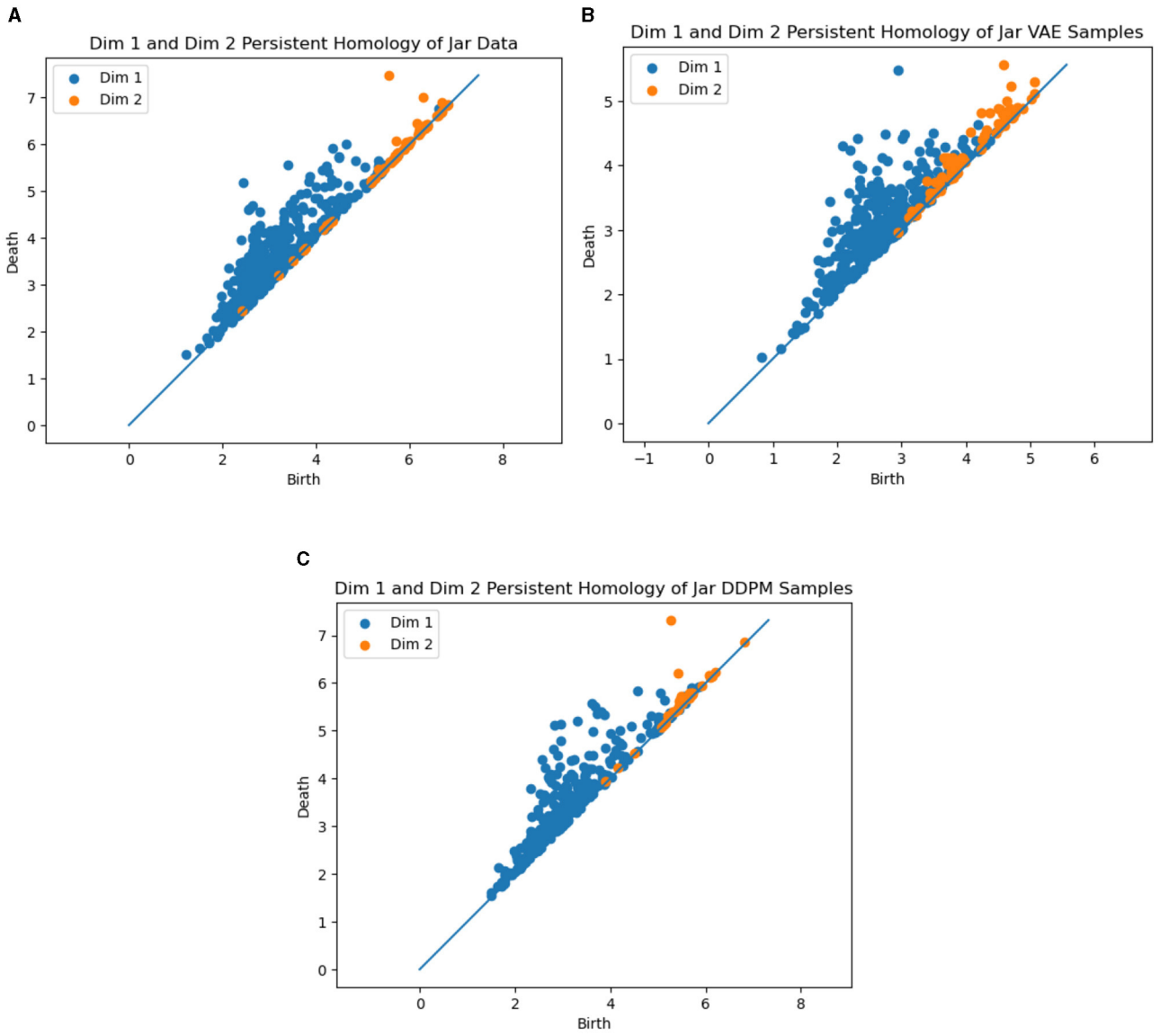


FIGURE 11
Persistent homology of the ground truth data and generator samples on jar dataset. **(A)** Ground truth data. **(B)** Samples from VAE. **(C)** Samples from DDPM.

Data availability statement

The datasets presented in this study are either publicly available or can be generated from publicly available packages/softwares that can be found at online repositories. The names of the repository/repositories and accession number(s) can be found in the article/Supplementary material.

Author contributions

YJ: Writing – original draft, Writing – review & editing. RM: Writing – original draft. NT: Writing – original draft. MC: Writing – original draft. AS: Writing – original draft, Writing – review & editing. PB: Writing – original draft, Writing – review & editing. MD: Writing – original draft, Writing – review & editing. PF: Writing – original draft, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was funded by the Agreement HR0011-22-9-0076 from the Defense Advanced Research Projects Agency (DARPA), as part of the Geometries of Learning (GoL) Artificial Intelligence Exploration (AIE) program, the National Science Foundation under awards 2019239, 2129824, and 2205417, and the Air Force Office of Scientific Research under award number FA9550-21-0164.

Conflict of interest

NT is employed by STR. MC, AS, and PB are employed by Geometric Data Analytics, Inc.

The remaining 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.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2024.1260604/full#supplementary-material

References

Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2018). "Latent space oddity: on the curvature of deep generative models," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*.

Barannikov, S., Trofimov, I., Sotnikov, G., Trimbach, E., Korotin, A., Filippov, A., et al. (2021). "Manifold topology divergence: a framework for comparing data manifolds," in *Proceedings of Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, 7294–7305.

Borji, A. (2022). Pros and cons of GAN evaluation measures: new developments. *Comput. Vis. Image Underst.* 215:103329. doi: 10.1016/j.cviu.2021.103329

Carriére, M., Chazal, F., Ike, Y., Lacombe, T., Royer, M., and Umeda, Y. (2020). "Perslay: a neural network layer for persistence diagrams and new graph topological signatures," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, 2786–2796.

Chen, C., Ni, X., Bai, Q., and Wang, Y. (2019). "A topological regularizer for classifiers via persistent homology," in *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, 2573–2582.

Chen, N., Klushyn, A., Kurle, R., Jiang, X., Bayer, J., and van der Smagt, P. (2018). "Metrics for deep generative models," in *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, 1540–1550.

Chong, M. J., and Forsyth, D. (2020). Effectively unbiased FID and inception score and where to find them. *arXiv:1911.07023*.

Cohen-Steiner, D., Edelsbrunner, H., and Harer, J. (2007). Stability of persistence diagrams. *Discr. Comput. Geom.* 37, 103–120. doi: 10.1007/s00454-006-1276-5

Du, Y., and Mordatch, I. (2019). Implicit generation and generalization in energy-based models. *arXiv:1903.08689*.

Edelsbrunner, H., and Harer, J. (2010). *Computational Topology - an Introduction*. New York: American Mathematical Society. doi: 10.1090/mbk/069

Falkner, S., Klein, A., and Hutter, F. (2018). "BOHB: robust and efficient hyperparameter optimization at scale," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 1436–1445.

Fasy, B. T., Lecci, F., Rinaldo, A., Wasserman, L., Balakrishnan, S., and Singh, A. (2014). Confidence sets for persistence diagrams. *Ann. Stat.* 42, 2301–2339. doi: 10.1214/14-AOS1252

Genevay, A., Peyré, G., and Cuturi, M. (2018). "Learning generative models with sinkhorn divergences," in *International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, 1608–1617.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). "Generative adversarial nets," in *Proceeding of Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, NEURIPS 2014*, 2672–2680.

Hendrickson, J. B. (1967). Molecular geometry. V. Evaluation of functions and conformations of medium rings. *J. Am. Chem. Soc.* 89, 7036–7043. doi: 10.1021/ja01002a036

Hensel, F., Moor, M., and Rieck, B. (2021). A survey of topological machine learning methods. *Front. Artif. Intell.* 4:681108. doi: 10.3389/frai.2021.681108

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Adv. Neural Inf. Process. Syst.* 30, 6626–6637.

Hinton, G. E., and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science* 313, 504–507. doi: 10.1126/science.1127647

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Adv. Neural Inf. Process. Syst.* 33, 6840–6851. doi: 10.48550/arXiv.2006.11239

Jonker, R., and Volgenant, T. (1988). "A shortest augmenting path algorithm for dense and sparse linear assignment problems," in *Proceedings of the 16th Annual Meeting of DGOR in Cooperation with NSOR*, 622–622. doi: 10.1007/978-3-642-73778-7_164

Khrulkov, V., and Oseledets, I. V. (2018). "Geometry score: a method for comparing generative adversarial networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2626–2634.

Kingma, D. P., and Ba, J. (2015). "Adam: a method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*.

Kingma, D. P., and Welling, M. (2014). "Auto-encoding variational bayes," in *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014*.

Martin, S., Thompson, A., Coutsias, E. A., and Watson, J.-P. (2010). Topology of cyclo-octane energy landscape. *J. Chem. Phys.* 132:234115. doi: 10.1063/1.3445267

Naeem, M. F., Oh, S. J., Uh, Y., Choi, Y., and Yoo, J. (2020). "Reliable fidelity and diversity metrics for generative models," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, 7176–7185.

Naitzat, G., Zhitnikov, A., and Lim, L.-H. (2020). Topology of deep neural networks. *J. Mach. Learn. Res.* 21, 184:7503–184:7542. doi: 10.48550/arXiv.2004.06093

Nigmetov, A., and Morozov, D. (2022). Topological optimization with big steps. *arXiv:2203.16748*.

Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., and Harrington, H. A. (2017). A roadmap for the computation of persistent homology. *EPJ Data Sci.* 6, 1–38. doi: 10.1140/epjds/s13688-017-0109-5

Oudot, S. Y. (2017). *Persistence Theory: From Quiver Representations to Data Analysis*. New York: American Mathematical Society.

Parmar, G., Zhang, R., and Zhu, J.-Y. (2022). On aliased resizing and surprising subtleties in GAN evaluation. *arXiv:2104.11222*.

Persistence of Vision Pty. Ltd. (2004). *Persistence of vision raytracer (version 3.6)* [*computer software*]. Available online at: http://www.povray.org/download/ (accessed July 18, 2023).

Rezende, D. J., and Mohamed, S. (2015). "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 1530–1538.

Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. (2018). "Assessing generative models via precision and recall," in *Proceedings of Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 5234–5243.

Schonsheck, S., Chen, J., and Lai, R. (2019). Chart auto-encoders for manifold structured data. *arXiv:1912.10094*.

Shao, H., Kumar, A., and Fletcher, P. T. (2018). "The Riemannian geometry of deep generative models," in *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018*, 315–323. doi: 10.1109/CVPRW.2018.00071

Shukla, A., Uppal, S., Bhagat, S., Anand, S., and Turaga, P. K. (2018). "Geometry of deep generative models for disentangled representations," in *Proceedings of ICVGIP 2018: 11th Indian Conference on Computer Vision, Graphics and Image Processing*, 68:1–68:8. doi: 10.1145/3293353.3293422

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 2256–2265.

Solomon, E., Wagner, A., and Bendich, P. (2021). "A fast and robust method for global topological functional optimization," in *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021*, 109–117.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). "Score-based generative modeling through stochastic differential equations," in *Proceedings of the 9th International Conference on Learning Representations, ICLR* 2021.

Tauzin, G., Lupo, U., Tunstall, L., Pérez, J. B., Caorsi, M., Medina-Mardones, A. M., et al. (2021). giotto-tda:: a topological data analysis toolkit for machine learning and data exploration. *J. Mach. Learn. Res.* 22, 39:1–39:6. doi: 10.48550/arXiv.2004.02551

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in python. *Nat. Methods* 17, 261–272. doi: 10.1038/s41592-019-0686-2

Wheeler, M., Bouza, J., and Bubenik, P. (2021). "Activation landscapes as a topological summary of neural network performance," in *Proceedings of 2021 IEEE International Conference on Big Data*, 3865–3870. doi: 10.1109/BigData52589.2021.9671368

frontiers | Frontiers in Computer Science

TYPE Brief Research Report
PUBLISHED 15 January 2025
DOI 10.3389/fcomp.2024.1255517

OPEN ACCESS

EDITED BY
Anuj Srivastava,
Florida State University, United States

REVIEWED BY
Xin Li,
West Virginia University, United States
Henry Kirveslahti,
Swiss Federal Institute of Technology Lausanne, Switzerland

*CORRESPONDENCE
Yunye Gong
yunye.gong@sri.com

RECEIVED 09 July 2023
ACCEPTED 17 December 2024
PUBLISHED 15 January 2025

CITATION
Gong Y, Yao J, Lian R, Lin X, Chen C, Divakaran A and Yao Y (2025) Recovering manifold representations via unsupervised meta-learning. *Front. Comput. Sci.* 6:1255517. doi: 10.3389/fcomp.2024.1255517

Recovering manifold representations via unsupervised meta-learning

Yunye Gong[1]*, Jiachen Yao[2], Ruyi Lian[2], Xiao Lin[1], Chao Chen[2], Ajay Divakaran[1] and Yi Yao[1]

[1]Center for Vision Technologies, SRI International, Princeton, NJ, United States, [2]Department of Computer Science, Stony Brook University, Stony Brook, NY, United States

Manifold representation learning holds great promise for theoretical understanding and characterization of deep neural networks' behaviors through the lens of geometries. However, data scarcity remains a major challenge in manifold analysis especially for data and applications with real-world complexity. To address this issue, we propose manifold representation meta-learning (MRML) based on autoencoders to recover the underlying manifold structures without uniformly or densely sampled data. Specifically, we adopt episodic training, following model agnostic meta-learning, to meta-learn autoencoders that are generalizable to unseen samples specifically corresponding to regions with low-sampling density. We demonstrate the effectiveness of MRML via empirical experiments on LineMOD, a dataset curated for 6-D object pose estimation. We also apply topological metrics based on persistent homology and neighborhood graphs for quantitative assessment of manifolds reconstructed by MRML. In comparison to state-of-the-art baselines, our proposed approach demonstrates improved manifold reconstruction better matching the data manifold by preserving prominent topological features and relative proximity of samples.

KEYWORDS

manifold representation learning, autoencoder, meta-learning, persistent homology, data scarcity

1 Introduction

Challenges such as model transferability, explanability, and adversarial robustness prevent the application of deep learning to real-world problems with safety and mission importance. One research direction of growing interest is to address these challenges by studying deep learning from the perspective of geometry and topology (Watanabe and Yamana, 2022; Aktas et al., 2019). Based on the widely accepted assumption that high dimensional data often lie on a low dimensional manifold, manifold representation learning, which seeks to capture underlying manifold structure, serves as a critical first step towards principled geometric and topological analysis of deep learning (Tenenbaum et al., 2000; Bengio et al., 2013).

While autoencoders (Liou et al., 2014; Bank et al., 2020) have been widely adopted for learning intrinsic structure from high dimensional empirical data in an unsupervised manner, sample scarcity and sparsity remains a major challenge for capturing underlying manifolds especially for real-world problems. Existing solutions are often prone to issues of bad generalization and incorrect local geometry due to sparse sampling in high dimensional space and noisy samples of real-world complexity (Lee Y. et al., 2021). Meanwhile, meta-learning has been widely adopted as a technique to address the challenge of data scarcity and to learn models that are easy to adapt given few samples from new task domain (Hospedales et al., 2022; Snell et al., 2017; Finn et al., 2017).

In this work, inspired by model agnostic meta-learning (MAML) originally designed for domain adaptation and domain generalization (Finn et al., 2017; Li et al., 2018), we aim to combine the strength of autoencoders and meta-learning by proposing manifold representation meta-learning (MRML) to improve manifold representation learning considering training distributions containing low sampling density regions. We compare three different sampling schemes that mirror different types of shifts between training and testing distributions in an episodic training. Accordingly, we train models to achieve good generalization performance at different levels of difficulty. Thanks to the improved generalizability of meta-learned models, we demonstrate that manifold regions with low sample density can be faithfully recovered.

To evaluate the generalization performance of MRML, we tap into topological metrics based on persistent homology (Edelsbrunner and Harer, 2008) and neighborhood graphs to quantify the reconstructions at the manifold level. We perform experiments using the LineMOD dataset (Hinterstoisser et al., 2012) which is designed for 6-D pose estimation. We perform both qualitative and quantitative comparison between MRML under three different settings and multiple baseline methods including recent state-of-the-art autoencoders considering local connectivity (Lee Y. et al., 2021) and geometric regularization (Duque et al., 2022). We demonstrate consistent qualitative and quantitative improvement of manifold reconstruction against baselines with respect to topological metrics considering generalization to hold-out test samples corresponding to a missing gap/hole in the complete manifold. In comparison to baseline reconstruction, our best performing meta-learning procedure captures a manifold better matching the data manifold and leading to a relative reduction of topological distance at 14.44% considering the hold-out neighborhood and at 4.44% considering the entire manifold.

In summary, our major contributions include the following:

(1) We propose MRML (manifold representation meta-learning) with novel episodic sampling strategies to improve autoencoders' generalization performance in reconstructing manifold especially for regions with low sampling density.
(2) In addition to standard metrics focusing on sample-level reconstruction accuracy, we introduce topological and geometric metrics based on persistent homology and neighborhood graphs for quantitative evaluation of MRML with respect to manifold reconstruction.
(3) We demonstrate both qualitative and quantitative improvement via MRML against state-of-the-art baselines for manifold reconstruction evaluated based on topological and geometric metrics, using training data with low sampling density regions.

2 Related work

2.1 Manifold representation learning

Autoencoders (Liou et al., 2014; Bank et al., 2020) are commonly adopted for unsupervised representation learning where data in high dimensional input space is projected onto a lower dimensional latent space by an encoder and restored back to data dimension in the output space by a decoder. Several recent works are proposed to incorporate topological analysis in design of autoencoders for preserving the local geometry in unsupervised representation learning. Moor et al. (2020) propose Topological Autoencoder using topological loss to regularize the representation learning and thus improve the alignment between input and latent space based on persistent homology features. Schönenberger et al. (2020) propose Witness Autoencoder (W-AE) to improve the regularization by defining the alignment between input and latent space via geodesic distances computed based on witness complexes. Schonsheck et al. (2019) propose Chart Autoencoder (CAE) which use an ensemble of decoders to model a multi-chart latent space representing the manifold with a collection of overlapping local neighborhoods. With this formulation, the authors discuss the local proximity and manifold approximation theoretically. More recent works investigate the use of geometric regularization such as regularization based on local contraction and expansion of the decoder (Nazari et al., 2023) or isometry to preserve local distance (Gropp et al., 2020; Lee et al., 2022). In this work, we perform qualitative and quantitative comparison against two recent baselines addressing underlying geometry of autoencoders. Lee Y. et al. (2021) propose Neighborhood Reconstructing Autoencoder (NRAE) which seeks to correct local geometry and overfitting of autoencoders simultaneously with novel reconstruction loss leveraging neighborhood graph and local quadratic approximation of the decoder. Duque et al. (2022) propose Geometry Regularized Autoencoders (GRAE) which introduce regularization to specifically match latent representations of the autoencoder to representations from manifold learning computation. In comparison to existing works, our approach does not need explicit modeling or calculation of topological features at set or batch level during the training. It is based on meta-learning framework with episodic training and thus enables improved generalization of manifold learning with respect to topological metrics considering training manifolds consisting low sampling density regions.

2.2 Model agnostic meta learning

Meta-learning or 'Learning to Learn' techniques seek to extract generalizable knowledge from learning processes of diverse tasks to achieve fast adaptation or generalization to new tasks (Andrychowicz et al., 2016; Hospedales et al., 2022) and have demonstrated tremendous success in tasks such as few-shot learning (Snell et al., 2017; Hsu et al., 2019) and domain generalization (Li et al., 2019; Liu et al., 2021). Specifically model agnostic meta-learning (MAML) (Finn et al., 2017) is proposed to learn to achieve good adaptation to new tasks given a few samples with a few steps of gradient descent. The learned model serves an initial condition that is easy to fine-tune. MLDG (Li et al., 2018) extends the model agnostic episodic training framework and learns to achieve good zero-shot transfer by simulating domain shifts in supervised classification tasks. Recent works also apply meta-learning techniques to improve the training of generative models such as Variational AutoEncoders (VAE) where representation

learning is performed considering a sets of related probabilistic models to achieve transferrable representation (Lee D. B. et al., 2021; Wu et al., 2020). Our proposed methods follow the episodic training framework in MLDG and explore novel episodic sampling strategies simulating data shifts in transfer learning based on local geometry without supervised labels. In comparison to recent works based on variational inference, our work does not make any prior assumption such as Gaussian or Gaussian Mixtures (Lee D. B. et al., 2021) about the latent distribution.

3 Method

3.1 Meta-learning for manifold representation learning

We develop unsupervised manifold representation learning based on autoencoders to capture meaningful representation of the data which faithfully encodes the underlying manifolds. To address the challenge of data scarcity, we propose a generalizable model which not only provides good reconstruction at sample level but also preserves underlying geometry of the data (e.g., relative proximity between samples; topological features highlights the shape of underlying manifolds) given unseen data corresponding to low sampling density region in the underlying manifold. We integrate autoencoders with meta-learning for domain generalization based on episodic training (MLDG) and examine novel sampling strategies specifically simulating low sampling regions in manifolds.

Following MLDG framework, we first split data into disjoint meta training set S and meta testing set S'. We adopt episodic training and at each training episode, we sample two disjoint batches from the meta training set S, namely episodic training batch S_{train} and episodic testing batch S_{test}. The split of episodic training batch and episodic testing batch is designed to resemble the distribution shift between source (S) and unseen target data (S') so as to test models' generalization performance. We compute the gradient on S_{train} with respect to model parameters θ and compute the updated parameters θ^* after one step of gradient descent. At the episodic testing step, we perform a virtual evaluation of the updated model on the episodic testing set S_{test} with a task loss term $\mathcal{L}$. Herein, we adopt the Binary Cross Entropy (BCE) loss. Given input $\mathbf{x}$, model f_θ with parameters θ, the task loss term is defined as

$$\mathcal{L}(x, \theta) = -[f_\theta(\mathbf{x}) \log(\mathbf{x}) + (1 - f_\theta(\mathbf{x})) \log(1 - \mathbf{x})]. \quad (1)$$

At the meta-optimization step, we update the model parameters θ considering a meta-optimization loss as the weighted sum of task loss terms evaluated on the episodic training and episodic testing after one step of virtual update. The detailed meta-learning procedures are specified in Algorithm 1.

In comparison to the original MLDG where the sampling of episodic training set and episodic testing set is designed based on different image domains (e.g., cartoon, painting, photo, etc.) for supervised object classification, in this work, we devise three sampling schemes specifically targeting the task of unsupervised reconstruction of data manifold given data scarcity. Figure 1 provides a notional depiction of sample distribution in 2D space which highlights the comparison between the different sampling strategies, where each dot referes to a sample. In each setting, we split the entire data space to meta training set S (light blue) and meta testing set S' (light red). Specifically to simulate a low sampling density region in high dimensional data, we consider samples in a random local neighborhood as the S' which is hold out from model training. Within S, at each episode, we consider uniformly sampled S_{train} and construct S_{test} to include (1) the nearest neighbor sample of each sample in S_{train} (Figure 1, Setting A), (2) a disjoint random batch uniformly sampled from the training data S (Figure 1, Setting B), or (3) a disjoint batch containing a random local neighborhood in S (Figure 1, Setting C). The three settings simulate different distribution shifts to encourage model generalization with increased difficulty. Correspondingly, the model is encouraged to generalize to (1) unseen test samples close to training samples in the Euclidean space in Setting A, (2) unseen test samples from the same training distribution in Setting B, and (3) unseen test samples from a low sampling density region corresponding to a hole or a gap in the training manifold in setting C.

Input: Training Data S; reconstruction loss $\mathcal{L}$
Init: Model parameters θ, Hyperparameters α, β, γ.
for iter **in** iterations **do**
  Sampling: sample disjoint S_{train} and S_{test} from S
  Episodic Train: Compute $\nabla_\theta = \mathcal{L}'_\theta(S_{train}; \theta)$; update $\theta^* = \theta - \alpha\nabla_\theta$
  Episodic Test: Evaluate $\mathcal{L}(S_{test}; \theta^*)$
  Meta Optimization: update $\theta \leftarrow \theta - \gamma \frac{\partial(\mathcal{L}(S_{train};\theta)+\beta\mathcal{L}(S_{test};\theta-\alpha\nabla_\theta))}{\partial\theta}$
end for

Algorithm 1. Meta-learning for manifold reconstruction.

3.2 Topological metrics for manifold reconstruction

To quantitatively evaluate the performance of manifold reconstruction, we perform evaluation including metrics beyond classic reconstruction errors describing instance-level reconstruction quality. Based on various representation of manifolds adopted in topological data analysis, we adopt two sets of metrics to characterize reconstructions based on different types of topological and geometric features of learned manifolds.

For each manifold, we first compute its persistence diagram which characterizes the evolution of topological structures in persistent homology and provides a summarized description of the manifold shape (Cohen-Steiner et al., 2005; Pun et al., 2018; Agami, 2020; Watanabe and Yamana, 2022). We focus on the points in the diagrams describing 0-dimension homology (i.e., connected components) and 1-dimensional homology (i.e., holes) in the manifold. We adopt the Wasserstein distance metric (Mileyko et al., 2011) to compute the similarity between two persistence

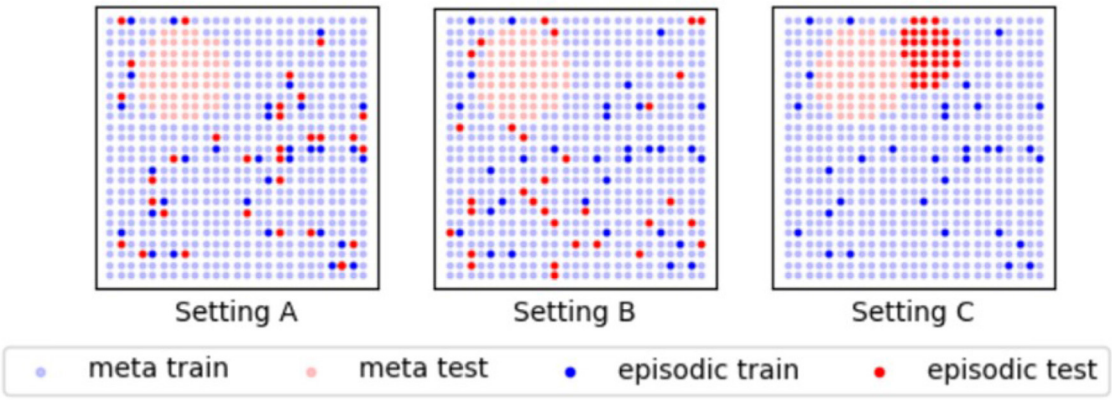


FIGURE 1
Experimental setting A, B and C of MRML with increased difficulties in generalization. Each dot represents a sample when projected into 2D space. A Meta-test set (light red) is held out from Meta-train set (light blue) to simulate random hole in the manifold corresponding low-sampling density region. For each training episode, a random batch of episodic-train set (blue) and a disjoint batch of episodic-test set (red) are sampled from Meta-train set to simulate the data shift to encourage generalization of the model.

diagrams. Considering two persistence diagrams D_1 and D_2, the p-th Wasserstein distance $W_p(D_1, D_2)$ is defined as:

$$W_p(D_1, D_2) = (\inf_{M} \sum_{x \in D_1} ||x - M(x)||_\infty^p)^{\frac{1}{p}} \quad (2)$$

where M denotes all bijection mappings from D_1 to D_2. The Wasserstein distance measures the distance between coupled points from an optimal matching between two diagrams and thus characterizes the similarity of topological features between two manifolds.

In addition to the persistence diagram, we further construct a k-nearest neighbor (KNN) graph for each of the manifolds as a representation characterizing the manifold from the perspective of local geometry at different resolutions. Considering each point in the manifold as a vertex in the graph, we connect vertices based on the Euclidean distance between corresponding samples (Omohundro, 1989). We describe each graph via its binary adjacency matrix and evaluate the similarity between two graphs based on adjacency spectral distance (Wills and Meyer, 2020). Let A_1 and A_2 denote the adjacency matrices of two KNN graphs G_1 and G_2 of size n. An adjacency matrix A is computed such that the (i, j)-th element in the matrix is labeled as 1 if i-th sample is one of the k-nearest neighbors to the j-th sample and is labeled as 0 otherwise. The adjacency spectral distance is computed as

$$S(G_1, G_2) = \sqrt{\sum_{i=1}^{n} |\lambda_i^{A_1} - \lambda_i^{A_2}|^2} \quad (3)$$

with λ^A denoting the eigenvalues of matrix A and $|\cdot|$ computing the magnitude of values.

4 Experiment

4.1 Dataset

4.1.1 LineMOD

To demonstrate manifold reconstruction, we use the LineMOD dataset (Hinterstoisser et al., 2012) which is widely adopted for 6D object pose estimation. The dataset includes 3D object models and RGB-D images along with the ground-truth 6D object poses. There are 15 texture-less objects with discriminative colors, shapes, and sizes. For each of 15 objects, there are 1313 samples at 640x480 resolution which are obtained by rendering object mesh models with surface color and normal from a densely sampled view sphere. For our experiments, we use RGB data rescaled to 64x64. We train autoencoders to learn manifolds corresponding to data of single object class and data of all 15 object classes.

4.2 Experimental setting

To demonstrate manifold reconstruction with data scarcity, we perform multiple baseline experiments using vanilla autoencoder (AE) and recent approaches emphasizing preservation of geometry (Lee Y. et al., 2021; Duque et al., 2022). We perform experiments using MRML under three episodic sampling schemes as illustrated in Figure 1. We first split the data into (meta-) training and (meta-) testing set by randomly selecting a sample and holding out nearest k samples from the neighbor in Euclidean space. By holding out this random cluster away from training, we simulate a training manifold with a low density region corresponding to unseen latent variations. For all MRML experiments, we use the same encoder and decoder architectures. The encoder consists of 4 convolutional layers followed by 3 fully connected layers and the decoder consists of 3 fully connected layers followed by 4

deconvolutional layers. The latent space is set to 10 dimension. We use the Adam optimizer (Kingma and Ba, 2015) and batch size 64 for training all the models. We performed two sets of experiments, one for learning the manifold from images of a single object and one for learning the manifold from images of all 15 object classes. For single object experiment, we hold out a local neighborhood containing 100 images. We perform training with 1,000 iterations at learning rate 10^{-3}. For experiments on multiple object classes, we hold out a local neighborhood containing 1,000 images. The hold out data are from the same object class. We use the same batch size, learning rate, episodic step size and episodic testing weight as used in the single object experiments. We set the episodic training step size α and weight on episodic testing loss β to be 10^{-7} and 10^{-3} for all three settings. For vanilla AE and NRAE experiments, we use the same encoder and decoder architecture, optimizer setting and learning rate as used in MRML experiments. For GRAE, we followed the reported implementation. The hyperparameters are selected based on the convergence of reconstruction accuracy.

4.3 Results and discussion

We compare MRML under three episodic training settings against the baselines. To qualitatively compare the learned manifolds, we show 3-dimensional t-SNE visualizations (van der Maaten and Hinton, 2008) of the manifolds in the data space, the latent space of the encoder and the reconstruction space of the decoder in Figure 2. We observe that for baselines and MRML, the contour of the manifold is largely preserved when projected to the latent space and the reconstructed space. Specifically focusing on the hold-out test samples unseen at training stage (red in Figure 2), we notice that in comparison to the ones learned via baselines, manifold representations learned via MRML produce more uniformly distributed samples in both the latent space and the reconstructed space, which better matches the original data manifold.

For qualitative comparison, in addition to sample-level reconstruction accuracy measured via mean square error (MSE), we measure the manifold-level reconstruction via the topological distance between learned manifold of the reconstructed images and the data manifold based on both persistence diagrams and KNN graphs. Table 1 shows the quantitative comparison for experiments on learning the manifold of a single object class. For comparison based on MSE, we observe that most methods have comparable sample-level reconstruction accuracy on average. While GRAE shows an edge in overall reconstruction accuracy but has considerably higher error when evaluated on the hold out test data. We would like to note that, while selected baselines were proposed to address local geometry, our problem setting pose further challenge in generalization as we consider holdout data specifically corresponding to holes/gaps in training manifolds as visualized in Figure 2, in opposed to, e.g., unseen samples following the same training distribution. Comparing performance over three sets of metrics, it is observed that for approaches with similar averaged reconstruction accuracy, its capability in preserving topological or geometric features can still vary. This emphasize the need for including topological and geometric metrics for examining representation learning to support the use of the representation in topological or geometric analysis. For comparison based on Wasserstein distance between persistence diagrams, we observe that MRML methods consistently achieve improved or comparative performance against baseline approaches over different orders of Wasserstein distance (p) and considering persistent homology features at different dimensions, especially for MRML setting B and setting C. This quantitative improvement is aligned with qualitative observation as shown in Figure 2. This improvement is consistent when we investigate manifold reconstruction of the local neighborhood that is held out from the training as well as the manifold reconstruction for the complete dataset including both training and hold-out testing samples. We also note that the improvement on reconstructing the manifold is demonstrated even against baselines showing higher average accuracy at sample level (e.g., comparing MRML against NRAE on holdout set and against GRAE on complete data). Specifically, MRML with best performing sampling strategy reduces the second order Wasserstein distance by a factor of 14.44% considering the hold-out neighborhood and by a factor of 4.44% considering the complete manifold against the best performing baseline.

For comparison based on adjacency spectral distance between KNN graphs at different resolutions, we observe that when considering KNN graphs at finer granularity (k = 5), the comparison better correlates with the comparison based on averaged accuracy between pairwise samples. While proposed MRML methods, especially setting B and setting C on holdout test samples, demonstrate better match of KNN graphs at a coarser resolution (k = 20) which suggests the alignment on the intrinsic shape of the manifold. In this case, MRML with best performing sampling strategy demonstrates a reduction in adjacency spectral distance by a factor of 13.82% against the best performing baseline.

We also observe the improvement with proposed MRML approaches in both experiments of single object class where a manifold containing a single connected piece is considered and experiments of multiple object classes where the manifold of increased complexity contains multiple pieces, as shown in Table 2. Note that for the multi-class experiments, due to the computation cost, the comparison is only reported on the large holdout test set corresponding to the missing gap/hole in the manifolds. In this case, our proposed strategies again demonstrate consistent improvement with respect to topological metrics based on persistence diagrams and geometric metrics based on KNN graphs, comparing to baselines at similar or inferior generalization accuracy at sample level. Specifically, considering the hold-out neighborhood, MRML with best performing sampling strategy demonstrates a reduction in the second order Wasserstein distance by a factor of 9.62% against the best performing baseline.

Finally, we compare the performance of MRML under different episodic sampling schemes. We observe that the comparison between MRML based strategies against baselines are mostly consistent. For single object experiments, we observe that Setting B and C yield relatively better generalization on the hold-out test samples and better reconstruction on the overall manifold. For experiments on multiple object classes where the overall manifold has higher complexity, setting A and C yield relatively better generalization. This comparison validate the use of sampling scheme specifically simulating a generalization to missing gaps/holes in the underlying manifold under setting C.

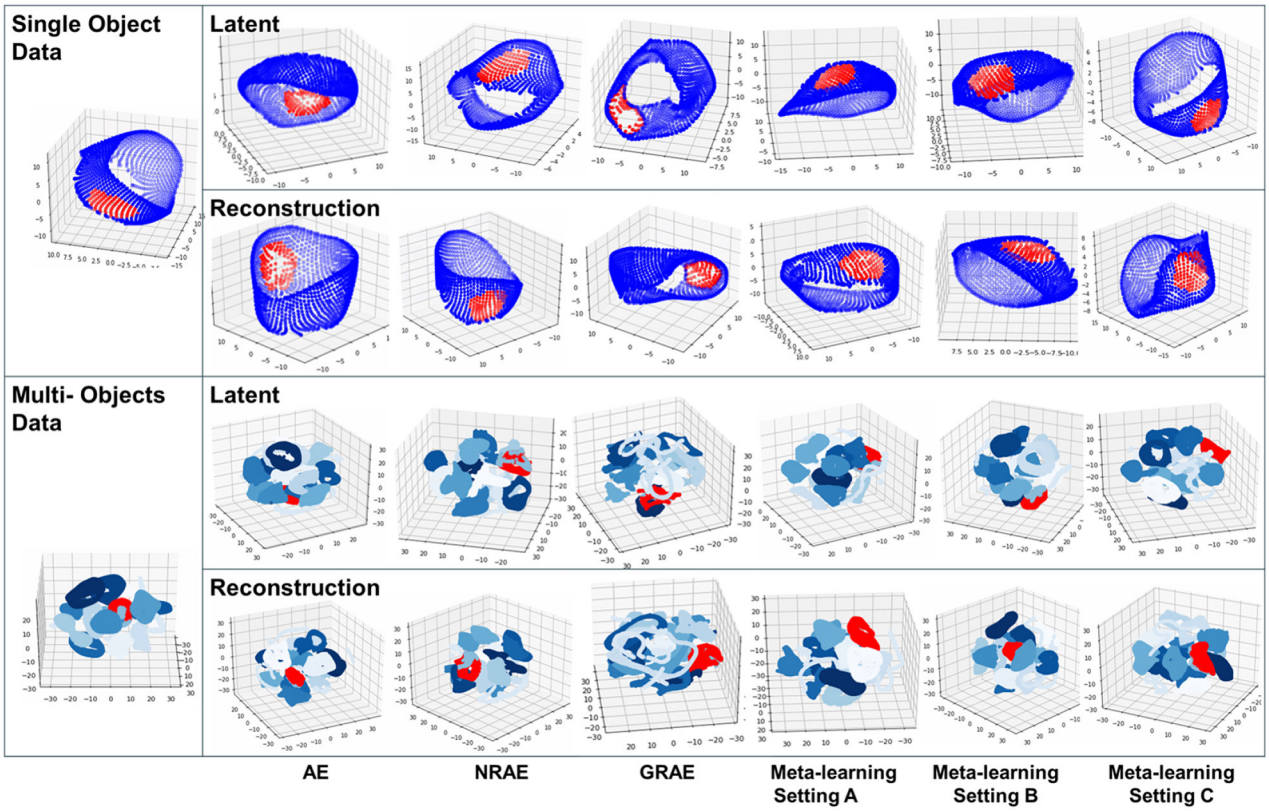


FIGURE 2
TSNE visualization of data and learned manifolds using LineMOD data of single object class **(top panel)** and all 15 object classes **(bottom panel)**. In single object manifolds, blue represents meta-training samples and red represents hold-out meta-testing samples. In multiple object manifolds, blue with different shades represents meta-training samples of different objects and red represents hold-out meta-testing samples.

TABLE 1 Experiments on LineMOD dataset with single object class.

Test samples	Algorithms	MSE	Persistence diagram						KNN Graph	
			Wasserstein p=1			Wasserstein p=2			Spectral distance	
			Dim 1	Dim 2	Total	Dim 1	Dim 2	Total	$k = 5$	$k = 20$
Holdout	AE	4.03e-3	158.43	22.60	181.03	15.94	2.73	18.67	2.98	8.90
	NRAE	**3.55e-3**	189.51	25.20	214.70	18.99	2.85	21.85	**2.49**	9.04
	GRAE	6.85e-3	140.52	**14.37**	154.89	14.45	**2.03**	16.48	3.47	9.41
	Meta A (Ours)	4.11e-3	145.92	17.91	163.83	14.69	2.37	17.06	3.76	8.66
	Meta B (Ours)	3.95e-3	122.49	16.94	139.44	12.35	2.26	14.61	3.26	**7.67**
	Meta C (Ours)	3.82e-3	**116.26**	17.23	**133.49**	**11.72**	2.38	**14.10**	2.85	8.36
All	AE	2.05e-3	2,053.00	229.08	2,282.08	59.64	8.60	68.24	10.01	28.31
	NRAE	1.92e-3	2,541.94	279.99	2,821.93	71.40	10.43	81.83	**8.47**	30.58
	GRAE	**1.25e-3**	1,952.196	214.83	2,167.03	57.85	8.16	66.01	9.90	**28.06**
	Meta A (Ours)	2.15e-3	2,139.59	213.28	2,352.87	62.28	9.23	71.51	10.24	29.37
	Meta B (Ours)	2.05e-3	**1873.38**	**185.59**	**2058.96**	**55.40**	**7.67**	**63.08**	10.44	30.99
	Meta C (Ours)	1.96e-3	1,902.57	194.37	2,096.94	56.07	8.24	64.32	9.24	32.70

Evaluation based on topological distance between data and reconstructed manifolds and mean squared error (MSE) between data and reconstructed samples. Evaluation on manifolds of hold-out testing data (top) and manifolds of complete dataset including both training and testing data (bottom). The **best** and second best performing approaches are highlighted.

TABLE 2 Experiments on LineMOD dataset with multiple (15) object classes.

Test Samples	Algorithms	MSE	Persistence diagram						KNN Graph	
			Wasserstein p=1			Wasserstein p=2			Spectral distance	
			Dim 1	Dim 2	Total	Dim 1	Dim 2	Total	$k = 5$	$k = 20$
Holdout	AE	3.88e-4	154.88	19.92	174.81	5.87	0.90	6.76	10.61	28.45
	NRAE	7.91e-4	305.03	30.83	335.85	9.22	1.40	10.62	10.51	30.07
	GRAE	9.48e-4	290.33	30.52	320.85	9.00	1.41	10.40	13.24	29.37
	Meta A (Ours)	**3.18e-4**	134.66	**15.95**	150.60	5.30	0.81	6.11	10.68	28.65
	Meta B (Ours)	3.47e-4	145.13	17.26	162.39	5.73	**0.77**	6.50	10.43	28.49
	Meta C (Ours)	3.89e-4	141.15	17.03	158.17	5.49	0.81	6.30	**9.96**	**28.08**

Evaluation based on topological distance between data and reconstructed manifolds and mean squared error (MSE) between data and reconstructed samples. Evaluation is performed on the manifold of hold-out test set only due to computation cost. The **best** and second best performing approaches are highlighted.

5 Conclusion

We propose manifold representation meta-learning to address data scarcity in manifold reconstruction. Our framework is based on model agnostic meta-learning, a state-of-the-art learning paradigm that utilize episodic training to achieve better performance given domain shifts. We specifically adapt the framework to address the challenge task of unsupervised manifold representation learning considering manifold regions with low sampling densities. We adopt two sets of topological and geometric metrics for quantitative comparison between data and model reconstruction at manifold level. The metrics are computed based on persistence diagrams characterizing homology features in the manifold and KNN graphs characterizing relative proximity of samples in the Euclidean space. We demonstrate that, in comparison to state-of-the-art baselines, our MRML can better preserve topological and geometric structures and better match the data manifold, especially for regions with low sampling densities. In our future work, we plan to integrate topological and geometric measures with model training to better capture the underlying manifold especially for real-world data with increasing complexity in shape and increasing noise level in the data samples.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YG: Writing – original draft, Writing – review & editing. JY: Writing – review & editing. RL: Writing – review & editing. XL: Writing – review & editing. CC: Writing – review & editing, Supervision. AD: Writing – review & editing. YY: Writing – review & editing, Supervision.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR0011-22-9-0077.

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.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Author disclaimer

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

Agami, S. (2020). Comparison of persistence diagrams. *Commun. Stat. Simulat. Comput.* 52, 1948–1961. doi: 10.1080/03610918.2021.1894335

Aktas, M. E., Akbas, E., and Fatmaoui, A. E. (2019). Persistence homology of networks: methods and applications. *Appl. Netw. Sci.* 4, 1–28. doi: 10.1007/s41109-019-0179-3

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., et al. (2016). "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 29.

Bank, D., Koenigstein, N., and Giryes, R. (2020). Autoencoders. *arXiv preprint arXiv:2003.05991*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: a review and new perspectives. *IEEE Trans. Patt. Analy. Mach. Intell.* 35, 1798–1828. doi: 10.1109/TPAMI.2013.50

Cohen-Steiner, D., Edelsbrunner, H., and Harer, J. (2005). "Stability of persistence diagrams," in *Proceedings of the Twenty-First Annual Symposium on Computational Geometry*, 263–271. doi: 10.1145/1064092.1064133

Duque, A. F., Morin, S., Wolf, G., and Moon, K. R. (2022). Geometry regularized autoencoders. *IEEE Trans. Patt. Analy. Mach. Intell.* 45, 7381–7394. doi: 10.1109/TPAMI.2022.3222104

Edelsbrunner, H., and Harer, J. (2008). Persistent homology – a survey. *Contemp. Mathem.* 453, 257–282. doi: 10.1090/conm/453/08802

Finn, C., Abbeel, P., and Levine, S. (2017). "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning* (PMLR), 1126–1135.

Gropp, A., Atzmon, M., and Lipman, Y. (2020). Isometric autoencoders. *arXiv preprint arXiv:2006.09289*.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., et al. (2012). "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Computer Vision-ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11* (Springer Berlin Heidelberg), 548–562. doi: 10.1007/978-3-642-37331-2_42

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2022). Meta-learning in neural networks: a survey. *IEEE Trans. Patt. Analy. Mach. Intell.* 44, 5149–5169. doi: 10.1109/TPAMI.2021.3079209

Hsu, K., Levine, S., and Finnl, C. (2019). Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*.

Kingma, D. P., and Ba, J. (2015). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, D. B., Min, D., Lee, S., and Hwang, S. J. (2021). "Meta-GMVAE: mixture of gaussian vaes for unsupervised meta-learning," in *International Conference on Learning Representations*.

Lee, Y., Kwon, H., and Park, F. C. (2021). "Neighborhood reconstructing autoencoders," in *Advances in Neural Information Processing Systems*, 536–546.

Lee, Y., Yoon, S., Son, M., and Park, F. C. (2022). "Regularized autoencoders for isometric representation learning," in *ICLR*.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. (2018). "Learning to generalize: meta-learning for domain generalization," in *Proceedings of the AAAI Conference on Artificial Intelligence*. doi: 10.1609/aaai.v32i1.11596

Li, D., Zhang, J., Yang, Y., Liu, C., Song, Y.-Z., and Hospedales, T. M. (2019). "Episodic training for domain generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1446–1455. doi: 10.1109/ICCV.2019.00153

Liou, C.-Y., Cheng, W.-C., Liou, J.-W., and Liou, D.-R. (2014). Autoencoder for words. *Neurocomputing* 139, 84–96. doi: 10.1016/j.neucom.2013.09.055

Liu, Q., Chen, C., Qin, J., Dou, Q., and Heng, P.-A. (2021). "FEDDG: federated domain generalization on medical image segmentation via episodic learning in continuous frequency space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1013–1023. doi: 10.1109/CVPR46437.2021.00107

Mileyko, Y., Mukherjee, S., and Harer, J. (2011). Probability measures on the space of persistence diagrams. *Inverse Problems* 27:124007. doi: 10.1088/0266-5611/27/12/124007

Moor, M., Horn, M., Rieck, B., and Borgwardt, K. (2020). "Topological autoencoders," in *International Conference on Machine Learning* (PMLR), 7045–7054.

Nazari, P., Damrich, S., and Hamprecht, F. A. (2023). Geometric autoencoders-what you see is what you decode. *arXiv preprint arXiv:2306.17638*.

Omohundro, S. (1989). *Five balltree construction algorithms*. International Computer Science Institute Technical Report.

Pun, C. S., Xia, K., and Lee, S. X. (2018). Persistent-homology-based machine learning and its applications - a survey. *arXiv preprint arXiv:1811.00252*.

Schönenberger, S. T., Varava, A., Polianskii, V., Chung, J. J., Kragic, D., and Siegwart, R. (2020). "Witness autoencoder: shaping the latent space with witness complexes," in *NeurIPS 2020 Workshop TDA and Beyond*.

Schonsheck, S., Chen, J., and Lai, R. (2019). Chart auto-encoders for manifold structured data. *arXiv preprint arXiv:1912.10094*.

Snell, J., Swersky, K., and Zemel, R. S. (2017). "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 30.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 2319–2323. doi: 10.1126/science.290.5500.2319

van der Maaten, L., and Hinton, G. (2008). Visualizing data using t-SNE. *JMLR* 9, 2579–2605.

Watanabe, S., and Yamana, H. (2022). Topological measurement of deep neural networks using persistent homology. *Ann. Mathem. Artif. Intell.* 90, 75–92. doi: 10.1007/s10472-021-09761-3

Wills, P., and Meyer, F. G. (2020). Metrics for graph comparison: a practitioner's guide. *PLoS ONE* 15:e0228728. doi: 10.1371/journal.pone.0228728

Wu, M., Choi, K., Goodman, N., and Ermon, S. (2020). "Meta-amortized variational inference and learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 6404–6412. doi: 10.1609/aaai.v34i04.6111

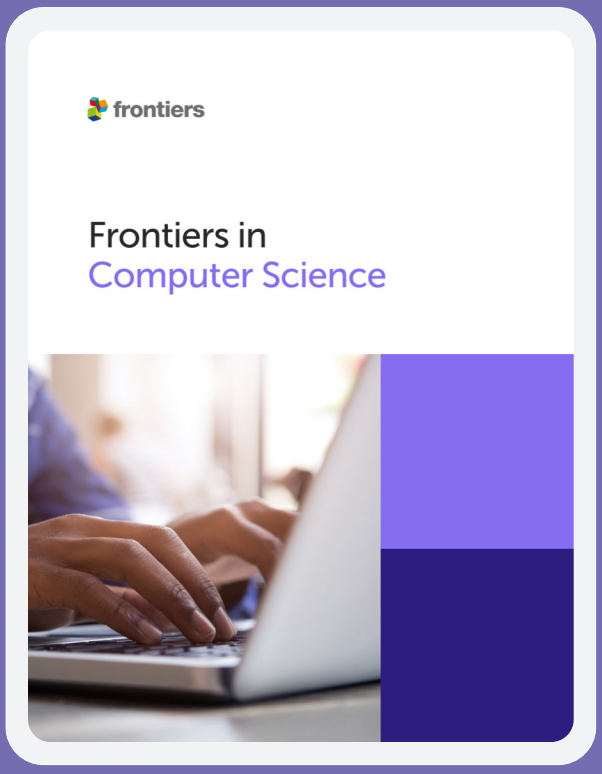
Frontiers in Computer Science

Explores fundamental and applied computer science to advance our understanding of the digital era

An innovative journal that fosters interdisciplinary research within computational sciences and explores the application of computer science in other research domains.

Discover the latest Research Topics

See more →



Frontiers

Avenue du Tribunal-Fédéral 34
1005 Lausanne, Switzerland
frontiersin.org

Contact us

+41 (0)21 510 17 00
frontiersin.org/about/contact