On incorporating inductive biases into deep neural networks

Sameera Ramasinghe

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

March 2022

Except where otherwise indicated, this thesis is my own original work.

Sameera Ramasinghe
25 March 2022

To the eternal silence of the timeless and boundless universe, where all become one, and one becomes all.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisors Dr. Salman Khan, Prof. Nick Barnes, Prof. Stephen Gould, and Dr. Lars Petersson, for their continuous support of my Ph.D. research. Without their invaluable knowledge in computer vision, feedback, and guidance, I would not have been able to complete my studies.

I would also like to thank my colleagues and collaborators, specifically Moshiur Farazi, Kanchana Ranasinghe, and Ali Cheraghian. Their constant help, valuable suggestions, and friendship made my Ph.D. journey a whole lot easier. I will dearly cherish all the good times we had at Data61.

I am grateful for all the administration staff members at both ANU and Data61, who were immensely helpful in various administrative tasks from the beginning of my studies. Also, thank you for being extremely responsive to all the emails, requests, and irritative questions.

A special thanks go to all the members of the Ridma society, especially Mr. Sanath Ranjitha, for guiding me throughout my life and academics. They always pushed me to do beyond my best, and I cannot express my gratitude deeply enough for all the vision, encouragement, and support. Our countless discussions on academic matters, life, and literature strived me to be a better person in various aspects, and I will continue to try to do so in the future.

Most importantly, I am forever indebted to my mother, father, and brother, who were the pillars of my success. Thank you for always believing in me and encouraging me to follow my dreams, and supporting me throughout my life.

Last but not least, I owe all my success to my lovely wife, Gayathri Warnakulahewa, who has shown me never-ending support throughout my life. Since the day I met her, she was the source of my motivation through various ups and downs of my life. Being a wife of a Ph.D. student is not easy, yet you supported me unconditionally without a single complaint. Thank you for all the dedication and sacrifices you made for me.

Abstract

A machine learning (ML) algorithm can be interpreted as a system that learns to capture patterns in data distributions. Before the modern *deep learning era*, emulating the human brain, the use of structured representations and strong inductive bias have been prevalent in building ML models, partly due to the expensive computational resources and the limited availability of data. On the contrary, armed with increasingly cheaper hardware and abundant data, deep learning has made unprecedented progress during the past decade, showcasing incredible performance on a diverse set of ML tasks. In contrast to *classical ML* models, the latter seeks to minimize structured representations and inductive bias when learning, implicitly favoring the flexibility of learning over manual intervention. Despite the impressive performance, attention is being drawn towards enhancing the (relatively) weaker areas of deep models such as learning with limited resources, robustness, minimal overhead to realize simple relationships, and ability to generalize the learned representations beyond the training conditions, which were (arguably) the forte of classical ML. Consequently, a recent hybrid trend is surfacing that aims to blend structured representations and substantial inductive bias into deep models, with the hope of improving them.

Based on the above motivation, this thesis investigates methods to improve the performance of deep models using inductive bias and structured representations across multiple problem domains. To this end, we inject a priori knowledge into deep models in the form of enhanced feature extraction techniques, geometrical priors, engineered features, and optimization constraints. Especially, we show that by leveraging the prior knowledge about the task in hand and the structure of data, the performance of deep learning models can be significantly elevated.

We begin by exploring equivariant representation learning. In general, the real-world observations are prone to fundamental transformations (e.g., translation, rotation), and deep models typically demand expensive data-augmentations and a high number of filters to tackle such variance. In comparison, carefully designed equivariant filters possess this ability by nature. Henceforth, we propose a novel *volumetric convolution* operation that can convolve arbitrary functions in the unit-ball ($\mathbb{B}^3$) while preserving rotational equivariance by projecting the input data onto the Zernike basis. We derive theoretical formulae to perform the convolution entirely in the function space, which paves the way to efficient implementation and allows it to be used as a differentiable and easily pluggable layer in deep networks. Similarly, we derive a novel set of orthogonal basis functions that are complete in $\mathbb{B}^3$, where the derived basis functions comprise the properties required to achieve both rotational and translational equivariant convolutions. We conduct extensive experiments and show that our formulations can be used to construct significantly cheaper ML models.

Next, we study generative modeling of 3D objects and propose a principled ap-

proach to synthesize 3D point-clouds in the spectral-domain by obtaining a structured representation of 3D points as functions on the unit sphere ($\mathbb{S}^2$). Using the prior knowledge about the spectral moments and the output data manifold, we design an architecture that can maximally utilize the information in the inputs and generate high-resolution point-clouds with minimal computational overhead.

Then, we direct our attention to stochastic deep generative models. Conditioned on an input variable, real-world scenarios carry more than one possible solution for a given generation task. This aspect is less focused in the current deep generative models, restraining their full potential towards generation of diverse outputs. We introduce a novel framework for conditional generation in multimodal spaces that uses latent variables to model generalizable learning patterns while minimizing a family of regression cost functions. Our *generic* model can outperform highly engineered pipelines tailored using domain expertise on various tasks while generating diverse outputs. We further conduct theoretical analysis on the conditional GANs (cGAN) and identify several critical problems associated with the existing training approaches. We demonstrate—both theoretically and empirically—that by imposing constraints to encourage a homeomorphism between the latent and output manifolds, one can dramatically improve the realism and the diversity of the cGAN outputs.

Finally, we propose a framework to build normalizing flows (NF) based on increasing triangular maps and Bernstein-type polynomials. Compared to the existing NF approaches, our framework consists of favorable characteristics for fusing inductive bias within the model i.e., theoretical upper bounds for the approximation error, robustness, higher interpretability, suitability for compactly supported densities, and the ability to employ higher degree polynomials without training instability. Further, owing to the known mathematical properties of the Bernstein-type polynomials, we can directly control the bounds of the outputs of our model. Most importantly, we present a constructive universality proof, which permits us to analytically derive the optimal model coefficients for known transformations without training.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Machine learning has shown tremendous success in various tasks during the past decade, primarily due to the advancement of *deep learning*. Modern deep networks have even exceeded human performance in many domains spanning image classification [Krizhevsky et al., 2012; Szegedy et al., 2017; Brock et al., 2021], video analysis [Kalfaoglu et al., 2020; Bertasius et al., 2021], natural language processing [Sutskever et al., 2014; Bahdanau et al., 2014], speech recognition [Chung et al., 2019; Zeinali et al., 2019], and game play [Silver et al., 2016; Moravčík et al., 2017]. Nevertheless, there is still a fundamental difference between an end-to-end learned deep network and human intelligence: humans have a remarkable ability to generalize beyond past experiences and learn from limited data, which remains challenging for cutting-edge ML models.

Deep models usually deter explicit structured representations and computational assumptions, seeking to minimize hand-designed choices to model inductive bias. This can also be seen as a trade-off between inductive bias and the flexibility of learning. This trade-off has paid well in the last decade, supported by the high-performing hardware and abundant data [Krizhevsky et al., 2012; Bahdanau et al., 2014; Moravčík et al., 2017]. This approach is seemingly counter-intuitive, as structured representations and inductive bias help models discover patterns and semantic information in data easily. However, having an extensive amount of data allows the deep models to discover structured information in the data implicitly [Welling, 2019]. Hence, the aforementioned trade-off is suitable as long as a sufficient amount of data is available. However, datasets are empirical estimates of the real-world distributions and may comprise strong biases. Hence, there is a strong possibility that (naively trained) models on these datasets would not function well when applied to noisy real-world tasks. Further, the minimal inductive bias hampers the ability of the deep networks to respond adequately to out-of-distribution data.

A natural solution for this problem is to encourage the model to capture the underlying properties of data that generalize the model beyond the training set. For example, assume a 3D object recognition task where the objects are not aligned with respect to a canonical coordinate system. Intuitively, we have the prior knowledge that the learned function should be symmetric against the 3D rotations and translations.

However, since the training dataset is finite, it cannot represent all possible rotations of all the objects. Given enough data, the network may incorporate some rotation invariance, but, it is not likely that the network will systematically discover that the learned function should be invariant to all the rotations, unless the invariance is explicitly built into the layers of the model. In contrast, conscientiously designed equivariant filters can help the model to be trained with limited data and generalize to unseen rotations in the real-world. We will revisit inductive bias more formally in the next sub-section.

1.2 Inductive bias

Learning generic patterns and rules using a set of observations is called *inductive reasoning*. Any ML model should be able to perform inductive reasoning in order to achieve sensible results over a training set. However, many optimal models may exist that are equally consistent with an available set of training data. Inductive bias is the process of preferring a certain set of generalizations in a model, that may work better on the test set than other possible solutions. This concept is more formally presented in the *no-free-lunch* theorem of machine learning [Wolpert, 1996], which basically says given a dataset and a loss function, there is no absolutely general-purpose model. That is, any model will generalize better on a particular test distribution and perform worse on others. Therefore, the important problem becomes identifying inductive biases that better coincide with the human perception and understanding of the world.

Since inductive bias is essential in any ML model, deep models implicitly or explicitly incorporate inductive bias in the form of regularization objectives [Srivastava et al., 2014; Kukačka et al., 2017; Zhang et al., 2017a], weight sharing, design constraints [Yu et al., 2017; Long et al., 2015; Dumoulin and Visin, 2016; Huang et al., 2017a], equivariance [Defferrard et al., 2020; Cohen et al., 2018b; Ramasinghe et al., 2019c; Esteves et al., 2018a], or prior distributions in probabilistic models [Kingma and Welling, 2013; Rezende and Mohamed, 2015; Ramasinghe et al., 2021]. From another perspective, inductive bias can not only improve the performance but also ease the optimization [Battaglia et al., 2018]. This is intuitive, since adding inductive bias to a model can be considered as artificially forcing the model to learn relationships in data without training. Similarly, inductive bias can also be treated as *training data in disguise* [Goyal and Bengio, 2020] and one can compensate for the lack of inductive bias with more data [Welling, 2019]. This is one primary motivation of deep learning: the user-specified inductive bias in a model can be minimized if enormous datasets are available. However, to capture certain relationships without sufficient inductive bias, the model may need an infeasible number of training samples, which may not be practical in certain tasks (e.g., 3D reconstruction).

1.3 Machine learning vs human intelligence

In contrast to ML models, humans can learn with limited data and generalize the learned representations to new inputs. This remarkable ability of the human brain has inspired ML/AI researchers to study how AI should relate to human intelligence since the earliest days of AI research [TURING, 1950; Newell and Simon, 1961; Newell et al., 1972; Bobrow and Winograd, 1977; Hayes-Roth and Hayes-Roth, 1979; Schank, 1972; Fukushima and Miyake, 1982; Holyoak, 1987; Schmidhuber, 2015; Lake et al., 2017; Battaglia et al., 2018; Goyal and Bengio, 2020]. At a high-level, many of these works posit that human knowledge representation is built upon sub-symbolic reasoning, where specialized units work in parallel to tackle a complex task. Particularly, they emphasize two fundamental attributes of human intelligence: a) *intuitive physics* and b) *combinatorial generalization*. We shall discuss these attributes and their motivation to our work in the next sub-sections.

1.3.1 Intuitive physics

A key signature of human intelligence in understanding the external world is *intuitive physics* [Wellman and Gelman, 1992]. Infants have an abstract understanding of physics concepts that allow them to anticipate certain outcomes beyond past experiences. For instance, infants can discount unlikely object trajectories beyond the paucity of observations [Spelke, 1990; Kosslyn and Osherson, 1995]. Similarly, they can recognize objects seen from entirely new angles by utilizing the understanding of the symmetry of the objects in the physical world [Baillargeon, 2004; Baillargeon et al., 2009]. Such understandings directly correspond to physics concepts such as inertia, support, collisions, and geometry. Fig. 1.1 depicts a rough timeline of early conceptual acquisition in infants.

Classical ML approaches have tried to fuse these physics concepts into learnable models. The seminal work by Valiant [1984] and Hopfield [1982] were perhaps among the earliest attempts on integrating natural rules of the physical world into learnable models. Notably, Valiant [1984] embarked precise statistical learning in AI, and Hopfield [1982] blended the rich concepts from spin glass theory to neural networks models, which was later extended by Amit et al. [1985]. Similarly, Gardner [1988] applied the *replica trick* [1] to neural networks to solve the problem of storing a given set of p random patterns as N-bit Ising spin configurations. Their work allowed analytical error computation and learning curves of (specific classes of) neural networks as a function of the number of training samples [Györgyi and Tishby, 1990; Seung et al., 1992], that can be generalized even to modern machine learning models [Zdeborová and Krzakala, 2016]. These seminal works inspired the integration of mathematical and physics concepts into classical ML models, few examples being the stochastic block model study for detecting clusters in sparse networks [Fortunato, 2010; Decelle et al., 2011], spectral algorithms for sparse data, [Krzakala et al., 2013], message

[1] The replica trick is generally used in statistical physics for computations that involve analytic functions.

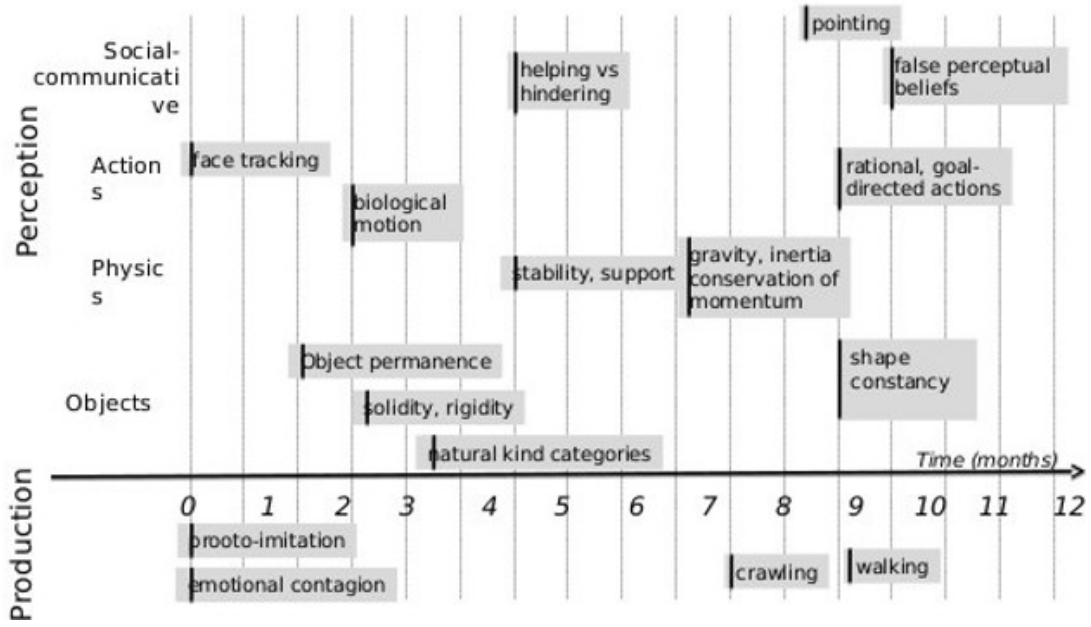


Figure 1.1: Early conceptual acquisition in infants (Dupoux [2018]).

passing [Bolthausen, 2014; Javanmard and Montanari, 2013; Rangan et al., 2012], and the work on restricted boltzman machines (RBM) [Smolensky, 1986; Hinton, 2002; Gabrié et al., 2015; Tramel et al., 2018].

On the other hand, with the birth of deep learning, deep networks have been employed across diverse disciplines that require exploiting natural laws of nature, including applied physics [Baldi et al., 2016; Ibarra-Berastegi et al., 2015], computational biology [Alipanahi et al., 2015], material science [Kauwe et al., 2018; Ryan et al., 2018; Wei et al., 2018], aquatic science [Jia et al., 2020], and climate analysis [Nowack et al., 2018; Schmidt et al., 2019]. However, most of these works follow the *end-to-end* philosophy of deep learning, which lessens the constraints and inductive bias. Consequently, these models suffer from the need for extensive labeled data and poor generalization to unseen conditions [Willard et al., 2020]. For instance, recent work by Lerer et al. [2016] used a deep network (PhysNet) to predict the stability of block towers from images. Remarkably, their model exceeded human performance on both real and synthetic images on this task. However, the model demonstrated two limitations: a) the model needs a humongous amount of data to be properly trained ($\sim 200,000$ scenes per single task) and b) the model does not generalize well to more complex scenes (block towers with a higher number of blocks). In contrast, ML models designed to strictly follow physics concepts, such as modern animation engines [Lerer et al., 2016; Battaglia et al., 2013; Gerstenberg et al., 2015] and deep implicit networks [Gould et al., 2019, 2021; Johnson et al., 2016b; Amos and Kolter, 2017; Agrawal et al., 2019] have shown much better generalizations in alien conditions (see section 9.2.1 for a more comprehensive discussion on deep implicit layers). This

makes its clear that even when trained with an abundant amount of data, the model might actually not learn the underlying physical rules or semantic concepts without properly induced inductive biases.

The models proposed in chapters 3, 4, 5 and 6 follow this philosophy. We use equivariant representation learning techniques that are explicitly designed to be symmetric against basic transformation groups, use known polynomial types to construct flow-models, and use geometrical priors and constraints to improve generative models. Chapter 3 proposes a convolutional network comprising of layers that can learn rotational equivariant representations. Instead of performing the convolution in the Euclidean space, our model operates in the unit ball, making it easier to achieve rotational equivariance. Although performing convolution in Euclidean geometries is fairly straightforward, its extension to other topological spaces—such as a sphere ($\mathbb{S}^2$) or a unit ball ($\mathbb{B}^3$)—entails unique challenges. We introduce a novel *volumetric convolution* operation that can effectively model and convolve arbitrary functions in $\mathbb{B}^3$. We develop a theoretical framework for volumetric convolution based on Zernike polynomials and efficiently implement it as a differentiable and easily pluggable layer in deep networks. By construction, our formulation leads to the derivation of a novel formula to measure the symmetry of a function in $\mathbb{B}^3$ around an arbitrary axis, which is useful in function analysis tasks. We demonstrate the efficacy of the proposed volumetric convolution operation on a viable use case, i.e., 3D object recognition.

Chapter 4 further extends the aforementioned work to achieve both rotational and translation equivariance while simultaneously reducing the impact of the redundancy and the irregularity of 3D point clouds. Most models used in shape analysis directly learn feature representations on 3D point clouds. We argue that 3D point clouds are highly redundant and hold irregular (permutation-invariant) structures, making it difficult to efficiently achieve inter-class discrimination. Chapter 4 proposes a two-pronged solution to this problem that is seamlessly integrated into a single blended convolution and synthesis layer. This fully differentiable layer performs two critical tasks in succession. The first step projects the input 3D point clouds into a latent 3D space to synthesize a highly compact and inter-class discriminative point cloud representation. Since 3D point clouds do not follow an Euclidean topology, standard 2/3D convolutional neural networks offer limited representation capability. Therefore, in the second step, we propose a novel 3D convolution operator functioning inside $\mathbb{B}^3$ to extract useful volumetric features. We derive formulae to achieve both translation and rotation of our novel convolution kernels. Finally, we present an extremely light-weight, end-to-end architecture that achieves compelling results on 3D shape recognition and retrieval using the proposed techniques. Most importantly, the operations proposed in chapter 3 and 4 allows the construction of significantly cheaper deep networks that are robust to noisy data, compared to deep models that do not utilize explicit equivariant representation learning.

Moreover, chapter 5 studies how to improve the performance of conditional GANs (cGAN) using geometrical priors. cGANs, in their rudimentary form, suffer from critical drawbacks such as the lack of diversity in generated outputs and distortion between the latent and output manifolds. Although efforts have been made to

improve results, they can suffer from unpleasant side-effects such as the topology mismatch between latent and output spaces. In contrast, we tackle this problem from a geometrical perspective and propose a novel training mechanism that increases both the diversity and the visual quality of a vanilla cGAN, by systematically encouraging a bi-lipschitz mapping between the latent and the output manifolds. We validate the efficacy of our solution on a baseline cGAN (*i.e.*, Pix2Pix [Isola et al., 2017]) with no diversity, and show that by only modifying its training mechanism (*i.e.*, with our proposed Pix2Pix-Geo), one can achieve more diverse and realistic outputs on a broad set of image-to-image translation tasks.

In chapter 6, we focus on Normalizing Flows (NF). NFs are a class of generative models that allows exact density evaluation and sampling. We propose a novel framework to construct NFs based on increasing triangular maps and Bernstein-type polynomials. Due to the known properties of Bernstein-type polynomials, our method allows us to induce significant inductive bias into the model and manipulate the model more, compared to the existing (universal) NF frameworks. For instance, our model is highly interpretable, as we can analyze the model end-to-end by examining the learned Bernstein coefficients. Utilizing the same properties, we can analytically derive the upper-bound of the approximation error and directly control the bounds of the output densities. Further, we theoretically show that our model is the most robust NF framework among the other polynomial and spline-based flow-models against noisy data. Moreover, we present a constructive universality proof, which yields analytic expressions of the approximations for known transformations.

1.3.2 Combinatorial generalization

A crucial factor that helps humans to generalize knowledge beyond the experience is called *combinatorial generalization*, which attributes to the human brain's capacity to link learned representations in limitless ways [Chomsky, 2014]. Thus, the human brain can realize complex systems as compositions of more trivial entities and their intercommunications [Navon, 1977; Plaut et al., 1996; Goodwin and Johnson-Laird, 2005; Kemp and Tenenbaum, 2008]. For instance, when we desire to assume something spoken in a foreign language, we often analyze facial expressions, hand gestures, and vocal tones to understand what is being said at a high level. Then, the learned representations are put into the memory for future reference. This insight into the human brain was further analyzed by Botvinick et al. [2001]. They affirmed that the human brain acts in two different ways when exposed to default situations vs novel environments. They labeled the former as *habitual processing* and the latter as *conscious processing*. They especially showed that humans put extra mental effort when dealing with conscious processing. Kahneman [2011], presented a similar notion from a different perspective, where they classified the processing of the brain into two categories: *system 1* and *system 2*. System 1 processing is mostly unconscious, extremely fast, and occurs in familiar environments. In contrast, system 2 processing happens in novel conditions, is slower and requires a train of thoughts.

Current deep models' operations are more comparable to the habitual or system

1 processing of the brain. Given a large amount of data, the model learns to correct itself incrementally, based on an optimization objective. After training, they can efficiently produce answers to complicated problems, as long as the training and testing conditions are similar to an extent. In comparison, humans are more proficient in performing conscious or system 2 processing, allowing rapid adoption to alien conditions. The Global Workspace Theory [Baars, 1988] and the Global Neuronal Workspace model [Shanahan, 2006, 2010, 2012; Dehaene et al., 2017] postulates that the brain works in a modular manner in conscious processing. The brain would look for the most relevant modules for the task in hand and employ them to develop a possible solution. Each of these modules is an expert in some task and holds a unique ability. To prevent the over-use of resources, the information is passed through a bottleneck, allowing only the most relevant modules to work at a time.

On the other hand, learning from limited resources was extremely important in classical ML due to insufficient hardware resources and data. Therefore, emulating the combinatorial generalization of the human brain using structured representations has been the core of classical ML/AI models since its infancy, in many domains such as natural language processing, Bayesian models, probabilistic programming, and reinforcement learning [Lees, 1957; Fikes and Nilsson, 1971; Russell and Norvig, 2002; Džeroski et al., 2001; Koller et al., 2007]. Therefore, it is intriguing to investigate how to incorporate structured representations into deep models.

Chapter 7 and 8 are inspired by this concept. In chapter 7, we use a set of cascaded GANs that work in unison to generate high-resolution point-clouds. Each of these GANs is an expert in identifying and generating patterns in a predetermined frequency band. At the training and inference, each GAN communicates with the others to generate and refine the outputs. Current deep generative models for 3D data generally work on simplified representations (e.g., voxelized objects) and cannot deal with the inherent redundancy and irregularity in point-clouds. A few recent efforts on 3D point-cloud generation offer a limited resolution, but their complexity grows with the increase in output resolution. Chapter 7 develops a principled approach to synthesize 3D point-clouds using a spectral-domain GAN, where our spectral representation is highly structured and allows us to disentangle various frequency bands such that the learning task is simplified for a GAN model. Compared to spatial-domain generative approaches, our formulation allows us to generate high-resolution point-clouds with minimal computational overhead. Furthermore, we propose a fully differentiable block to transform from the spectral to the spatial domain and back, thereby allowing us to integrate knowledge from well-established spatial models. We demonstrate that Spectral-GAN performs well for point-cloud generation tasks. Additionally, it can learn a highly discriminative representation in an unsupervised fashion and can be used to reconstruct 3D objects accurately.

Moreover, most of the current deep conditional generative models assume one-to-one mappings between inputs and outputs. However, many real-world scenarios contain more than one possible solutions for a given condition. For instance, a human artist can draw multiple possible faces for a set of facial landmarks. This aspect is less investigated in current deep generative models, limiting their full potential

[Lee et al., 2019a; Isola et al., 2017]. Chapter 8 introduces a novel general-purpose framework for conditional generation in multimodal spaces that uses latent variables to model generalizable learning patterns, while minimizing a family of regression cost functions. At inference, the latent variables are optimized to find optimal solutions corresponding to multiple output modes. An interesting property of our model is that it uses a separate module to find the optimal latent codes at the inference. The model comprises of two main modules: the generator and the path-finding-expert. During training, the latter learns to traverse to latent codes that correspond to multiple outputs, and at the inference, interacts with the generator to produce diverse outputs. Compared to existing generative solutions, our approach demonstrates faster and stable convergence, and can learn better representations for downstream tasks. Importantly, it provides a simple generic model that can beat highly engineered pipelines tailored using domain expertise on a variety of tasks while generating diverse outputs.

1.4 Thesis Outline

The rest of the chapters are organized as follows:

Chapter 2—Background. This chapter provides the necessary technical background and the preliminaries needed to understand the rest of the thesis.

Chapter 3—Equivariant representation learning. In this chapter, we propose a novel '*volumetric convolution*' operation that can model and convolve arbitrary functions in $\mathbb{B}^3$ in the spectral space. We develop a theoretical framework for *volumetric convolution* based on Zernike polynomials and efficiently implement it as a differentiable and readily pluggable layer for deep networks.

Chapter 4—Blended convolution and synthesis. In this chapter, we propose a two-part solution for 3D object analysis that is combined in a single *blended convolution and synthesis* layer. This fully differentiable layer performs two critical tasks in succession. In the **first** step, it projects the input 3D point clouds into a latent 3D space to synthesize a highly compact and inter-class discriminative point cloud representation. Since 3D point clouds do not follow a Euclidean topology, standard 2/3D convolutional neural networks offer sub-par performance. Therefore, in the **second** step, we propose a novel 3D convolution operator functioning inside the unit ball to extract useful volumetric features.

Chapter 5—Rethinking conditional-GAN training. In this chapter, we show that the cGANs, in their basic form, suffer from significant drawbacks in-terms of diversity and realism. We propose a novel training algorithm that can increase both realism and the diversity of the outputs that are generated by cGANs while preserving the structure of the latent manifold. To this end, we enforce a bi-lipschitz mapping between the latent and generated output manifolds while encouraging Euclidean shortest paths on the latent manifold to be mapped to the geodesics on the generated manifold.

Chapter 6—Robust normalizing flows using Bernstein-type polynomials. This chapter focuses on a framework to construct NFs based on increasing triangular

maps and Bernstein-type polynomials. Compared to the existing (universal) NF frameworks, our method provides compelling advantages like theoretical upper bounds for the approximation error, robustness, higher interpretability, suitability for compactly supported densities, and the ability to employ higher degree polynomials without training instability.

Chapter 7—Spectral-GAN for high-resolution 3D point cloud generation. In this chapter, we develop a principled approach to synthesize 3D point-clouds using a spectral-domain Generative Adversarial Network (GAN). The proposed model consists of a structured series of GANs that operate together to improve the outputs. Furthermore, our spectral representation is highly compact and allows us to disentangle various frequency bands such that the learning task is simplified for a GAN model.

Chapter 8—Conditional generative modeling via learning the latent space. This chapter proposes a novel general-purpose framework for conditional generation in multimodal spaces, which utilizes a path-finding-module to learn the behavior of latent variables while minimizing a family of regression cost functions. At inference, the latent variables are optimized using the path-finding-module to find optimal solutions corresponding to multiple output modes. Compared to existing generative solutions, our approach demonstrates faster and stable convergence, and can learn better representations for downstream tasks.

Chapter 9—Conclusions. We conclude the thesis with a summary of our main contributions while discussing the possible future research directions for incorporating inductive biases and structured representations in deep models.

1.4.1 List of Publications

This thesis is based on five publications and two papers that are currently under review at conferences:

- Ramasinghe, Sameera, Salman Khan, Nick Barnes. “Learned and Hand-crafted Feature Fusion in Unit Ball for 3D Object Classification" in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods* - Volume 1: pp. 115-125. [Best student paper award]

- Ramasinghe, Sameera, Salman Khan, Nick Barnes, and Stephen Gould. “Representation Learning on Unit Ball with 3D Roto-translational Equivariance" in *International Journal of Computer Vision* (2019): 1-23.

- Ramasinghe, Sameera, Salman Khan, Nick Barnes, and Stephen Gould.“Blended convolution and synthesis for efficient discrimination of 3d shapes" in *The IEEE Winter Conference on Applications of Computer Vision*, pp. 21-31. 2020.

- Ramasinghe, Sameera, Salman Khan, Nick Barnes, and Stephen Gould. “Spectral-GANs for High-Resolution 3D Point-cloud Generation", in *IEEE/RSJ International Conference on Intelligent Robots and Systems* , 2020.

- Ramasinghe, Sameera, Kanchana Ranasinghe, Salman Khan, Nick Barnes, and Stephen Gould. "Conditional Generative Modeling via Learning the Latent Space", in *International Conference on Learning Representations*, 2021.
- Ramasinghe, Sameera, Farazi Moshiur, Salman Khan, Nick Barnes, and Stephen Gould. "Rethinking conditional GAN training" in *Neural Information Processing Systems*, 2021.
- Ramasinghe, Sameera, Kasun Fernando, Salman Khan, and Nick Barnes."Robust normalizing flows using Bernstein-type polynomials", under review in *Conference on Learning Theory*, 2022.

Apart from that, the author contributed to the following work, which is not included as part of this thesis.

- Ali Cheraghian, Shafin Rahman, Sameera Ramasinghe, Pengfei Fang, Christian Simon, Lars Petersson, and Mehrtash Harandi. "Synthesized Feature based Few-Shot Class-Incremental Learning on a Mixture of Subspaces." Accepted at the International Conference on Computer Vision, 2021.

Chapter 2

Background and Related Work

This chapter contains an exposition of the machinery and fundamentals necessary to understand the technical details discussed in later chapters of this thesis. We discuss group equivariant networks, generative adversarial networks and normalizing flows, in the respective order.

2.1 Group equivariant networks

The seminal work by Klein [1893] rendered a new perspective on the question *What defines geometry?* In this work, author showed that geometry is, in fact, the study of symmetry groups, i.e., the properties of objects that are invariant under a class of transformations. This revelation was remarkably productive and quickly found its way into physics, which allowed novel insights into complex physics problems. Few such notable works are the phenomenal theorem by Emmy Noether [Noether, 1918], which paved the way to rigorously prove the conservation laws of physics using the symmetry groups, gauge equivariance [Weyl, 1929], and Yang-Mills theory [Yang and Mills, 1954]. In fact, Philip Anderson, the winner of the Nobel prize for physics in 1977, even stated that *"it is only slightly overstating the case to say that physics is the study of symmetry."* Hence, symmetry is a vital concept for any system that aims to discover patterns of data that consist of an underlying geometry.

Therefore, this notion of symmetry is a critical form of inductive bias for ML models that hope to learn generalized semantic features from real-world data. Consequently, group equivariant neural networks have recently grasped the attention of the ML community due to their exciting properties from both theoretical and practical standpoints [Esteves et al., 2018a; Cohen et al., 2018b; Rao et al., 2019; Ramasinghe et al., 2019c; Cohen et al., 2018a; Sosnovik et al., 2019; Horie et al., 2020]. In cases where the function from inputs to outputs contain symmetries with respect to a specific group of transformations (e.g., 3D rotation group ($\mathbb{SO}(3)$) in the 3D object classification task), equivariant networks can significantly reduce the model complexity. We begin the discussion by reviewing basic concepts and then move on to practical aspects.

2.1.1 Symmetry of neural networks

A group is formally defined follows:

Definition 2.1. *A group $(\boldsymbol{G}, \cdot)$ is a set $\boldsymbol{G}$ equipped with an associative binary operation $\boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$, an identity element, and where every element has an inverse that is also in the set.*

Further, a symmetry group of an object can be loosely defined as below:

Definition 2.2. *A symmetry group of an object is the group of all transformations under which (some aspect of) the object is invariant.*

The notion of symmetry can be considered under two aspects: invariance and equivariance. For example, take the transformation $f : X \to X$ and the label function $L : X \to Y$, where X and Y are the input and output spaces, respectively. Then, L is invariant to f if it satisfies $L \circ f = L$. Similarly, L is equivariant to f if it satisfies $L \circ f = f \circ L$. In other words, under invariance, the function output does not change, while under equivariance, the function output changes in a deterministic manner. In most cases, we are interested in the equivariance rather than the invariance. For instance, consider a face recognition system. If the learned model is invariant to the relative positions of the facial features, it will not detect any abnormality in an image where the facial attributes are arbitrarily placed. On the contrary, in an equivariant model, this information will be propagated forward in a deterministic manner, which helps the model to raise a red flag in such a situation.

Interestingly, many such natural symmetries exist in common data-types we work with. Consequently, the networks that work exceedingly well on these data types intrinsically preserve these symmetries while extracting features. Figure 3.5 depicts several such examples.

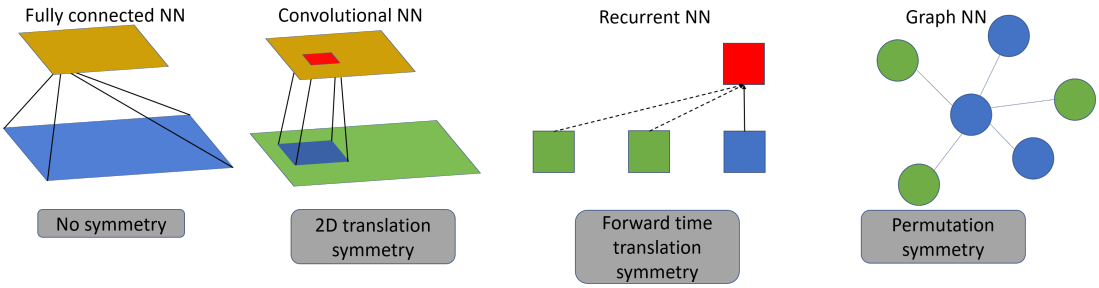


Figure 2.1: Many commonly used layers in deep networks naturally preserve a form of symmetry (currently processing data points are indicated by blue).

In equivariant networks, we typically consider linear group actions, i.e., linear group representations. Although this might seem restrictive, the space of all linear group representations is vast, hence does not pose a problem for practical applications. A more formal definition of the linear group representations is given below:

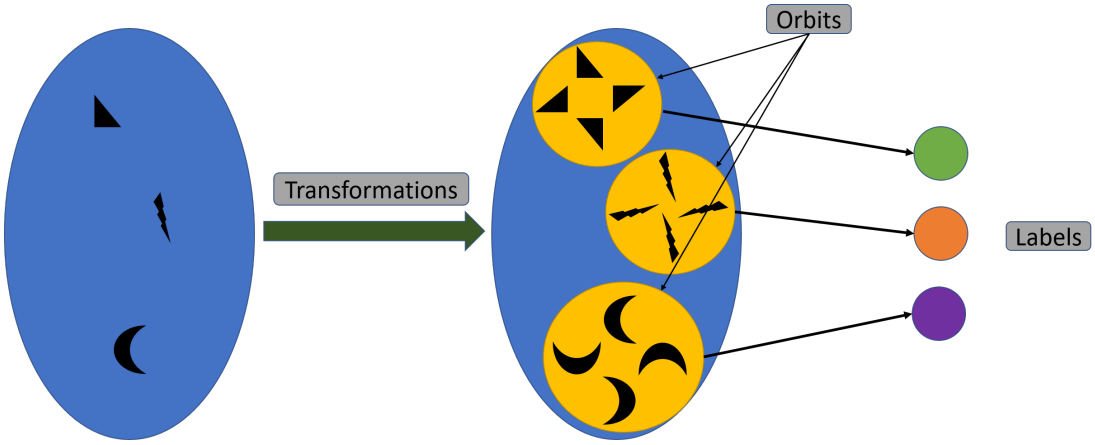


Figure 2.2: Orbits generated by group transformations.

Definition 2.3. *A group homomorphism between groups* $\boldsymbol{G}$ *and* $\boldsymbol{H}$ *is a map* $f : \boldsymbol{G} \to \boldsymbol{H}$ *such that* $f(g_1 g_2) = f(g_1) f(g_2)$. *Let* $\boldsymbol{G}$ *be a group and* $\boldsymbol{V}$ *a vector space over some field. A linear group representation is a group homomorphism* $\rho : \boldsymbol{G} \to \boldsymbol{GL}(\boldsymbol{V})$, *where* $\boldsymbol{GL}(\boldsymbol{V})$ *is the general linear group.*

It can be shown that when $\mathbf{V}$ is finite-dimensional, $\mathbf{GL}(\mathbf{V})$ is identifiable with the group of $n \times n$ invertible matrices, where n is the dimension of $\mathbf{V}$.

Interestingly, the equivariant networks can be generalized using linear group representations. Let us denote x_i and $\rho_i(g)$ as features and group representations, respectively, corresponding to layers $i = 1, 2, \ldots, N$ of a network and $g \in \mathbf{G}$ where $\mathbf{G}$ is a symmetry group. Further, consider f_i to be the function that maps x_{i-1} to x_i. Then, an equivariant network satisfies the following relationship:

$$f_i \circ \rho_{i-1}(g) = \rho_i(g) \circ f_i. \tag{2.1}$$

2.1.2 Orbits and equivalance relations

Let X and $\mathbf{G}$ be a set of data points and a group of transformations. Then, an orbit is defined as $O_x = \{g(x) | x \in X, g \in \mathbf{G}\}$. In other words, an orbit is the set of all points we can obtain by applying a transformation from a group. If the label function is symmetric with respect to G, then an equivariant network has only to learn general features corresponding to each orbit instead of concentrating on each sample, which dramatically reduces the required computational complexity.

Further, a model can be made *symmetry-aware* in three ways:

- **Data augmentation.** This is the most straightforward approach to follow in order to achieve equivariance/invariance. The motive is to artificially expand the dataset by applying relevant transformations on the data points and let the network learn a generic mapping from inputs to labels. However, if the orbit of

the transformation group is vast, this method becomes infeasible. Further, the required model capacity significantly increases since features are learned per data point, rather than per orbit. However, this approach is appropriate in cases where it is difficult to formalize or identify the required symmetry group.

- **Transform data into invariant representations.** This is only suitable for situations where invariant representations are needed. The strategy is to apply a pre-processing step on the data before feeding to the network, so the data becomes invariant to a transformation group. Consequently, the network only observes already invariant representations, therefore, does not have to learn an invariant function. The challenge here is to convert the data to invariant representations, which might not be plausible for some transformation groups. Moreover, since tackling the equivariance is not possible using this approach, the network might suffer from the drawbacks discussed earlier.

- **Equivariant network design**. The ideal way to learn equivariant representations is to design models using equivariant layers. It can be shown that a composition of equivariant layers is also equivariant under the same transformation group. Hence, carefully designed equivariant layers can be used as building blocks to construct deep equivariant networks.

2.2 Generative adversarial networks

Generative adversarial networks (GAN) are a class of deep generative models that has become prevalent in many modern generative tasks, including image generation [Ledig et al., 2017; Choi et al., 2018; Royer et al., 2020], video generation [Chu et al., 2020b; Saito and Saito, 2018; Tulyakov et al., 2018], natural language processing [Dash et al., 2017; Guo et al., 2018; Ahamad, 2018], and audio synthesis Donahue et al. [2018]; Dong et al. [2018]; Engel et al. [2019], due to its ability to generate high-quality content that can even fool human observers. In this section, we provide a basic overview of GANs.

A GAN consists of two main components, i.e., the generator and the discriminator. The optimization procedure of GANs can be analyzed from a game-theoretic perspective, where the generator and the discriminator play a min-max game against each other until they approach a Nash equilibrium. This min-max game forces the discriminator to learn to classify samples as real or synthetic, while the generator attempts to fool the discriminator by generating samples that are indistinguishable from real samples. It can be shown that the equilibrium of the above game is reached when the Jensen-Shannon (JS) divergence between the synthetic and real sample distributions approaches zero. We formally demonstrate this in Section 2.2.1.

2.2.1 Equilibrium of GANs

The Kullback–Leibler (KL) divergence between two probability distributions p and q can be obtained by,

$$D_{KL}(p||q) = \int_x p(x)\log\frac{p(x)}{q(x)}dx \tag{2.2}$$

KL-divergence is an asymmetric measure. In contrast, JS divergence is a symmetric metric that can measure the similarity between two distributions.

$$\mathbf{JSD}(p||q) = \frac{1}{2}(D_{KL}(p||\frac{p+q}{2}) + D_{KL}(q||\frac{p+q}{2})). \tag{2.3}$$

The min-max game between the discriminator D and the generator G can be formally stated as follows:

$$\begin{aligned}\min_G\max_D V(D,G) &= \mathbb{E}_{x\sim p_d}[\log D(x)] + \mathbb{E}_{z\sim p_z}[1-\log D(G(z))] \\ &= \mathbb{E}_{x\sim p_d}[\log D(x)] + \mathbb{E}_{x\sim p_g}[1-\log D(G(z))],\end{aligned} \tag{2.4}$$

where p_d is the training distribution and p_g is the generated distribution. It can be shown that optimal value $D^*(x)$ that maximizes $V(D,G)$ is achieved when,

$$D^*(x) = \frac{p_d}{(p_d+p_g)}. \tag{2.5}$$

The Nash equilibrium of Eq. 2.4 occurs as $p_d \to p_g$, i.e., $D^*(x) = \frac{1}{2}$. This is intuitive, since when the generator produces identical samples to the real distribution, the discriminator cannot distinguish between them. Therefore, it can be seen that at the global optimum $V(D,G) = -2\log 2$.

This equilibrium can be interpreted using the JS-divergence.

$$\begin{aligned}\mathbf{JSD}(p||q) &= \frac{1}{2}(D_{KL}(p||\frac{p+q}{2}) + D_{KL}(q||\frac{p+q}{2})), \\ &= \frac{1}{2}[2log2 + \int_x p_d\log\frac{p_d}{p_d+p_g}dx + \int_x p_g\log\frac{p_g}{p_d+p_g}dx] \\ &= \frac{1}{2}(2log2 + V(D,G))\end{aligned} \tag{2.6}$$

Therefore,

$$V(D^*,G) = 2\mathbf{JSD}(p_d||p_g) - 2log2 \tag{2.7}$$

Since $V(D,G) = -2\log 2$ at the equilibrium, $\mathbf{JSD}(p_d||p_g) = 0$.

2.2.2 Problems in GANs

Compared to generative models such as Boltzman machines and variational autoencoders (VAE), a main advantage of GANs is that they do not need to handle the partition function. However, despite the remarkable success, GANs are known to suffer from exigent problems. We discuss several such problems below:

Convergence. The convergence of GANs is only guaranteed when $\max_D V(g,d)$ is convex in the parameter space of the generator [Goodfellow et al., 2014a]. However, this is fallacious in many practical scenarios, and drives GANs to be highly unstable during the training if careful measures were not taken. The underlying cause for this phenomenon is that the equilibrium for Eq. 2.4 is not a local minima, but a saddle point in the parameter space: V has to be a local maxima with respect to a one player and a local minima with respect to the other player. Therefore, unless $\max_D V(g,d)$ is (almost) convex in the parameter space of the generator, the players can keep minimizing and maximizing V, never approaching an equilibrium. As a toy example, consider the function $V(x,y) = xy$. The player 1 tries to minimize xy by controlling x while the player 2 tries to minimize $-xy$ by controlling y. Then, the path of V is a circular orbit, assuming small gradient steps.

Nevertheless, it has been empirically validated that if model architecture and the configurations are chosen carefully, GANs are able to generate exceptionally high quality samples. One of the earliest attempts of generating realistic images is the deep convolutional GAN (DCGAN) proposed by Radford et al. [2015]. They confirmed that despite the training instability, carefully chosen hyper-parameters and architectures could lead to well-optimized GANs. Since then, many pioneering works on GANs have produced remarkable results using precisely engineered architectures [Denton et al., 2015; Karras et al., 2017; Achlioptas et al., 2017b; Mao et al., 2017; Zhang et al., 2019; Arjovsky et al., 2017]

Training instability. GANs are latent variable generative models, i.e., the generator outputs are encoded in a low dimensional latent space. Such a construct works well in practice since image distributions, though seemingly high dimensional, are merely low dimensional manifolds embedded in high dimensional spaces. For instance, although there are 256^{1024} permutations of a 32×32 size image (with 8 bits), real-world images are concentrated only on disjoint small areas considering the entire high-dimensional manifold. Arjovsky and Bottou [2017] confirmed that having such disjoint ground truth manifolds makes the model highly unstable during training. We refer the reader to Arjovsky and Bottou [2017] for a more comprehensive analysis on the training dynamics of GANs.

Vanishing gradient. Consider a scenario where the discriminator is perfect. At this state, $V(D,G) = 0$. Hence, the gradients approach zero, and the generator weights are not updated. Particularly, if the discriminator becomes better at classifying real and fake samples at the initial stages of training, the generator is not updated at all. Fig. 2.3 demonstrates a result from a simple experiment extracted from Arjovsky and Bottou [2017]. As a remedy, they proposed an alternative GAN training scheme, using a modified loss function, that can significantly improve the stability of learning

regardless of the state of the generator and the discriminator.

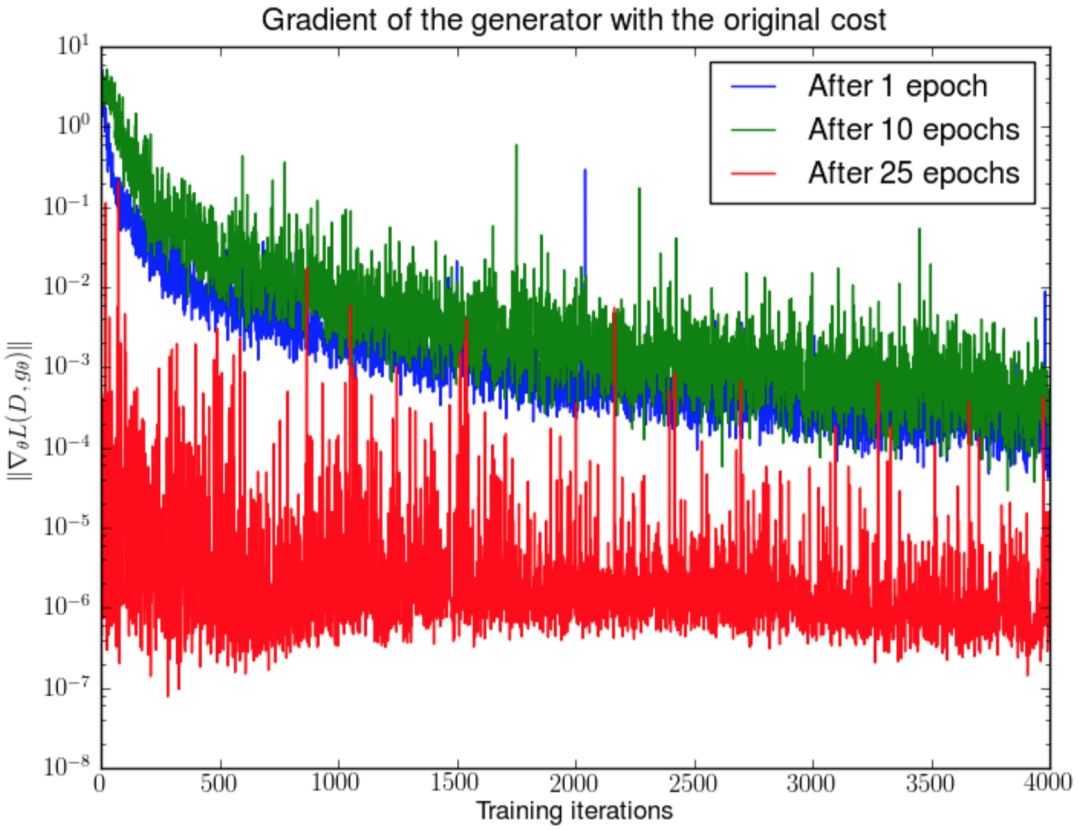


Figure 2.3: First, a DCGAN is trained for 1, 10 and 25 epochs. Then, with the generator fixed a discriminator is trained from scratch. It is evident that the error quickly goes to 0, even with very few iterations on the discriminator. This even happens after 25 epochs of the DCGAN, when the samples are remarkably good and the supports are likely to intersect, pointing to the non-continuity of the distributions. Note the logarithmic scale. For illustration purposes the accuracy of the discriminator is also shown, which goes to 1 in sometimes less than 50 iterations. This is 1 even for numerical precision, and the numbers are running averages, pointing towards even faster convergence.

Lack of a proper evaluation metric It is not possible to directly evaluate the density of samples using GANs. As a result, the direct evaluation of a GAN's performance is an open research problem. Consequently, comparing GAN architectures and determining the optimal model while training is cumbersome. Although alternative metrics have been proposed, there is no agreed-upon method as to which metric best provides a fair insight into the GAN performance. Few such popular metrics are the inception score (IS), frechet inception distance (FID), Image Retrieval Performance, and human judgment.

Mode collapse Mode collapse is a commonly observed problem with GANs. Let

us denote the real data distribution associated with a generative task be p_r. Note that p_r is the distribution of set of all possible samples, and is larger than number of samples in the training distribution p_d. Then, the objective function Eq. 2.4 is used to optimize the generator G to generate the distribution $p_g \approx p_r$. However, in practice, training datasets do not cover the entire p_r distribution. Therefore, Objective 2.4 assumes a *sufficiently large* number of training samples M to converge p_g to p_r. Then, the expectation $\mathbb{E}_{x\sim p_r}[D(x)]$ is approximated by the empirical value $\frac{1}{J}\sum_{j=1}^{J} D(x_j))$, where $x_1, ..., x_J \sim p_d$. Arora et al. [2017] showed that the required J could be as large as $\exp(M)$, otherwise, p_g can be far from the optimal distribution, causing the generator to have very low support, i.e., *mode collapse*.

2.3 Normalizing flows

Normalizing flows (NF) can be broadly categorized as generative models that can estimate the density of arbitrary data points. To the best of our knowledge, the term *normalizing-flows* was first introduced by Tabak and Turner [2013], where they proposed a scheme for density estimation using a composition of simple maps. In their model, the parameters of each map are estimated by the maximization of a local quadratic approximation to the log-likelihood. Further developing this concept of composing transformations for obtaining a more expressive composite transformation, Rippel and Adams [2013] used deep neural networks to construct flows. Afterward, the seminal work by Rezende and Mohamed [2015] used normalizing flows to approximate variational inference. They showed that replacing the commonly used Gaussian prior with a multimodal distribution obtained through NFs can dramatically improve the performance of variational inference. NFs learn an invertible mapping between a prior and a more complex distribution (the target) that have the same dimensionality. Typically, the prior is chosen to be a Gaussian with an identity covariance or that is uniform on the unit cube, and the target is the one we intend to learn. Below, we present a summary of related ideas and refer the readers to Jaini et al. [2019] and Kobyzev et al. [2020] for a comprehensive discussion.

More formally, let $\mathbf{z}$ and $\mathbf{x}$ be sampled data from the prior with density P_z and the target distribution with density P_x, respectively. Then, NFs learn $f(\mathbf{z}) = \mathbf{x}$ which is differentiable and invertible with a differentiable inverse. Such transformations are called diffeomorphisms and they allow the estimation of the probability density $P_x(\mathbf{x})$ via the change of variables formula, as follows,

$$P_x(\mathbf{x}) = \frac{P_z(f^{-1}\mathbf{x})}{|\mathbf{J}_f(f^{-1}\mathbf{x})|} \tag{2.8}$$

where $\mathbf{J}_f$ is the Jacobian determinant of f.

An important property of diffeomorphisms is that they can be composed together to obtain a more complex diffeomorphism. Consider two transformations f_1 and f_2.

Then, the following properties hold:

$$(f_1 \circ f_2)^{-1} = f_2^{-1} \circ f_1^{-1}, \tag{2.9}$$

$$\left|\mathbf{J}_{f_1 \circ f_2}(x)\right| = \left|\mathbf{J}_{f_1}(f_2(x))\right| \left|\mathbf{J}_{f_2}(x)\right|. \tag{2.10}$$

Hence in practice, most NFs use a sequence of invertible functions to increase the expressiveness, and hence, we have

$$f = \tilde{f}_N \circ \tilde{f}_{N-1} \circ \cdots \circ \tilde{f}_1 \tag{2.11}$$

where $\tilde{f}_k$'s are diffeomorphisms, as the transformation between the two densities.

2.3.1 Optimization

Given an independent and identically distributed (i.i.d.) sample $\{x_1, \ldots, x_n\}$ with law P_x, learning the target density P_x and the transformation f (within an expressive function class $\mathfrak{F}$) is done simultaneously via minimizing the Kullback-Leibler (KL) divergence between P_x and the pushforaward of P_z under f denoted by $f_* P_z$. In situations where the density of the training samples is not available, we can use the following method to obtain an estimator for minimizing the KL-divergence:

$$\begin{aligned} &\min_{f \in \mathfrak{F}} \mathrm{KL}(P_x \| f_* P_z) \\ &\qquad = -\max_{f \in \mathfrak{F}} \int \log \frac{P_z(f^{-1}\mathbf{x})}{|\mathbf{J}_f(f^{-1}\mathbf{x})|} \cdot P_x(\mathbf{x})\, d\mathbf{x}. \end{aligned} \tag{2.12}$$

In the implementation, the integral in (2.12) is replaced by an estimator, the empirical average,

$$\frac{1}{n} \sum_{k=1}^{n} \Big(\log P_z(f^{-1} x_k) - \log |\mathbf{J}_f(f^{-1} x_k)| \Big), \tag{2.13}$$

and the problem reduces to learning f that maximizes (2.13).

On the contrary, in scenarios where we cannot sample from the target density P_x, but density estimation of arbitrary data points is possible, we can use an alternative objective function to optimize the flow model.

$$\begin{aligned} KL(f_* P_z \| p_x) &\approx \frac{1}{N} \sum_{k=1}^{N} \Big(\log P_x(f(z)) - \log f_* P_z(z) \Big) \\ &\approx \sum_{k=1}^{N} \Big(\log P_z(z) - \log \Big|\mathbf{J}_f(z)\Big| - \log f_* P_z(z) \Big) \end{aligned} \tag{2.14}$$

Consequently, the objective is to minimize the empirical estimator 2.14.

2.3.2 Triangular maps

Optimizing the flow model using Eq. 2.12 requires efficiently calculating the Jacobian as well as f^{-1}. Note that in this case, it is not necessarily required to compute f. However, the computation of f is needed if we hope to sample from the trained model. In contrast, if we use Eq. 2.14 to optimize the model, computation of f^{-1} is not required. However, in this case, f^{-1} needs to be computed if we want to estimate the sample density.

All of the above requirements—the calculation of f, f^{-1}, and jacobian determinants—can be achieved via constraining f to be an *increasing triangular map*. That is, taking $P_x(\mathbf{x})$ to be a multivariate distribution where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, and the prior $P_z(\mathbf{z})$ where $\mathbf{z} = (z_1, z_2, \ldots z_d)$, the components of $\mathbf{x}$ are expressed as $x_j = f_j(z_1, z_2, \ldots, z_j)$ for suitably defined transformations $f_j, j = 1, 2, \ldots, d$ where f_j is increasing with respect to z_j. In this case, the Jacobian determinant is the product $\prod_{j=1}^{d} \partial_{z_j} f_j$. Also, because f_j is increasing in z_j, inversion can be done recursively starting from f_1^{-1}. Picking f from the class of increasing triangular maps does not compromise the expressiveness due to the following theorem in Bogachev et al. [2005]

Theorem 2.1 (Bogachev et al.)**.** *If μ and ν are absolutely continuous Borel probability measures on $\mathbb{R}^d$, then there exists an increasing triangular map (unique up to null sets of μ), $f : \mathbb{R}^d \to \mathbb{R}^d$, such that $\nu = f_*\mu$.*

The origin of triangular maps runs back to the density transformation model suggested by Rosenblatt [1952], and similar studies that were later conducted by Knothe [1957] and Talagrand [1996]. Due to their appealing properties, triangular maps have been a popular construct in many density estimation problems. Redlich [1993] and Deco and Brauer [1995] were the first to use the term *triangular maps* in density estimation. More recently, the models proposed by El Moselhy and Marzouk [2012]; Marzouk et al. [2016]; Dinh et al. [2014, 2016]; Jaini et al. [2019]; Durkan et al. [2019b]; Ramasinghe et al. [2021] utilized the same concept. In fact, Jaini et al. [2019] affirmed that many other existing popular flow-based frameworks [Uria et al., 2016; Kingma et al., 2016; Huang et al., 2018a; Germain et al., 2015; Papamakarios et al., 2017] also are, in fact, variants of triangular maps.

Chapter 3

Equivarient Representation Learning in Unit Ball

Natural objects consist of redundancies, repeatable patterns, and symmetries. A key focus of machine learning models is to exploit these patterns in order to construct compact representations. Learning symmetries of natural signals plays an integral part in this process. One way this task can be achieved is via data augmentation, *i.e.*, transforming the dataset using a particular set of group transformations, and attempting to learn these transformations using a neural network. However, in this approach, there is no guarantee that the network will systematically learn the symmetries. Further, Bubeck and Sellke [2021] recently showed that the amount of parameters a neural networks needs to model a robust representation of n data points is nd, where n is the number of parameters and d is the dimension of the inputs. Therefore, one can expect to end up with unnecessarily large networks when trying to learn symmetries via data augmentation. In contrast, a more elegant approach to learn symmetries is carefully designing feature extraction methods such that the learned representations are equivariant with respect to the transformations of interest. Ubiquitous 2D CNNs are a good example for this, as 2D convolutions are equivariant over 2D translations.

Consequently, convolution-based deep neural networks have performed exceedingly well on 2D representation learning tasks (Krizhevsky et al. [2012], He et al. [2016]). The convolution layers perform parameter sharing (referred as '*weight tying*') to learn repetitive features across the spatial domain while having lower computational cost by using local neuron connectivity. However, state-of-the-art convolutional networks can only work on Euclidean geometries and their extension to other topological spaces e.g., spheres, is an open research problem. Remarkably, the adaptation of convolutional networks to spherical domain can advance key application areas such as robotics, geoscience and medical imaging.

Some recent efforts have been reported in the literature that aim to extend convolutional networks to spherical signals. Initial progress was made by Boomsma and Frellsen [2017], who performed conventional planar convolution with carefully selected padding on a spherical-polar representation and its cube-sphere transformation Ronchi et al. [1996]. A recent pioneering contribution by Cohen et al. [2018b] used

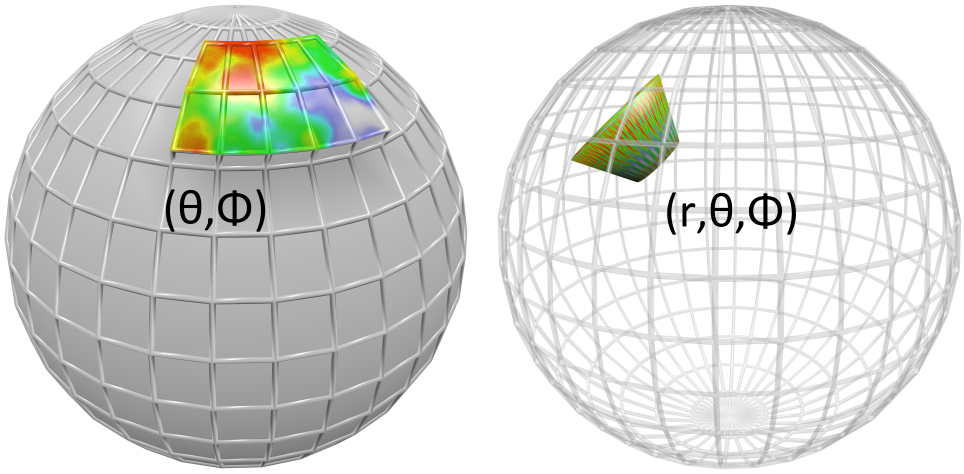


Figure 3.1: Fig. 1: Kernel representations of spherical convolution (*left*) vs. volumetric convolution (*right*). In volumetric convolution, the shape is modeled and convolved in $\mathbb{B}^3$ and in contrast, spherical convolution is performed in $\mathbb{S}^2$.

harmonic analysis to perform efficient convolution on the surface of the sphere to achieve rotational equivariance. The aforementioned works however do not systematically consider radial information in a 3D shape and the feature representations are learned at fixed radii. Specifically, Cohen et al. [2018b] estimated similarity between spherical surface and convolutional filter in $\mathbb{S}^2$, where the kernel moves in 3D rotation group $\mathbb{SO}(3)$.

In this chapter, we propose a novel approach to perform *volumetric convolution* inside a unit ball ($\mathbb{B}^3$) that explicitly learns representations across the radial axis. Although we derive generic formulae to convolve functions in $\mathbb{B}^3$ we stick to two popular use cases here i.e., 3D shape recognition and retrieval. In comparison to closely related spherical convolution approaches, modeling and convolving 3D shapes in $\mathbb{B}^3$ entail key advantages: '*volumetric convolution*' can capture both 2D texture and 3D shape features and can handle non-polar 3D shapes. Furthermore, volumetric convolution is equivariant to both 3D rotation and and radial translation, which enhances its ability to capture more robust features from 3D functions. We develop the theory of volumetric convolution using orthogonal Zernike polynomials Canterakis [1999], and use careful approximations to efficiently implement it as low-cost matrix multiplications. Our experimental results demonstrate significant boost over spherical convolution and confirm the high discriminative ability of features learned through volumetric convolution. Fig. 4.1 compares volumetric and spherical convolution kernels.

Given that our proposed convolution operation is based on 3D orthogonal moments, we derive an explicit formula in terms of Zernike polynomials to measure the axial symmetry of a function in $\mathbb{B}^3$, around an arbitrary axis. This relation is generally applicable to function analysis tasks and here we demonstrate one particular use case with relevance to 3D shape recognition and retrieval. Specifically, we use the

derived formula to propose a hand-crafted descriptor that accurately encodes the axial symmetry of a 3D shape. Moreover, we decompose the implementation of both volumetric convolution and axial symmetry measurement into differentiable steps, which enables them to be integrated in any end-to-end architecture.

Finally, we propose an experimental architecture to demonstrate the practical usefulness of proposed volumetric convolution. A remarkable feature of our architecture is the novel spectral domain pooling layer that enhances performance, enables learning more compact features and significantly reduces the number of trainable parameters in the network. It is worth pointing out that the proposed experimental architecture is only a single possible example out of many possible architectures, and is primarily focused on demonstrating the usefulness of the volumetric convolution layer as a fully differentiable and easily pluggable layer, which can be used as a building block in end-to-end deep architectures.

The main contributions of this work include:

- Development of the theory for volumetric convolution that can efficiently convolve functions in $\mathbb{B}^3$ and achieve equivariance over 3D rotation and translation of local patterns.
- Implementation of the proposed volumetric convolution as a fully differentiable module that can be plugged into any end-to-end deep learning architecture.
- A novel formula to measure the axial symmetry of a function defined in $\mathbb{B}^3$, around an arbitrary axis using Zernike polynomials.
- The first approach to perform volumetric convolution on 3D objects that can simultaneously model 2D (appearance) and 3D (shape) features.
- An experimental end-to-end trainable architecture with a novel spectral pooling layer that automatically learns rich 3D shape descriptors.

The rest of the chapter is structured as follows. We first introduce related work and basic concepts extensively used in the chapter in Sec. 3.1 and 3.2 respectively followed by a detailed description of proposed volumetric convolution approach in Sec. 3.3. The axial symmetry measurement formula is derived in Sec. 3.4. We present an example CNN architecture based on proposed convolution technique in Sec. 3.5. In Sec. 6.6 we demonstrate the effectiveness of the derived operators through extensive experiments. Finally, we conclude the chapter in Sec. 3.8.

3.1 Related works

Equivariance in 3D: The convolution operation in 2D provides translation equivariance i.e., $f(t(\cdot)) = t(f(\cdot))$ where f, t denote the convolution and transformation functions respectively. However, conventional convolution does not guarantee equivariance to an object's pose (rotation, translation). This is a highly desirable property in 3D shape analysis, e.g., a simple rotation of an object should not alter its category.

To resolve this, Cohen and Welling Cohen et al. [2018b] proposed Spherical CNN that performs cross-correlation after projecting images on the surface of the sphere. Worrall and Brostow [2018] introduced an operator for voxelized inputs that is linearly equivariant to 3D rotations and translations. Another interesting extension of Cohen et al. [2018b] has recently been reported in Kondor et al. [2018] where Clebsch-Gordon transform is used as a spectral domain non-linearity to realize a fully Fourier domain Spherical CNN. Thomas et al. [2018] proposed a tensor field network that uses spherical harmonics similar to Cohen et al. [2018b]; Worrall et al. [2017]; Worrall and Brostow [2018] and exhibits local equivariance to rotations, translations and 3D point permutations. These efforts are focused on spherical projections Cohen et al. [2018b]; Kondor et al. [2018] or point-clouds Kondor [2018]; Thomas et al. [2018] and cannot be directly applied to volumetric inputs. Furthermore, Weiler et al. [2018a] recently proposed a solution to the problem of SE(3) equivariance by modeling 3D data as dense vector fields in 3D Euclidean space. In this work however, we focus on $\mathbb{B}^3$ to achieve radial translational and rotational equivariances over local patterns. Note that an earlier version of this work lacks translational equivariance [Ramasinghe et al., 2019a].

Orthogonal Moments: Orthogonal moments are useful tools for analyzing structured data. Generally, the goal of orthogonal moments is to obtain a descriptor from a data representation, that is invariant to certain deformations and transformations such as translation, rotation and scaling (TRS). Compared to geometric moments, orthogonal moments behave favorably under aforementioned transformations and therefore have been extensively used in 2D data analysis in past Hu [1962]; Lin and Chellappa [1987]; Arbter et al. [1990]; Tieng and Boles [1995]; Khalil and Bayoumi [2001]; Suk and Flusser [1996] . Many 3D TRS invariant moments are extensions of their 2D counter-parts, although extending invariant moments from 2D to 3D is not a straight forward task as rotation in 3D is not commutative. Despite this complexity, many attempts to obtain TRS invariant orthogonal moments for 3D data have been reported in literature Guo [1993]; Reiss [1992]; Canterakis [1996, 1999]; Flusser et al. [2003]. The behaviour of orthogonal moments are strongly dependant on the Hilbert space in which they are defined. For example, some moments are orthogonal inside a cube and other moments are orthogonal on a sphere or inside a unit ball. The moments defined inside a cube are less convenient for extracting rotation invariants, compared to a sphere and a ball. Although El Mallahi et al. [2017] and Yang et al. [2015] proposed orthogonal moments inside unit ball, they lack two key properties, which prevents them from being used as basis functions for convolution operations: *1)* loss of orthogonality under 3D rotation, *2)* the *completeness* of basis polynomials has not been proved in unit ball, which hampers its ability to represent an arbitrary complex function with minimal number of terms. In contrast, 3D Zernike polynomials Canterakis [1999] have both aforementioned properties, which makes them an attractive choice for basis polynomials of our volumetric convolution. Recently, Janssen et al. [2018] also used generalized 3D Zernike basis functions to represent a 3D version of cake-wavelets, which then obtain orientation scores between elongated 3D structures. *First*, they implement the 3D cake-wavelet functions using a discrete

Fourier transform based method, which does not have an analytical description in the spatial domain. Therefore, they present an analytical version of the same 3D cake-wavelets using a 3D Zernike basis functions, followed by a continuous Fourier transform. They primarily evaluate their method on obtaining orientation scores between 3D biomedical data, such as 3D rotational Xray images, which illustrates the capacity of 3D Zernike moments in representing highly non-polar and textured data. Our work, however, is not limited to obtaining handcrafted analytical features, as we learn deep features using the properties of 3D Zernike moments.

3D Shape Recognition and Retrieval: As a case study, this chapter considers popular 3D shape recognition and retrieval problems that can directly benefit from discriminative volumetric representations. Traditionally, a diverse set of approaches have been developed for this task including handcrafted features Vranic and Saupe [2002]; Guo et al. [2016], unsupervised learning Wu et al. [2015, 2016]; Khan et al. [2018] and deep learning Qi et al. [2017a]; Li et al. [2016]; Qi et al. [2016]. Among hand-crafted shape descriptors, a popular choice is spherical harmonics that are computed using Fourier domain coefficients Vranic and Saupe [2002]; Canterakis [1996]. The hand-designed descriptors were targeted towards encoding both global (e.g., angle histograms Ankerst et al. [1999] and shape distributions Osada et al. [2002]) and local 3D shape patterns (e.g., signature of histogram of orientations Tombari et al. [2010] and 3D shape context Frome et al. [2004]). More recently, Wu et al. [2015] introduced a convolutional deep belief network (DBN) that models the probabilistic distributions of 3D data. Since 3D convolutions are computationally expensive, Li et al. [2016] proposed to approximate 3D spaces as volumetric fields. However, all of these deep learning based approaches perform convolutions in Euclidean space which is sub-optimal for 3D shapes. Different from them, Ramasinghe et al. [2019b] incorporated volumetric convolution within deep networks to achieve equivariance to translations and rotations. Hierarchical non-linear networks that operate on point-clouds were proposed in Qi et al. [2017a,b]. The proposed network design provides invariance to point permutations but do not achieve equivariance to 3D rotations.

3.2 Preliminaries

We have defined commonly used mathematical symbols in this chapter, in Table 3.1. Before delving into the details of proposed volumetric convolution, we briefly cover basic concepts below.

3.2.1 Moments

Moments are projections of a function f onto a polynomial basis defined in a certain space. If the polynomial basis is orthogonal and complete, any arbitrary function in that space can be reconstructed using the corresponding moments.

Definition: *Let $\Phi(X_p)$ be a n-variable polynomial basis of the space Ω. Let $p = (p_1, ..., p_n)$ be a multi-index of non-negative integers which shows the highest power of the respective*

Table 3.1: Mathematical symbols frequently used in this chapter.

Symbol	Description
$\mathbb{B}^3$	Unit ball ($\mathbb{B}^3$) can be regarded as the set of points $u \in \mathbb{R}^3$ where $\|u\| < 1$. Any point in unit ball can be parameterized using coordinates (θ, ϕ, r).
$\mathbb{S}^2$	Surface of the unit sphere ($\mathbb{S}^2$) can be regarded as the set of points $u \in \mathbb{R}^3$ where $\|u\| = 1$. Any point in $\mathbb{S}^2$ can be parameterized using coordinates (θ, ϕ).
†	Complex conjugate.
$\langle \cdot, \cdot \rangle$	Let f and g be complex functions defined in a space Ω. Then $\langle f, g \rangle = \int_\Omega f(X) g(X)^\dagger dX, X \in \Omega$
$\mathbb{SO}(3)$	3D rotation group.
$\tau_{\alpha,\beta}$	A rotation operation which aligns the north-pole with the axis towards α (azimuth) and θ (polar) angles.
Re	Real component of a complex value.
$Imag$	Imaginary component of a complex value.
$R_y(\alpha)$	A 3D rotation applied around y axis.
$Z_{n,l,m}(\theta, \phi, r)$	$(n, l, m)^{th}$ order 3D Zernike polynomial.
$\Omega_{n,l,m}(f)$	$(n, l, m)^{th}$ order 3D Zernike moment of function f.
$Y_{l,m}(\theta, \phi)$	$(l, m)^{th}$ order spherical harmonic function.
$D^l_{m,n}(\alpha, \beta, \gamma)$	$(n, l, m)^{th}$ order Wigner D-matrix.
$sym_{(\alpha,\beta)}(g)$	Let S be the set of functions in $\mathbb{B}^3$ which are symmetric around the axis towards (α, β), where α and β are azimuth and polar angles respectively. Then $sym_{(\alpha,\beta)}(g)$ is the projection of a function $g \in \mathbb{B}^3$ into S.

variables in $\Phi(X_p)$. Then general moment M_p of f is defined as,

$$M_p = \int_\Omega \Phi(X_p) f(X) dX. \tag{3.1}$$

3.2.2 Equivariance

A function is said to be an equivariant map when its domain and codomain are acted on by the same symmetry group, and when the function commutes with the action of the group. That is, applying a symmetry transformation and then computing the function produces the same result as computing the function and then applying the transformation. We formally define equivariance as follows:

Definition: *Consider a set of transformations G, where individual transformations are indexed as $g \in G$. Consider also a function or feature map $\phi : X \longrightarrow Y$ mapping inputs $x \in X$ to outputs $y \in Y$. Transformations can be applied to any $x \in X$ using the operator $T_g^X : X \longrightarrow X$, so that $x \longrightarrow T_g^X[x]$. The same can be done for the outputs with $y \longrightarrow T_g^Y[y]$. We say that ϕ is equivariant to G if*

$$\phi(T_g^X[x]) = T_g^Y[\phi(x)]. \tag{3.2}$$

3.2.3 Spherical Harmonics

Spherical harmonics are a set of complete and orthogonal functions defined on the surface of the unit sphere as

$$Y_{l,m}(\theta, \phi) = (-1)^m \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}} P_l^m(\cos\phi) e^{im\theta}, \tag{3.3}$$

where $\phi \in [0, \pi]$ is the polar angle, $\theta \in [0, 2\pi]$ is the azimuth angle, $l \in \mathbb{Z}^+$ is a non-negative integer, $m \in \mathbb{Z}$ is an integer, $|m| < l$, and $P_l^m(\cdot)$ is the associated Legendre function

$$P_l^m(x) = (-1)^m \frac{(1-x^2)^{m/2}}{2^l l!} \frac{d^{l+m}}{dx^{l+m}} (x^2 - 1)^l. \tag{3.4}$$

Since spherical harmonics hold the orthogonality property

$$\int_0^{2\pi} \int_0^{\pi} Y_l^m(\theta, \phi) Y_{l'}^{m'}(\theta, \phi)^\dagger \sin\phi \, d\phi d\theta = \delta_{l,l'} \delta_{m,m'}, \tag{3.5}$$

where $\delta_{m,m'}$ is the Kronecker delta function defined as

$$\delta_{m,m'} = \begin{cases} 1, & \text{if } m = m' \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

Spherical harmonics form the basis for any continuous function over $\mathbb{S}^2$ with finite energy. Therefore, a function $f : \mathbb{S}^2 \to \mathbb{R}$ can be rewritten using spherical harmonics

as

$$f(\theta,\phi) = \sum_{l} \sum_{m=-l}^{l} \hat{f}(l,m) Y_{l,m}(\theta,\phi), \tag{3.7}$$

where $\hat{f}(l,m)$ can be obtained using

$$\hat{f}(l,m) = \int_0^{\pi} \int_0^{2\pi} f(\theta,\phi) Y_l^m(\theta,\phi)^{\dagger} \sin\phi \, d\phi d\theta. \tag{3.8}$$

3.2.4 Spherical Convolution

Let f and g be the shape functions of the object and kernel respectively. Then f and g can be expressed as

$$f(\theta,\phi) = \sum_{l} \sum_{m=-l}^{l} \hat{f}(l,m) Y_{l,m}(\theta,\phi) \tag{3.9}$$

$$g(\theta,\phi) = \sum_{l} \sum_{m=-l}^{l} \hat{g}(l,m) Y_{l,m}(\theta,\phi), \tag{3.10}$$

where $Y_{l,m}$ is the $(l,m)^{th}$ spherical harmonics function and $\hat{f}(l,m)$ and $\hat{g}(l,m)$ are $(l,m)^{th}$ frequency components of f and g respectively. Then, the frequency components of convolution $f * g$ can be easily calculated as

$$\widehat{f * g}(l,m) = \sqrt{\frac{4\pi}{2l+1}} \hat{f}(l,m) \hat{g}(l,0)^{\dagger}, \tag{3.11}$$

where † denotes the complex conjugate.

3.2.5 3D Zernike Polynomials

3D Zernike polynomials are a complete and orthogonal set of basis functions in $\mathbb{B}^3$, that exhibits a *form invariance* property under 3D rotation. A $(n,l,m)^{th}$ order 3D Zernike basis function is defined as

$$Z_{n,l,m}(r,\theta,\phi) = R_{n,l}(r) Y_{l,m}(\theta,\phi), \tag{3.12}$$

where $R_{n,l}$ is the 3D Zernike radial polynomial defined as

$$R_{n,l}(r) = \sum_{v=0}^{(n-1)/2} q_{nl}^{v} r^{2v+l} \tag{3.13}$$

and q_{nl}^{v} is a scaler defined as

$$q_{nl}^{v} = \frac{(-1)^{\frac{(n-l)}{2}}}{2^{(n-l)}}\sqrt{\frac{2n+3}{3}}\binom{(n-l)}{\frac{(n-l)}{2}}(-1)^{v}\frac{\binom{\frac{(n-l)}{2}}{v}\binom{2(\frac{(n-l)}{2}+l+v)+1}{(n-l)}}{\binom{\frac{(n-l)}{2}+l+v}{\frac{(n-l)}{2}}}. \tag{3.14}$$

$Y_{l,m}(\theta,\phi)$ is the spherical harmonics function, $n \in \mathbb{Z}^{+}$, $l \in [0,n]$, $m \in [-l,l]$ and $n-l$ is even. Since 3D Zernike polynomials are orthogonal and complete in $\mathbb{B}^3$, an arbitrary function $f(r,\theta,\phi)$ in $\mathbb{B}^3$ can be approximated using Zernike polynomials as follows:

$$f(\theta,\phi,r) = \sum_{n=0}^{\infty}\sum_{l=0}^{n}\sum_{m=-l}^{l}\Omega_{n,l,m}(f)Z_{n,l,m}(\theta,\phi,r) \tag{3.15}$$

where $\Omega_{n,l,m}(f)$ can be obtained using

$$\Omega_{n,l,m}(f) = \int_0^1\int_0^{2\pi}\int_0^{\pi} f(\theta,\phi,r)Z_{n,l,m}^{\dagger}r^2\sin\phi\, drd\phi d\theta. \tag{3.16}$$

Furthermore, 3D Zernike polynomials hold the orthogonality property as follows:

$$\int_0^1\int_0^{2\pi}\int_0^{\pi} Z_{n,l,m}(\theta,\phi,r)Z_{n',l',m'}^{\dagger}r^2\sin\phi\, drd\phi d\theta = \frac{4\pi}{3}\delta_{n,n'}\delta_{l,l'}\delta_{m,m'}, \tag{3.17}$$

where δ is the Kronecker delta function. In Section 3.3, we will derive the proposed volumetric convolution using the concepts introduced above.

3.3 Volumetric Convolution

3.3.1 Problem Formulation

Convolution is an effective method to capture useful features from data represented over uniformly spaced grids points in $\mathbb{R}^n$, within each dimension of n. For example, gray scale images can be represented as intensities distributed over grid points in $\mathbb{R}^2$, spatio-temporal data and RGB images over grid points in $\mathbb{R}^3$, and stacked planar feature maps over grid points in $\mathbb{R}^n$. Given a shape function f and a convolutional kernel h, this process can be more formally represented as follows:

$$(f*g)(x) = \int_{\mathbb{R}^n} f(y)g(x-y)dy, \quad x,y \in \mathbb{R}^n. \tag{3.18}$$

In such cases, uniformity of the grid within each dimension ensures the translation equivariance of the convolution. However, for topological spaces such as $\mathbb{S}^2$ and $\mathbb{B}^3$, it is not possible to construct such a grid due to curvilinear geometry. This limitation is illustrated in Fig. 3.2. A naive approach to perform convolution in $\mathbb{B}^3$ would be to create a uniformly spaced three dimensional grid in (r,θ,ϕ) coordinates (with necessary padding) and use a regular 3D kernel. However, as shown in Fig. 3.2, the spaces between adjacent points in each axis are dependant on their absolute position.

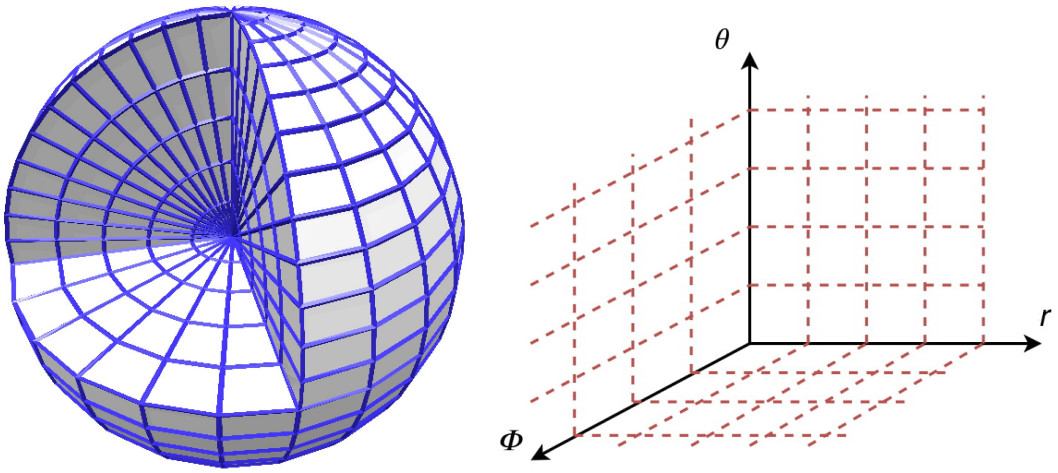


Figure 3.2: Grid representations in Spherical and Cartesian coordinates. *Left:* The space between grid points vary with r and from equator to poles. *Right:* A crude approach to represent the spherical grid with a uniformly spaced grid. This approach is inaccurate as spherical grids do not have uniform spacing.

Therefore, modeling such a space as a uniformly spaced grid is inaccurate.

To overcome these limitations, we propose a novel *volumetric convolution* operation which can effectively perform convolution on functions in $\mathbb{B}^3$. Volumetric convolution allows the convolution output to be a signal on $\mathbb{B}^3$, which opens up the possibility of achieving both rotation and radial translation equivariance with respect to the convolution operator.

3.3.2 Convolution of functions in $\mathbb{B}^3$

3.3.2.1 Convolution as a function on $\mathbb{SO}(3)$

Convolution in $\mathbb{B}^3$ can be achieved using two different approaches: 1) as a function on $\mathbb{S}^2$ or 2) as a function on $\mathbb{SO}(3)$. Cohen et al. [2018b] showed that for functions in $\mathbb{S}^2$, modeling convolution as a function on $\mathbb{SO}(3)$ improves the capacity of the network. However for functions in $\mathbb{B}^3$, following the same approach is hampered by implementation difficulties. More precisely, if modeled as a function on $\mathbb{SO}(3)$, for all $f, g \in \mathbb{B}^3$,

$$f * g(\alpha, \beta, \gamma) = \sum_{n} \sum_{l,m,m'} \hat{f}_{n,l,m} \hat{g}_{n,l,m'} D^{l}_{m,m'}(\alpha, \beta, \gamma), \tag{3.19}$$

where D is the Wigner D-matrix and α, β, γ are Euler angles. This relationship cannot be implemented as a matrix/tensor operation, since corresponding frequency components have to be extracted from spectral distributions and multiplied element-wise. Therefore, aiming for a more efficient implementation, we derive volumetric convolution as a function on $\mathbb{S}^2$, which is described in Section 3.3.2.2.

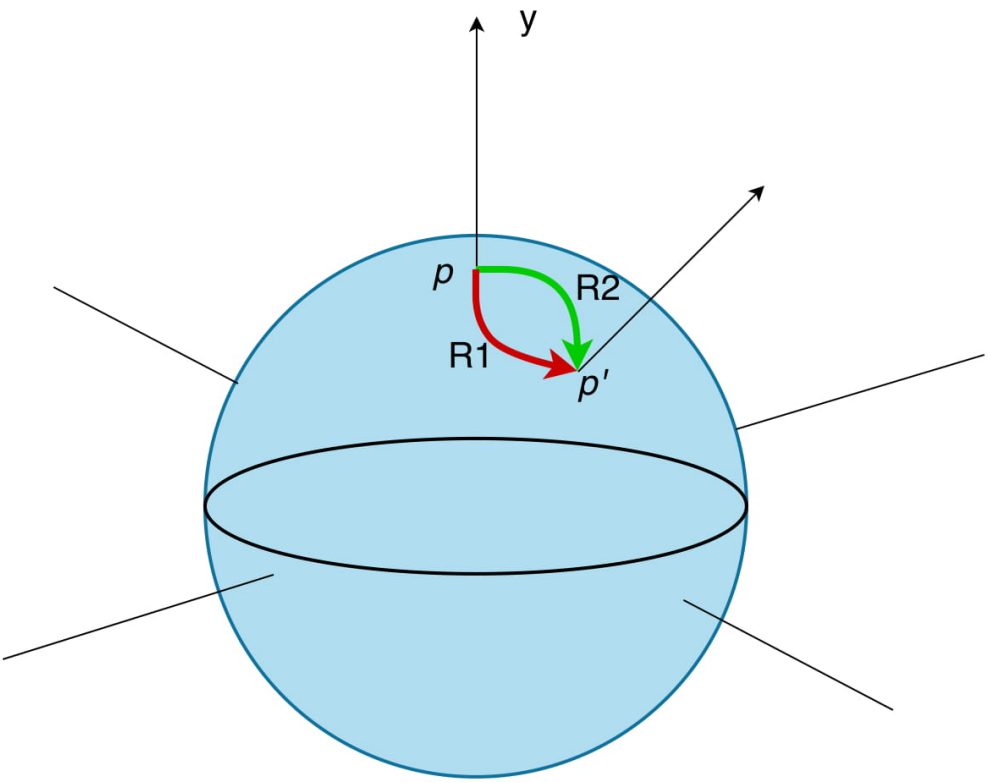


Figure 3.3: Consider the two rotations R_1 and R_2 which takes p to p'. Then R_1 and R_2 can be decomposed using Euler angles as $R_1 = R_y(\theta_1)R_x(\theta_2)R_y(\theta_3)$ and $R_2 = R_y(\theta_1)R_x(\theta_2)R_y(\theta_4)$, where the initial rotation around north pole is different in the two cases. Therefore, if the function is symmetric around north pole, the rotated function would only depend on p'.

3.3.2.2 Convolution as a function on $\mathbb{S}^2$

When performing convolution in $\mathbb{B}^3$ as a function on $\mathbb{S}^2$, a critical problem is that several rotation operations exist for mapping a point p to a particular point p'. For example, using Euler angles, we can decompose a rotation into three rotation operations $R(\theta,\phi) = R(\theta)_y R(\phi)_z R(\theta)_y$, and the first rotation $R(\theta)_y$ can differ while mapping p to p' (if y is the north pole) as shown in Fig. 3.3. However, if we enforce the kernel function to be symmetric around y, the function of the kernel after rotation would only depend on p and p'. This observation is important, as then we can uniquely define a 3D rotation of the kernel in terms of azimuth and polar angles.

Let the kernel be symmetric around y and $f(\theta,\phi,r)$, $g(\theta,\phi,r)$ be the functions of object and kernel respectively. Then we define volumetric convolution as

$$f * g(\alpha,\beta)\langle f(\theta,\phi,r), \tau_{(\alpha,\beta)}(g(\theta,\phi,r))\rangle = \int_0^1 \int_0^{2\pi} \int_0^{\pi} f(\theta,\phi,r), \tau_{(\alpha,\beta)}(g(\theta,\phi,r)) \sin\phi \, d\phi d\theta dr, \tag{3.20}$$

where $\tau_{(\alpha,\beta)}$ is an arbitrary rotation, that aligns the north pole with the axis towards

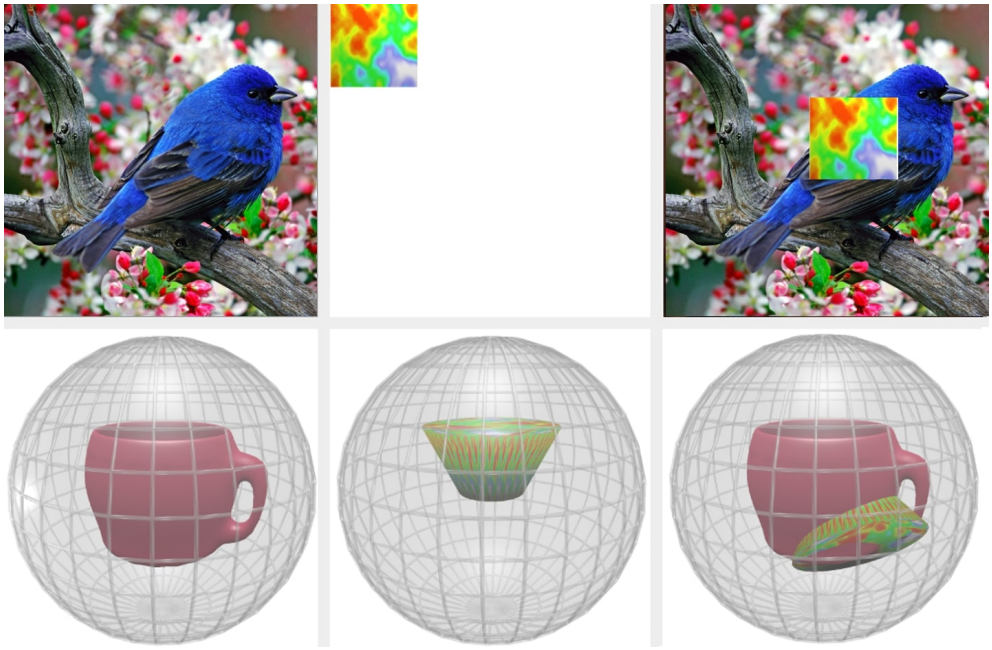


Figure 3.4: Analogy between planar and volumetric convolutions. Top *(left to right)*: 2D image, kernel and planar convolution in the Cartesian plane. Bottom *(left to right)*: 3D object, 3D kernel and volumetric convolution. In planar convolution the kernel translates and inner product between the image and the kernel is computed in (x, y) plane. In volumetric convolution, a 3D rotation and a radial translation are applied to the kernel and the inner product is computed between 3D function and 3D kernel over $\mathbb{B}^3$. This allows accurate encoding of shape and texture of 3D objects.

(α, β) direction (α and β are azimuth and polar angles respectively). Eq. 4.15 is able to capture more complex patterns compared to spherical convolution due to two reasons: *(1)* the inner product integrates along the radius and *(2)* the projection onto spherical harmonics forces the function into a polar function, that can result in information loss. It is important to note that the response of our convolution operator is a signal on $\mathbb{S}^2$, while the response of spherical convolution is a signal on 3D rotation group (Cohen et al. [2018b]). However, we extend our convolution operator to output a function on $\mathbb{B}^3$ in Section 3.5.1, which gives multiple advantages compared to Cohen et al. [2018b]. Fig. 3.4 shows the analogy between planar convolution, spherical convolution and volumetric convolution. In Section 3.3.3 we derive formulae—preserving differentiability—to obtain 3D Zernike moments for functions in $\mathbb{B}^3$.

3.3.3 Shape modeling of functions in $\mathbb{B}^3$ using 3D Zernike polynomials

Instead of using Eq. 3.16, we derive an alternative method to obtain the set $\left\{\Omega_{n,l,m}\right\}$. The motivation is two fold: *(1)* ease of computation and *(2)* the completeness property of 3D Zernike Polynomials ensures that $\lim_{n\to\infty} \parallel f - \sum_n \sum_l \sum_m \Omega_{n,l,m} Z_{n,l,m} \parallel = 0$

for any arbitrary function f. However, since n should be finite in practical implementation, aforementioned property may not hold, leading to an increased distance between the Zernike representation and the original shape. Therefore, minimizing the reconstruction error

$$\sum_{(\theta,\phi,r)\in\mathbb{B}^3} \left| \bar{f}(\theta,\phi,r) - f(\theta,\phi,r) \right|, \tag{3.21}$$

where $\bar{f}(\theta,\phi,r) = \sum_n \sum_l \sum_m \Omega_{n,l,m} Z_{n,l,m}$, $n \in [1,N]$ pushes the set $\left\{\Omega_{n,l,m}\right\}$ inside frequency space, where $\left\{\Omega_{n,l,m}\right\}$ has a closer resemblance to the corresponding shape. Following this conclusion, we derive the following method to obtain $\left\{\Omega_{n,l,m}\right\}$.

Since $Y_{l,m}(\theta,\phi) = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\phi) e^{im\theta}$, where $P_l^m(\cdot)$ is the associated Legendre function, it can be deduced that, $Y_{l,-m}(\theta,\phi) = (-1)^m Y_{l,m}^{\dagger}(\theta,\phi)$. Using this relationship we obtain $Z_{n,l,-m}(\theta,\phi,r) = (-1)^m Z_{n,l,m}^{\dagger}(\theta,\phi,r)$ and hence approximate Eq. 3.15 as

$$f(\theta,\phi,r) = \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=0}^{l} A_{n,l,m} Re\Big\{ Z_{n,l,m}(\theta,\phi,r) \Big\} + B_{n,l,m} Img\Big\{ Z_{n,l,m}(\theta,\phi,r) \Big\}, \tag{3.22}$$

where $Re\Big\{ Z_{n,l,m}(\theta,\phi,r) \Big\}$ and $Img\Big\{ Z_{n,l,m}(\theta,\phi,r) \Big\}$ are real and imaginary components of $Z_{n,l,m}$ respectively, and $A_{n,l,m}$ and $B_{n,l,m}$ are real valued constants to be calculated.

In practice, $f(\theta,\phi,r)$ is reconstructed using a limited number of sample points and a finite number of polynomials. Let N be the order of Zernike basis functions, K be the number of sample points and $\bar{f}$ be the reconstructed shape. The choice of K affects both computational efficiency and the modeling accuracy. For instance, a lower value of K increases the computational efficiency, but decreases modeling accuracy, and vice versa. With an appropriate choice of K and N, using Eq. 3.22, $\bar{f}$ can be approximated in matrix form as,

$$\bar{f} = Ua + Vb, \tag{3.23}$$

where,

$$\begin{aligned} \bar{f} = (f(\theta_1,\phi_1,r_1),\ldots,f(\theta_i,\phi_i,r_i),\ldots, \\ f(\theta_K,\phi_K,r_K))^T, (\theta_i,\phi_i,r_i) \in \mathbb{B}^3, \end{aligned} \tag{3.24}$$

$$a = (A_{0,0,0},\ldots,A_{n,l,m},\ldots A_{N,N,N})^T, \tag{3.25}$$

$$b = (B_{0,0,0},\ldots,B_{n,l,m},\ldots,B_{N,N,N})^T, \tag{3.26}$$

$\forall 0 \le m \le l \le n \le N$, and $n-l$ is even. U and V are matrices with $Re\Big\{ Z_{n,l,m}(\theta_i,\phi_i,r_i) \Big\}$

and $Img\Big\{Z_{n,l,m}(\theta_i,\phi_i,r_i)\Big\}$ as their entries respectively, as follows:

$$U = \begin{pmatrix} Re\Big\{Z_{0,0,0}(\theta_0,\phi_0,r_0)\Big\} & \dots & Re\Big\{Z_{n,l,m}(\theta_K,\phi_K,r_K)\Big\} \\ \vdots & \vdots & \vdots \\ Re\Big\{Z_{N,N,N}(\theta_0,\phi_0,r_0)\Big\} & \dots & Re\Big\{Z_{N,N,N}(\theta_K,\phi_K,r_K)\Big\} \end{pmatrix}, \tag{3.27}$$

$$V = \begin{pmatrix} Im\Big\{Z_{0,0,0}(\theta_0,\phi_0,r_0)\Big\} & \dots & Im\Big\{Z_{n,l,m}(\theta_K,\phi_K,r_K)\Big\} \\ \vdots & \vdots & \vdots \\ Im\Big\{Z_{N,N,N}(\theta_0,\phi_0,r_0)\Big\} & \dots & Im\Big\{Z_{N,N,N}(\theta_K,\phi_K,r_K)\Big\} \end{pmatrix}. \tag{3.28}$$

Let $X = (U, V)$ and $c = (a^T, b^T)^T$. Then, Eq. 3.23 can be rewritten as,

$$\bar{f} = Xc. \tag{3.29}$$

In other words, c is the approximated set of 3D Zernike moments $\{\Omega_{n,l,m}\}$. Eq. 3.29 can be interpreted as an overdetermined linear system, with the set $\{\Omega_{n,l,m}\}$ as the solution. To find the least squared error solution of the Eq. 3.29, we use the pseudo inverse of X. One easy option is to use a common non-differentiable approach like singular value decomposition to find the inverse X. However, this imposes the condition that the inputs to the volumetric convolution layer do not depend on any learnable function. To avoid imposing this condition and allow the volumetric convolution layer to be integrated to any deep network, we propose an alternative method. Li et al. [2011] proposed an iterative method to calculate the pseudo inverse of a matrix. They showed that V_n converges to A^+ where A^+ is the Moore-Penrose pseudo inverse of A if

$$V_{n+1} = V_n(3I - AV_n(3I - AV_n)), n \in \mathbb{Z}^+, \tag{3.30}$$

for a suitable initial approximation V_0. They also showed that a suitable initial approximation would be $V_0 = \alpha A^T$ with $0 < \alpha < 2/\rho(AA^T)$, where $\rho(\cdot)$ denotes the spectral radius. Empirically, we choose $\alpha = 0.001$ in our experiments. Next, we derive the theory of volumetric convolution within the unit sphere.

3.3.4 Convolution in $\mathbb{B}^3$ using 3D Zernike polynomials

We formally present our derivation of volumetric convolution using the following theorem.

Theorem 3.1. *Suppose $f, g : \mathbb{B}^3 \longrightarrow \mathbb{R}$ are square integrable complex functions defined in $\mathbb{B}^3$ so that $\langle f, f\rangle < \infty$ and $\langle g, g\rangle < \infty$. Further, suppose g is symmetric around north pole*

and $\tau(\alpha,\beta) = R_y(\alpha)R_z(\beta)$ *where* $R \in \mathbb{SO}(3)$. *Then,*

$$\begin{aligned}\int_0^1 \int_0^{2\pi} \int_0^{\pi} f(\theta,\phi,r), \tau_{(\alpha,\beta)}(g(\theta,\phi,r)) \sin\phi \, d\phi d\theta dr \\ \equiv \frac{4\pi}{3} \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f)\Omega_{n,l,0}(g)Y_{l,m}(\alpha,\beta),\end{aligned} \tag{3.31}$$

where $\Omega_{n,l,m}(f), \Omega_{n,l,0}(g)$ *and* $Y_{l,m}(\theta,\phi)$ *are* $(n,l,m)^{th}$ *3D Zernike moment of* f, $(n,l,0)^{th}$ *3D Zernike moment of* g, *and spherical harmonics function respectively.*

Proof. Completeness property of 3D Zernike Polynomials ensures that it can approximate an arbitrary function in $\mathbb{B}^3$, as shown in Eq. 3.15. Leveraging this property, Eq. 4.15 can be rewritten as

$$\begin{aligned}f * g(\alpha,\beta) = \langle \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) Z_{n,l,m}, \\ \tau_{(\alpha,\beta)} (\sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \sum_{m'=-l}^{l} \Omega_{n',l',m'}(g) Z_{n',l',m'}) \rangle.\end{aligned} \tag{3.32}$$

But since $g(\theta,\phi,r)$ is symmetric around y, the rotation around y should not change the function. Which ensures

$$g(r,\theta,\phi) = g(r,\theta-\alpha,\phi) \tag{3.33}$$

and hence,

$$\begin{aligned}\sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \sum_{m'=-l}^{l} \Omega_{n',l',m'}(g) R_{n',l'}(r) Y_{l',m'}(\theta,\phi) \\ = \sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \sum_{m'=-l}^{l} \Omega_{n',l',m'}(g) R_{n',l'}(r) Y_{l',m'}(\theta,\phi) e^{-im'\alpha}.\end{aligned} \tag{3.34}$$

This is true, if and only if $m' = 0$. Therefore, if $g(\theta,\phi,r)$ is a symmetric function around y, defined inside the unit sphere, it can be rewritten as

$$\sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \Omega_{n',l',0}(g) Z_{n',l',0}, \tag{3.35}$$

which simplifies Eq. 3.32 to

$$\begin{aligned}f * g(\alpha,\beta) = \langle \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) Z_{n,l,m}, \\ \tau_{(\alpha,\beta)} (\sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \Omega_{n',l',0}(g) Z_{n',l',0}) \rangle\end{aligned} \tag{3.36}$$

Using the properties of inner product, Eq. 3.36 can be rearranged as

$$f * g(\alpha, \beta) = \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) \Omega_{n',l',0}(g) \langle Z_{n,l,m}, \tau_{(\alpha,\beta)}(Z_{n',l',0}) \rangle. \tag{3.37}$$

Consider the term $\tau_{(\alpha,\beta)}(Z_{n',l',0})$. Then,

$$\begin{aligned} \tau_{(\alpha,\beta)}(Z_{n',l',0}(\theta, \phi, r)) &= \tau_{(\alpha,\beta)}(R_{n',l'}(r) Y_{l',0}(\theta, \phi)) \\ &= R_{n',l'}(r) \tau_{(\alpha,\beta)}(Y_{l',0}(\theta, \phi)) \\ &= R_{n',l'}(r) \sum_{m''=-l'}^{l'} Y_{l',m''}(\theta, \phi) D^{l'}_{m'',0}(\alpha, \beta, \cdot), \end{aligned} \tag{3.38}$$

where $D^{l}_{m,m'}$ is the Wigner D-matrix. But we know that $D^{l'}_{m'',0}(\alpha, \beta, \cdot) = Y_{l',m''}(\alpha, \beta)$. Then Eq. 3.37 becomes

$$\begin{aligned} f * g(\alpha, \beta) = \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{n'=0}^{\infty} \sum_{l'=0}^{n'} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) \Omega_{n',l',0}(g) \\ \sum_{m''=-l'}^{l'} Y_{l',m''}(\alpha, \beta) \langle Z_{n,l,m}, Z_{n',l',m''} \rangle, \end{aligned} \tag{3.39}$$

$$f * g(\alpha, \beta) = \frac{4\pi}{3} \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) \Omega_{n,l,0}(g) Y_{l,m}(\alpha, \beta), \tag{3.40}$$

which completes our proof.

□

3.3.5 Equivariance to 3D rotation group

One key property of the proposed volumetric convolution is its equivariance to 3D rotation group. To demonstrate this we present the following theorem.

Theorem 3.2. *Suppose $f, g : \mathbb{B}^3 \longrightarrow \mathbb{R}$ are square integrable complex functions defined in $\mathbb{B}^3$ such that $\langle f, f \rangle < \infty$ and $\langle g, g \rangle < \infty$. Also, let $\eta_{\alpha,\beta,\gamma}$ be a 3D rotation operator that can be decomposed into three Euler rotations $R_y(\alpha) R_z(\beta) R_y(\gamma)$ and $\tau_{\alpha,\beta}$ another rotation operator that can be decomposed into $R_y(\alpha) R_z(\beta)$. Suppose $\eta_{\alpha,\beta,\gamma}(g) = \tau_{\alpha,\beta}(g)$. Then,*

$$\eta_{(\alpha,\beta,\gamma)}(f) * g(\theta, \phi) = \tau_{(\alpha,\beta)}(f * g)(\theta, \phi), \tag{3.41}$$

where $*$ is the volumetric convolution operator.

Proof. Since $\eta_{(\alpha,\beta,\gamma)} \in \mathrm{SO}(3)$, we know that $\eta_{(\alpha,\beta,\gamma)}(f(x)) = f(\eta^{-1}_{(\alpha,\beta,\gamma)}(x))$. Also we

know that $\eta_{(\alpha,\beta,\gamma)} : \mathbb{R}^3 \to \mathbb{R}^3$ is an isometry. We define,

$$\langle \eta_{(\alpha,\beta,\gamma)} f, \eta_{(\alpha,\beta,\gamma)} g \rangle = \int_{B^3} f(\eta^{-1}_{(\alpha,\beta,\gamma)}(x)) g(\eta^{-1}_{(\alpha,\beta,\gamma)}(x)) dx. \tag{3.42}$$

Consider the Lebesgue measure $\lambda(B^3) = \int_{B^3} dx$. It can be proven that a Lebesgue measure is invariant under the isometries, which gives us $dx = d\eta_{(\alpha,\beta,\gamma)}(x) = d\eta^{-1}_{(\alpha,\beta,\gamma)}(x), \forall x \in B^3$. Therefore,

$$\begin{aligned} &\langle \eta_{(\alpha,\beta,\gamma)} f, \eta_{(\alpha,\beta,\gamma)} g \rangle = \langle f, g \rangle \\ &= \int_{S^3} f(\eta^{-1}_{(\alpha,\beta,\gamma)}(x)) g(\eta^{-1}_{(\alpha,\beta,\gamma)}(x)) d(\eta^{-1}_{(\alpha,\beta,\gamma)} x). \end{aligned} \tag{3.43}$$

Let $f(\theta, \phi, r)$ and $g(\theta, \phi, r)$ be the object function and kernel function (symmetric around north pole) respectively. Then volumetric convolution is defined as

$$f * g(\theta, \phi) = \langle f, \tau_{(\theta,\phi)} g \rangle. \tag{3.44}$$

Applying the rotation $\eta_{(\alpha,\beta,\gamma)}$ to f, we get

$$\eta_{(\alpha,\beta,\gamma)}(f) * g(\theta, \phi) = \langle \eta_{(\alpha,\beta,\gamma)}(f), \tau_{(\theta,\phi)} g \rangle \tag{3.45}$$

Using the result in Eq. 3.43, we have

$$\eta_{(\alpha,\beta,\gamma)}(f) * g(\theta, \phi) = \langle f, \eta^{-1}_{(\alpha,\beta,\gamma)}(\tau_{(\theta,\phi)} g) \rangle. \tag{3.46}$$

However, since $\eta_{\alpha,\beta,\gamma}(g) = \tau_{\alpha,\beta}(g)$, we get

$$\eta_{(\alpha,\beta,\gamma)}(f) * g(\theta, \phi) = \langle f, \tau_{(\theta-\alpha,\phi-\beta,)} g \rangle. \tag{3.47}$$

We know that,

$$f * g(\theta, \phi) = \langle f, \tau_{(\theta,\phi)} g \rangle = \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) \Omega_{n,l,0}(g) Y_{l,m}(\theta, \phi). \tag{3.48}$$

Then,

$$\begin{aligned} &\eta_{(\alpha,\beta,\gamma)}(f) * g(\theta, \phi) = \langle f, \tau_{(\theta-\alpha,\phi-\beta)} g \rangle \\ &= \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) \Omega_{n,l,0}(g) Y_{l,m}(\theta - \alpha, \phi - \beta) \\ &= (f * g)(\theta - \alpha, \phi - \beta) = \tau_{(\alpha,\beta)}(f * g)(\theta, \phi). \end{aligned} \tag{3.49}$$

Hence, we achieve equivariance over 3D rotations. □

In simple terms, the theorem states that if a 3D rotation is applied to a function defined in $\mathbb{B}^3$ Hilbert space, the output feature map after volumetric convolution

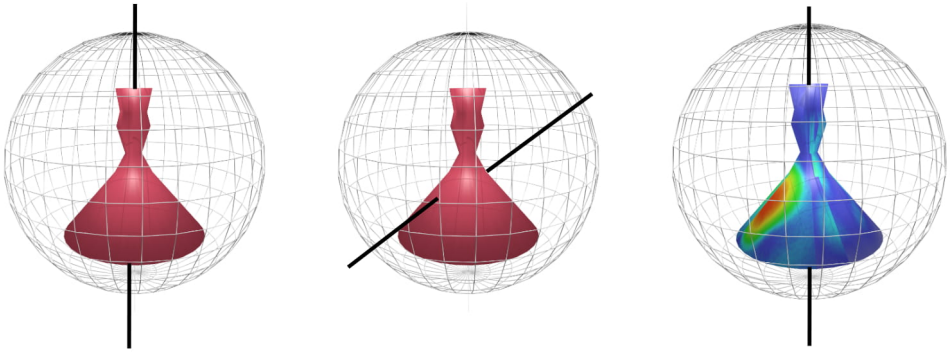


Figure 3.5: Three cases of axial symmetry: *left:* axial symmetry measurement is high, as both point values and overall shape of the function are symmetric around the axis. *Middle:* axial symmetry measurement is low, as overall shape of the function is not symmetric around the axis. *Right:* axial symmetry measurement is low, as point values of the function are not symmetrically distributed around the axis.

exhibits the same rotation. The output feature map however, is symmetric around north pole, hence the rotation can be uniquely defined in terms of azimuth and polar angles. In Section 3.4 we derive the axial symmetry measure of a function in $\mathbb{B}^3$ around an arbitrary axis using 3D Zernike polynomials.

3.4 Axial symmetry measure of a function in $\mathbb{B}^3$ around an arbitrary axis

In this section we present the following proposition to obtain the axial symmetry measure of a function in $\mathbb{B}^3$ around an arbitrary axis using 3D Zernike polynomials. An illustration of axial symmetry measurement is shown in Fig. 3.5.

Proposition 3.1. *Suppose $g : \mathbb{B}^3 \longrightarrow \mathbb{R}^3$ is a square integrable complex function defined in $\mathbb{B}^3$ such that $\langle g, g \rangle < \infty$. Then, the power of projection of g in to $S = \{Z_i\}$ where S is the set of Zernike basis functions that are symmetric around an axis towards (α, β) direction is given by*

$$\| \, sym_{(\alpha,\beta)} \left[g(\theta, \phi, r) \right] \, \| = \sum_{n} \sum_{l=0}^{n} \| \sum_{m=-l}^{l} \Omega_{n,l,m} Y_{l,m}(\alpha, \beta) \, \|^2, \tag{3.50}$$

where α and β are azimuth and polar angles respectively.

Proof. The subset of complex functions which are symmetric around north pole is $S = \left\{ Z_{n,l,0} \right\}$. Therefore, projection of the function into S gives

$$sym_y \left[g(\theta, \phi, r) \right] = \sum_{n} \sum_{l=0}^{n} \langle f, Z_{n,l,0} \rangle Z_{n,l,0}(\theta, \phi, r). \tag{3.51}$$

To obtain the symmetry function around any axis which is defined by (α, β), we rotate the function by $(-\alpha, -\beta)$, project into S, and finally compute the power of the projection

$$sym_{(\alpha,\beta)}\left[g(\theta,\phi,r)\right] = \sum_{n,l}\langle\tau_{(-\alpha,-\beta)}(f), Z_{n,l,0}\rangle Z_{n,l,0}(\theta,\phi,r). \tag{3.52}$$

For any rotation operator τ, and for any two points defined on a complex Hilbert space, x and y,

$$\langle\tau(x),\tau(y)\rangle_H = \langle x,y\rangle_H. \tag{3.53}$$

Applying this property to Eq. 3.52 gives

$$sym_{(\alpha,\beta)}\left[g(\theta,\phi,r)\right] = \sum_{n,l}\langle f, \tau_{(\alpha,\beta)}(Z_{n,l,0})\rangle Z_{n,l,0}(\theta,\phi,r). \tag{3.54}$$

Using Eq. 3.15 we get

$$\begin{aligned} sym_{(\alpha,\beta)}\left[g(\theta,\phi,r)\right] = \sum_{n}\sum_{l=0}^{n}\langle\sum_{n'}\sum_{l'=0}^{n'}\sum_{m'=-l'}^{l'}\Omega_{n'l'm'}Z_{n',l',m'}, \\ \tau_{(\alpha,\beta)}(Z_{n,l,0})\rangle Z_{n,l,0}(\theta,\phi,r). \end{aligned} \tag{3.55}$$

Using properties of inner product Eq. 3.55 further simplifies to

$$\begin{aligned} sym_{(\alpha,\beta)}\left[g(\theta,\phi,r)\right] = \sum_{n}\sum_{l=0}^{n}\sum_{n'}\sum_{l'=0}^{n'}\sum_{m'=-l'}^{l'}\Omega_{n'l'm'}\langle Z_{n',l',m'}, \\ \tau_{(\alpha,\beta)}(Z_{n,l,0})\rangle Z_{n,l,0}(\theta,\phi,r). \end{aligned} \tag{3.56}$$

Using the same derivation as in Eq. 3.38,

$$\begin{aligned} sym_{(\alpha,\beta)}\left[g(\theta,\phi,r)\right] = \sum_{n}\sum_{l=0}^{n}\sum_{n'}\sum_{l'=0}^{n'}\sum_{m'=-l'}^{l'}\Omega_{n'l'm'} \\ \sum_{m''=-l}^{l}Y_{l,m''}(\alpha,\beta)\langle Z_{n',l',m'}, Z_{n,l,m''}\rangle Z_{n,l,0}(\theta,\phi,r). \end{aligned} \tag{3.57}$$

Since 3D Zernike Polynomials are orthogonal we get

$$\begin{aligned} &sym_{(\alpha,\beta)}\left[g(\theta,\phi,r)\right] \\ &\quad = \frac{4\pi}{3}\sum_{n}\sum_{l=0}^{n}\sum_{m=-l}^{l}\Omega_{n,l,m}Y_{l,m}(\alpha,\beta)Z_{n,l,0}(\theta,\phi,r). \end{aligned} \tag{3.58}$$

In signal theory the power of a function is taken as the integral of the squared

function divided by the size of its domain. Following this we get

$$\begin{aligned} &\| \, sym_{(\alpha,\beta)} \left[g(\theta,\phi,r) \right] \, \| \\ &\qquad = \langle (\sum_{n} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m} Y_{l,m}(\alpha,\beta)) Z_{n,l,0}(\theta,\phi,r), \\ &\qquad (\sum_{n'} \sum_{l'=0}^{n'} \sum_{m'=-l'}^{l'} \Omega_{n',l',m'} Y_{l',m'}(\alpha,\beta) Z_{n',l',0}(\theta,\phi,r))^{\dagger} \rangle. \end{aligned} \tag{3.59}$$

We drop the constants here since they do not depend on the frequency. Simplifying Eq. 3.59 gives

$$\begin{aligned} \| \, sym_{(\alpha,\beta)} \left[g(\theta,\phi,r) \right] \, \| = \sum_{n} \sum_{l=0}^{n} \sum_{m=-l}^{l} \sum_{m'=-l}^{l} \Omega_{n,l,m} Y_{l,m}(\alpha,\beta) \\ \Omega_{n,l,m'} Y_{l',m}(\alpha,\beta), \end{aligned} \tag{3.60}$$

which leads to

$$\| \, sym_{(\alpha,\beta)} \left[g(\theta,\phi,r) \right] \, \| = \sum_{n} \sum_{l=0}^{n} \| \sum_{m=-l}^{l} \Omega_{n,l,m} Y_{l,m}(\alpha,\beta) \, \|^2 . \tag{3.61}$$

which completes our proof.

□

Using our derivation, one can obtain the distribution of symmetry the object has around a set of axes. However, to compare two objects by the amount of symmetry it has around a specific axis, it is needed to normalize the symmetry measurement by dividing the final result with the norm of the unprojected function.

3.5 A case study: Representation Learning on 3D objects

A 2D image is a function on Cartesian plane, where a unique value exists for any (x, y) coordinate. Similarly, a polar 3D object can be expressed as a function on the surface of the sphere, where any direction vector (θ, ϕ) has a unique value. To be precise, a 3D polar object has a boundary function in the form of $f : \mathbb{S}^2 \rightarrow [0, \infty]$.

Translation of the convolution kernel on (x, y) plane in 2D case, extends to movements on the surface of the sphere in $\mathbb{S}^2$. If both the object and the kernel have polar shapes, this task can be tackled by projecting both the kernel and the object onto spherical harmonic functions. However, using spherical convolution to capture features from 3D point clouds entail three critical limitations. *First*, the projection of the points on to the surface of the sphere smoothens the overall shape in to a polar one. In other words, since it formulates the shape as a function on $(\theta.\phi)$, it restricts the representation of complex (non-polar) objects. An illustration of 2D polar and non-polar shapes is shown in Fig. 3.6. *Second*, the integration happens over the surface

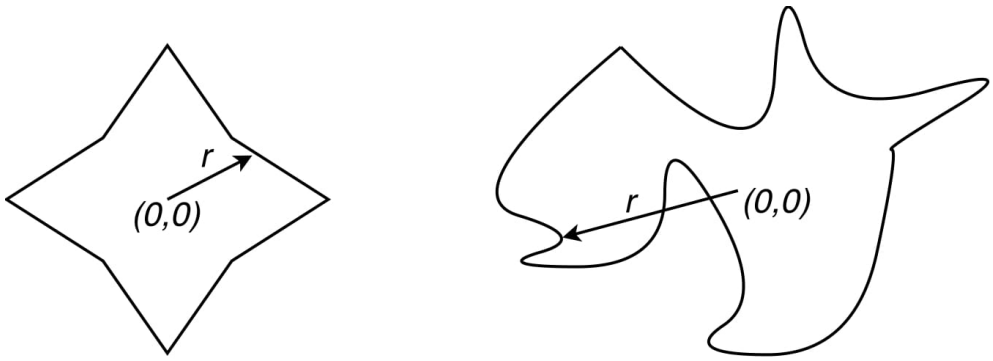


Figure 3.6: A 2D illustration of polar and non-polar shapes.

of the sphere, which is unable to capture patterns across radius. *Third*, spherical convolution is equivariant to only 3D rotation group.

These limitations can be addressed by representing and convolving the shape function inside the unit ball ($\mathbb{B}^3$). Representing the object function inside $\mathbb{B}^3$ allows the function to keep its complex shape information without any deterioration since each point is mapped to unique coordinates (r, θ, ϕ), where r is the radial distance, θ and ϕ are azimuth and polar angles respectively. Additionally, it allows encoding of 2D texture information simultaneously. The volumetric convolution can also achieve equivariance to both 3D rotation and radial translation of local patterns. Fig. 4.1 compares volumetric convolution and spherical convolution.

We conduct experiments on 3D objects with uniform surface values, therefore in this work we use the following transformation to apply a simple surface function to the 3D objects:

$$f(\theta, \phi, r) = \begin{cases} r, & \text{if surface exists at } (\theta, \phi, r) \\ 0, & \text{otherwise.} \end{cases} \tag{3.62}$$

3.5.1 Equivariance to 3D radial translation

Consider the case where the kernel is shifted along the radius and then convolved with the input function. Let $R_{n,l}$ be the linear component of the Zernike polynomial. Then, if we consider only the linear component, shifting the kernel by r' and then

convolving with the input function gives,

$$\begin{aligned}\int_0^1 R_{nl}(r)R_{n'l}(r-r')r^2dr \\ &= \int_0^1 \sum_{v=0}^{\frac{n-l}{2}} q_{nl}^v r^{2v+l} \sum_{v'=0}^{\frac{n'-l}{2}} q_{n'l}^{v'}(r-r')^{2v'+l}r^2dr \\ &= \sum_{v=0}^{\frac{n-l}{2}} q_{nl}^v \sum_{v'=0}^{\frac{n'-l}{2}} q_{n'l}^{v'} \int r^{2v+l}(r-r')^{2v'+l}r^2dr, \end{aligned} \tag{3.63}$$

which produces the result

$$\sum_{v=0}^{\frac{n-l}{2}} q_{nl}^v \sum_{v'=0}^{\frac{n'-l}{2}} q_{n'l}^{v'} \frac{(-r')^{l+2v'} 2F_1\left[-l-2v', 3+l+2v; 4+l+2v; \frac{1}{r'}\right]}{3+l+2v}, \tag{3.64}$$

where $2F_1$ is the hypergeometric function. This complex relationship hampers achieving equivariance to 3D translation directly using properties of 3D Zernike Polynomials, preserving differentiability. Hence, we follow an alternative approach to achieve this task which is explained below.

Let's consider the input function $f(\theta_i, \phi_i, r_i), \forall(\theta_i, \phi_i, r_i) \in \mathbb{B}^3$. Then, let us define q_k as,

$$q_k = 0.1k, \forall 0 \leq k < 10, k \in \mathbb{Z}. \tag{3.65}$$

Next, we extract the sets of points $f'_k \in f(\theta_i, \phi_i, r_k)$, $\forall q_k < r_k < q_{k+1}$. Then, let's consider the convolution kernel $g(\theta_i, \phi_i, r_i)$, $\forall(\theta_i, \phi_i, r_i) \in \mathbb{B}^3$. We take the radially translated kernel,

$$Tr_{(q_k)}\Big[g(\theta_i, \phi_i, r_i)] = g(\theta_i, \phi_i, r_i - q_k), \tag{3.66}$$

where, $0 \leq r_i - q_k < 1$. Here, $Tr_{(q_k)}\Big[\cdot]$ is radial translation by q_k.

Finally, we perform convolution between f'_k and $Tr_{(q_k)}\Big[g]$ for each k, as graphically illustrated in Fig. 3.7, which extends the response of our convolution operator to $\mathbb{B}^3$ as,

$$(f'_k * g)(\alpha, \beta, q_k) = f'_k * Tr_{(q_k)}\Big[\tau_{(\alpha,\beta)}g]. \tag{3.67}$$

Convolving the aforementioned point sets individually with corresponding radially translated kernel values allows us to share weights along radius, in other words, achieve equivariance over 3D radial translation for local feature patterns. Furthermore, the output of the convolution gives us a dense representation in $\mathbb{B}^3$, as illustrated in Fig. 3.8. Equivariance to 3D radial translation can be more formally illustrated as follows.

Let $p = f(\theta_i, \phi_i, r_i), \forall(\theta_i, \phi_i, r_i) \in P$, where P is a set of points which belongs to a local feature pattern of a function in $\mathbb{B}^3$. Then, we perform convolution on p with a kernel h,

$$(p * h)(\alpha, \beta, q_k) = p * Tr_{(q_k)}\Big[\tau_{(\alpha,\beta)}h]. \tag{3.68}$$

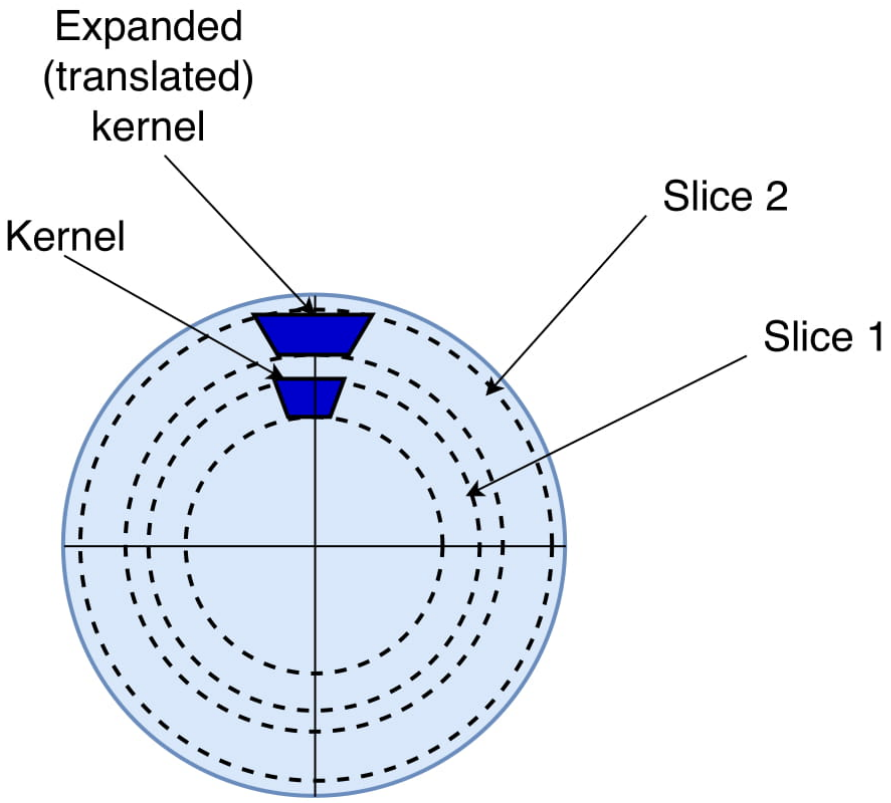


Figure 3.7: Weight sharing across radius.

Suppose we translate the local feature pattern radially. Then,

$$(Tr_{r'}\Big[p] * h)(\alpha, \beta, q_k) = (p(r - r') * h(r))(\alpha, \beta, q_k) \tag{3.69}$$

Let $r'' = r - r'$. Then,

$$\begin{aligned}(Tr_{r'}\Big[p] * h)(\alpha, \beta, q_k) &= (p(r'') * h(r' + r''))(\alpha, \beta, q_k), \\ &= (p(r'') * h(r''))(\alpha, \beta, q_k - r'), \\ &= Tr_{(r')}\Big[(p(r'') * h(r''))(\alpha, \beta, q_k)].\end{aligned} \tag{3.70}$$

Hence, we achieve equivariance over 3D radial translation of local patterns. The intuition behind the aforementioned process is that if a specific shape attribute of the object (not necessarily the whole object) translates along the radius, the corresponding output feature pattern of would also translate along the radius of the output feature map, which is in $\mathbb{B}^3$. A practical requirement to achieve this equivariance is that the kernel should cover approximately the same area as the local pattern. We achieve this requirement by designing the kernel as a concentrated set of points over a limiterd area, in the spatial domain.

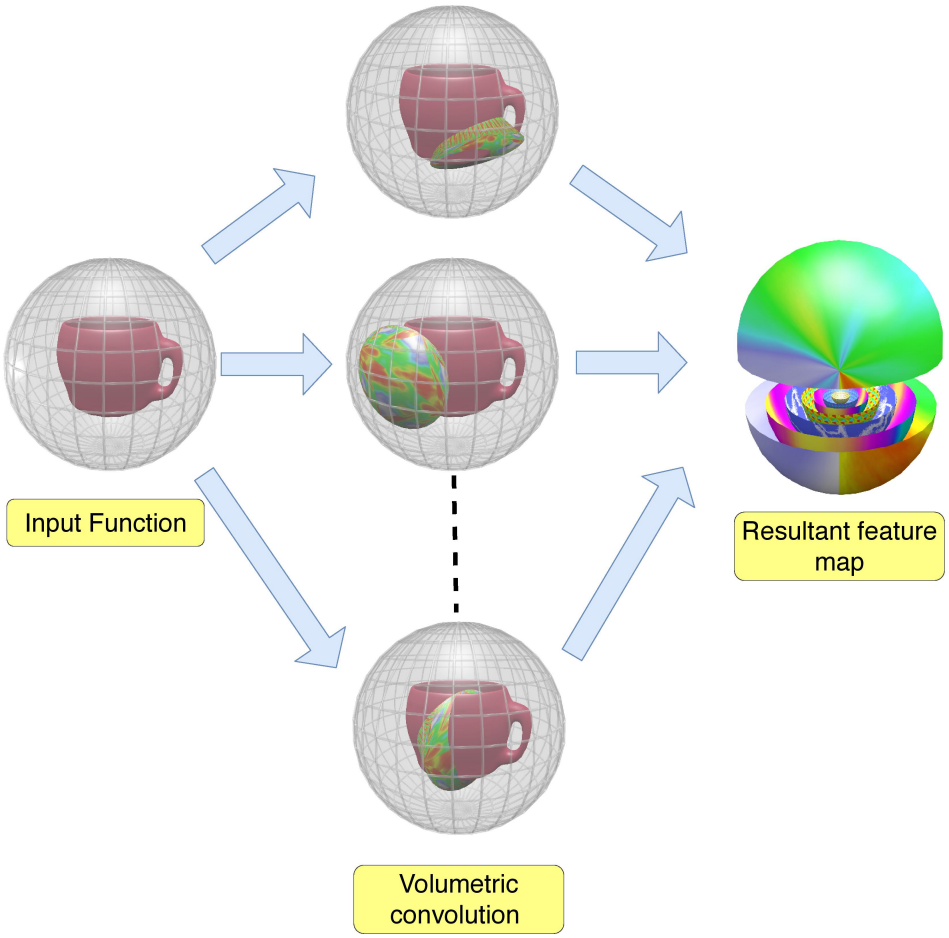


Figure 3.8: Illustration of volumetric convolution with weight sharing across radius. For the sake of clarity, this illustration only shows a single convolutional kernel. We bisect and show a cross section of the resultant feature map on right for better visualization. In the resultant feature map, each spherical heatmap corresponds to the response at a specific translation of the kernel. Each value in a spherical heatmap corresponds to the response at a specific 3D orientation of the kernel at a specified translation. Therefore, the resultant feature map is a signal on $\mathbb{B}^3$, which allows us to achieve equivariance over 3D rotation and radial translation of local patterns.

3.5.2 Adaptive Weighted Frequency Pooling

Feature pooling helps in aggregating information in spatial or filter response domain. Although feature pooling is an established mechanism in spatial domain, frequency domain pooling is largely an unsolved problem. Here, we propose a simple frequency pooling approach that fuses information across different frequencies to learn more compact and discriminative features.

Let us reconsider the proposed volumetric convolution formula at a specific translation of the kernel,

$$f * g(\alpha, \beta) \equiv \frac{4\pi}{3} \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) \Omega_{n,l,0}(g) Y_{l,m}(\alpha, \beta). \tag{3.71}$$

As evident from the above formula, the response is also in spatial domain and is a signal on S^2. However, any signal on S^2 can be completely characterized by its corresponding spherical harmonic frequencies. To leverage this property, we rearrange Equation 3.71 as follows,

$$f * g(\alpha, \beta) \equiv \frac{4\pi}{3} \sum_{l=0}^{n} \sum_{m=-l}^{l} \left(\sum_{n=0}^{\infty} \Omega_{n,l,m}(f) \Omega_{n,l,0}(g) \right) Y_{l,m}(\alpha, \beta). \tag{3.72}$$

It is obvious that $S_{l,m} = \left(\sum_{n=0}^{\infty} \Omega_{n,l,m}(f) \Omega_{n,l,0}(g) \right)$ represents $(l,m)^{th}$ spherical harmonics frequency of the response of volumetric convolution. Since, in practice we use $n = 6$, the set $\{S_{l,m}\}, \forall (m,l)$, where $0 \leq l \leq 5$ and $-l \leq m \leq l$, encodes all the shape information in a low dimensional vector, compared to spatial domain representation. Therefore, instead of spatial domain representation, we connect the spectral representation to the fully connected layer.

Furthermore, it can be observed that the set $\{S_{l,m}\}$ is within the linear span of $\Omega_{n,l,m}(f) \cup \Omega_{n,l,0}(g)$. Therefore, instead of calculating $\{S_{l,m}\}$ in a precise manner, we take the outer product between $\Omega_{n,l,m}(f)$ and $\Omega_{n,l,0}(g)$ to get a dense frequency map Ω as follows:

$$\Omega = \Omega_{n,l,m}(f)(\Omega_{n,l,0}(g)^T \tag{3.73}$$

where $\Omega \in \mathbb{R}^{(100 \times 100)}$ and $\Omega_{n,l,m}(f), \Omega_{n,l,0}(g) \in \mathbb{R}^{(100 \times 1)}$. Then, we obtain two dense weighted frequency maps, $F_1 \in \mathbb{R}^{(100 \times 100)}$ and $F_2 \in \mathbb{R}^{(100 \times 100)}$, by

$$F_1 = \Omega \circ W_1, \text{ and } F_2 = \Omega \circ W_2, \tag{3.74}$$

where $\circ$ is the Hadamard product and $W_1, W_2 \in \mathbb{R}^{(100 \times 100)}$ are trainable weights. Next, we take row-wise and column-wise sum of F_1 and F_2 to obtain two vectors $v_1 \in \mathbb{R}^{(100 \times 1)}$ and $v_2 \in \mathbb{R}^{(100 \times 1)}$:

$$v_1 = F_1 u^T, \text{ and } v_2 = (u F_2)^T, \tag{3.75}$$

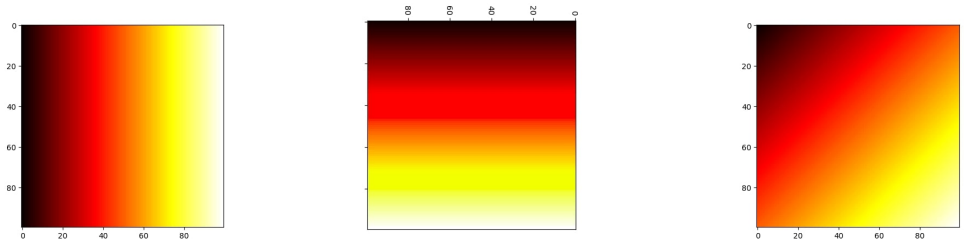


Figure 3.9: The heat-maps of the dense frequency map. *Left:* frequency heat-map with respect to kernel. *Middle:* frequency heat-map with respect to input function. *Right:* frequency The resultant heat-map of the dense frequency map.

where $u \in \mathbb{R}^{(1\times 100)}$ is a vector of ones. Although neither v_1 or v_2 is an exact replica of $\{S_{l,m}\}$, we have observed that empirically, this step increases the capacity of the network and makes it more robust to random movements of feature patterns. Our intuition for this behaviour is as follows: in practice, there may be other frequency components in $\Omega_{n,l,m}(f) \cup \Omega_{n,l,0}(g)$, other than $\{S_{l,m}\}$, which are invariant to certain pattern movements. Also, most discriminative and robust features may belong to certain frequency bands, and weighted sum of $F_{100\times 100}$ allows us to give more emphasis to such prominent frequency bands.

3.5.3 Experimental Architectures

In this section, we present two experimental architectures to demonstrate the usefulness of volumetric convolution in 3D object recognition tasks. The two types of architectures considered here are with a *single convolution layer* and *multi-convolution layers* respectively. Between these two types, single convolution layer showed better classification performance on popular object datasets with simple 3D shapes, as reported in Section 3.6.4. In contrast, the multi-convolution layer architecture shows better performance for complex 3D shapes, as demonstrated in the same section.

3.5.3.1 Single convolution layer architecture

In this architecture, the object is initially fed to a volumetric convolution layer with 16 kernels. Each kernel is translated 10 times as mentioned in Section 3.5.1, which gives a total of 160 kernels. We use $n = 6$ to implement Eq. 3.40, which gives 100 dimensional vectors $\Omega_{n,l,m}$ and $\Omega_{n,l,0}$ to represent the input object and each kernel respectively. Convolving input with 16 kernels results in $160 \times 100 \times 100$ dimensional output feature map, since we take the outer product between $\Omega_{n,l,m}$ and $\Omega_{n,l,0}$ as explained in Section 3.5.2. Afterwards, we perform frequency pooling in two orthogonal directions which reduces the dimensionality of the feature map to $160 \times 100 \times 2$. The output of the frequency pooling layer is then fed to a fully connected layer for classification. We

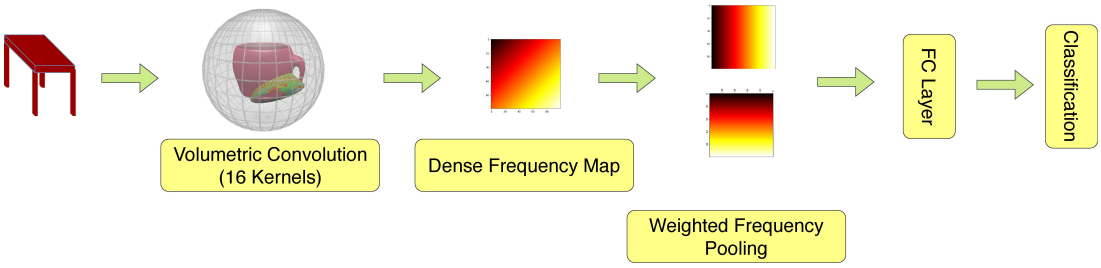


Figure 3.10: The overall experimental architecture.

do not use any non-linearity in this single convolution layer architecture. The overall experimental architecture is shown in Fig. 4.2.

3.5.3.2 Multi-convolution layer architecture

In the multi-convolution layer architecture, the penultimate layer operates similar to the explanation in Section 3.5.3.1, while the operation of other (intermediate) convolution layers differs slightly. The main difference is that both the input and the output of an intermediate convolution layer are in spatial domain, as opposed to the penultimate layer. Let there be N kernels for an intermediate convolution layer. Then, we calculate Zernike moments for both input function and kernels, and perform convolution as per Eq. 3.40. From the output, we sample 300 equi-spaced points for each θ and ϕ direction in the angular space, where $0 < \theta < 2\pi$ and $0 < \phi < \pi$. To sample points in r direction, we translate each kernel 10 times by an amount of 0.1, and perform convolution for each translated state. This overall procedure results in N output feature maps in $\mathbb{B}^3$, where each feature map has $10 \times 300 \times 300$ sampled points in the spatial domain. These feature maps are then fed to a ReLU layer, before being convolved again by the next convolution layer. We apply adaptive frequency pooling only to the penultimate layer, as we do not revert to spatial domain after that.

For both architectures, we use three iterations to calculate the Moore-Penrose pseudo inverse using Eq. 3.30. We use a decaying learning rate $lr = 0.1 \times 0.9^{\frac{g_{step}}{3000}}$, where g_{step} is incremented by one per iteration. For training, we use the Adam optimizer with hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$. All the weights are initialized using a random normal distribution with 0 mean and 0.5 standard deviation. All these values are chosen empirically. Since we have decomposed the theoretical derivations into sets of low-cost matrix multiplications, specifically aiming to reduce the computational complexity, the GPU implementation is highly efficient. For example, the model takes less than 25 minutes for an epoch during the training phase for ModelNet10, with a batch size 2, on a single GTX 1080Ti GPU.

3.6 Experiments

In this section, we discuss and evaluate the performance of the proposed approach on 3D object recognition and retrieval tasks. We first apply our experimental architecture on five recent datasets, and compare the performance with relevant state-of-the-art works. An extensive ablation study is also reported. We then evaluate the robustness of the captured features against loss of information and finally show that the proposed approach for computing 3D Zernike moments produce richer representations of 3D shapes compared to the conventional approach.

3.6.1 Datasets

− **ModelNet40** contains 40 object categories and a total of 12,311 CAD models. Train and test sets originally contain 9,843 and 2,468 models respectively. We use the standard train/test split to evaluate our model.
− **ModelNet10** is closely related to ModelNet40 dataset, and contains 10 object classes. We use the original train/test split provided by the authors of the dataset, which contains 3991 models for training and 908 models for testing.
− **McGill shape dataset** is a benchmark 3D shape dataset with 10 classes: ant, crab, spectacle, hand, human, octopus, plier, snake, spider and teddy-bear. The dataset contains a total of 255 objects with a variety of pose changes and part articulations.
− **SHREC'17 dataset** is a challenging state-of-the-art 3D object dataset. This large scale dataset contains about 51,300 3D models over 55 common categories. Each category is subdivided into several subcategories, but we use only the main 55 categories in our experiments. We use the original split by the authors which is 70%-30% for train and test respectively.

3.6.2 3D object classification

One key feature of our proposed volumetric convolution is that it is a natural extension of planar convolution to spherical domain (specifically $\mathbb{B}^3$). In the same way as a planar kernel finds distributed discriminative patterns across (x, y) plane, volumetric convolution is able to find such patterns distributed across the 3D space. Practically, this should enable our model to capture rich features with less number of layers, compared to other state-of-the-art models, that are somewhat ad-hoc extensions to 3D domain. To demonstrate this, we present the model complexity and accuracy analysis on ModelNet10 and ModelNet40 datasets. Table 3.2 shows the results on ModelNet10. Our model achieves an accuracy of 93.8% over ModelNet10 with only three trainable layers: one convolution layer, one frequency pooling layer and one fully connected layer. Our accuracy is the third highest, below SO-Net and Kd-Networks. Compared to models such as VRN and PairWise, which have 45 and 23 convolution layers respectively, our model achieves a higher accuracy with a significantly less number of layers. This clearly demonstrates the richness of computed features by volumetric convolution. Table 4.2 shows the results over ModelNet40. Our model achieves an accuracy of 91.0% and ranks third, same as the case of ModelNet10. These

Table 3.2: Comparison with state-of-the-art on ModelNet10 (ranked according to performance).

Method	Trainable layers	Trainable Params	ModelNet10
SO-Net (CVPR'18) Li et al. [2018b]	11FC	60M	95.7%
Kd-Networks (ICCV'17) Klokov and Lempitsky [2017]	15KD	4M	94.0%
Ours	**(1Conv, 1Adapt. FreqPool, 1FC)**	**0.7M**	**93.8%**
VRN (NIPS'16) Brock et al. [2016]	45Conv	90M	93.11%
Pairwise (CVPR'16) Johns et al. [2016]	23Conv	143M	92.8%
DeepPano (SPL'15) Shi et al. [2015]	(4Conv, 3FC)	-	85.45%
3DShapeNets (CVPR'15) Wu et al. [2015]	(4-3DConv,2FC)	38M	83.5%
PointNet (IJCNN'16) Garcia-Garcia et al. [2016]	(2Conv, 2FC)	<1M	77.6%

Table 3.3: Comparison with state-of-the-art on ModelNet40 (ranked according to performance).

Method	Trainable layers	Trainable Params	ModelNet40
SO-Net (CVPR'18) Li et al. [2018b]	11FC	60M	93.4%
Kd-Networks (ICCV'17) Klokov and Lempitsky [2017]	15KD	4M	91.8%
Ours	**(1Conv, 1Adapt. FreqPool, 1FC)**	**0.7M**	**91.0%**
VRN (NIPS'16) Brock et al. [2016]	45Conv	90M	90.8%
Pairwise (CVPR'16) Johns et al. [2016]	23Conv	143M	90.7%
MVCNN (ICCV'16)n Su et al. [2015]	(60Conv, 36FC)	200M	90.1%
PointNet (CVPR'17) Qi et al. [2017a]	(5Conv, 2STL)	80M	86.2%
ECC (CVPR'17) Simonovsky and Komodakis [2017]	(4Conv, 1FC)	-	83.2%
DeepPano (SPL'15) Shi et al. [2015]	(4Conv, 3FC)	-	77.63%
3DShapeNets (CVPR'15) Wu et al. [2015]	(4-3DConv, 2FC)	38M	77%

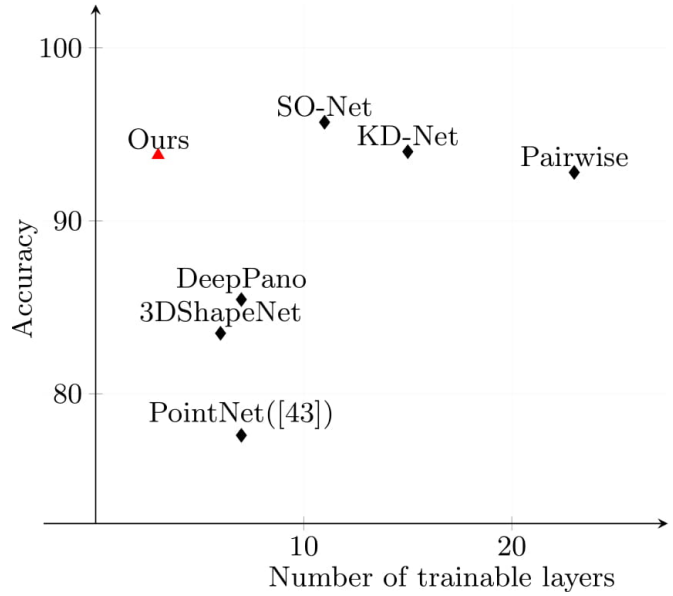


Figure 3.11: Accuracy comparison with state-of-the-art over ModelNet10 against the number of trainable layers.

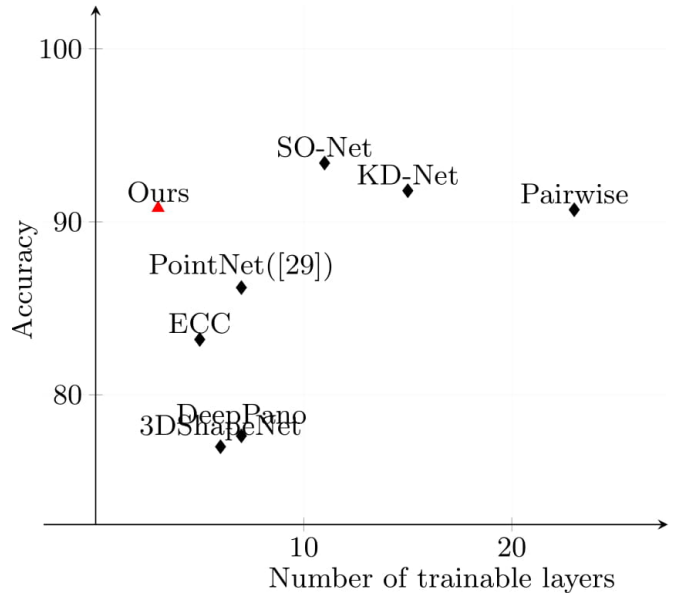


Figure 3.12: Accuracy comparison with state-of-the-art over ModelNet40 against the number of trainable layers.

results demonstrate that our model has a good generalization over a large number of object categories, without losing its advantage as a rich feature computer. Also, our model has only 0.7M trainable parameters, which is a drastically lower compared to state-of-the-art. This significant reduction in number of parameters is a fair indication of the effectiveness of our *adaptive-frequency-pooling* layer.

Table 3.4: 3D object retrieval results comparison with state-of-the-art on McGill Dataset.

Method	Accuracy
Tabia et al. [2014]	0.977%
Agathos et al. [2009]	0.976%
Tabia et al. [2013]	0.969%
Papadakis et al. [2008]	0.957%
Lavoué [2012]	0.925%
Xie et al. [2015]	0.988%
Ours	**0.988%**

Table 3.5: 3D object retrieval results comparison with state-of-the-art on SHREC'17.

Method	mAP
Furuya and Ohbuchi [2016] (BMVC'16)	0.476
Esteves et al. [2018b] (ECCV'18)	0.444
Tatsuma and Aono [2009]	0.418
Bai et al. [2016] (CVPR'16)	0.406
Ours	**0.452**

To illustrate the trade-off between model complexity and performance, Fig. 3.11 and Fig. 3.12 plot accuracy against number of trainable layers of state-of-the-art models on ModelNet10 and ModelNet40 datasets. These figures clearly show that volumetric convolution is in a better position compared to most recent models, in terms of the trade-off between complexity and accuracy. Note that, our method is not directly comparable with with some other recent works (e.g., Kanezaki et al. [2016]; Sedaghat et al. [2016]; Wu et al. [2016]; Qi et al. [2016]; Bai et al. [2016]; Maturana and Scherer [2015]) that use multiple-models and/or data and feature augmentation.

3.6.3 3D Object Retrieval

We evaluate the 3D object retrieval performance of our model on McGill and SHREC'17 datasets. We obtain the 200-dimensional feature descriptor after the frequency pooling layer, and measure the cosine similarity between the query shape and the shapes in the database. We first train the model as a classifier using train set, with softmax cross entropy as the loss function, and then use test set to evaluate the retrieval performance. The results for McGill dataset are shown in Table 4.4. We use the nearest neighbour performance measure for this task. For McGill dataset, we compare the performance of our model with six state-of-the-art techniques:Tabia et al. [2014], Agathos et al. [2009], Tabia et al. [2013], Papadakis et al. [2008], Lavoué [2012] and Xie et al. [2015]. As shown in Table 4.4, our feature vector is able to match the state of the art results achieved by Xie et al. [2015].

Table 4.5 depicts the performance comparison on SHREC'17 dataset (as reported in Esteves et al. [2018b]). This dataset includes random $\mathbb{SO}(3)$ perturbations. We use mean average precision (mAP) to evaluate the performance as done in other state-of-the-art techniques. Our model achieves the second best performance with a mAP value of 0.452, which is only a small drop (0.024) compared to Furuya and Ohbuchi [2016]. Overall, the results for 3D object retrieval task clearly demonstrate the richness of our proposed feature descriptors.

3.6.4 Ablation Study

To justify our model choices, we perform an extensive ablation study on ModelNet10 and SHREC'17 datasets. The results are reported in Table 3.6. First, we use two and three convolution layers instead of one. Accuracy drops from 93.8% to 92.0% and 89.8% in the cases of two layers and three layers respectively. This is an interesting result, as usually one might expect the model to compute richer features as the number of layers increase. However, it is known that in deep models, the accuracy does not always increase with the number of layer due to factors such as over-fitting on training set. Since our main focus of this work is to provide the theoretical framework of volumetric convolution and implement it as a differentiable layer that can be integrated into any deep architecture, we do not extensively investigate other architectural choices or regularization measures that might perform well with multiple layers. Rather, our main focus of the experiments is to show the richness of features computed using volumetric convolution.

Then, we replace the volumetric convolution layer with a spherical convolution layer, and achieve an accuracy of 76.2%. This is perhaps one of the most important comparisons in our ablation study. This clearly shows that modeling a 3D object and convolving it in $\mathbb{B}^3$ gives superior results as opposed to spherical convolution, which performs convolution in $\mathbb{S}^2$. To demonstrate one practical use-case of axial symmetry measurement of functions in $\mathbb{B}^3$, we measure the symmetry of objects around four equi-angular axes, and concatenate these measurement values to form a feature vector. Then we feed the generated feature vector to a fully connected layer to classify objects. Since we theoretically derive and implement the axial symmetry measurement formula as a fully differentiable module, this setting can be trained via backpropagation. We were able to achieve a 60.4% accuracy over ModelNet10 from using this simple hand-crafted feature.

Next, we investigate the effect of using various pooling mechanisms, instead of the proposed adaptive frequency pooling. In all the pooling operations, we create a (100×100) dimensional dense frequency map and perform pooling both row-wise and column-wise. First, we pool the outputs of the 16 kernels using mean-pooling and max-pooling, which drops the accuracy below 90% in both cases. Then we concatenate the kernel outputs and directly feed it to the fully connected layer, and achieve an accuracy of 87.8%. Ilse et al. [2018] recently proposed two novel attention based multiple instance learning (MIL) pooling mechanisms, that has trainable weights. Since our intuition for using frequency pooling is to capture prominent frequency

bands from frequency maps, we test the model using aforementioned MIL pooling mechanisms as it can learn to give attention to different frequency bands. We first construct the dense frequency map and then apply the two MIL pooling mechanisms—gated and non-gated—to achieve 89.8% and 90.3% accuracy respectively.

Recently, Liu et al. [2017] introduced a novel learning framework that gives angular representations on hyperspheres. This framework is supervised by two novel loss formulations that utilizes the angular similarity between the final descriptors. Since it is fair to assume that angular similarity plays a significant role in our model too—specially due to volumetric convolution's equivariance to 3D rotation group—we test the performance of weighted-softmax function and generalized-angular-softmax function proposed by Liu et al. [2017] in our experiments. However, as illustrated in Table 3.6, neither of these loss functions are able to outperform softmax loss function in our setting.

Furthermore, we use different similarity measures for retrieving 3D objects and compare the performances. As shown in Table 3.6, cosine similarity performs best while Euclidean, KL and Bhattacharya give inferior results.

3.6.5 Classification of highly non-polar and textured objects

The ablation study shown in Table 4.6 depicts that accuracy drops when multi-layer architectures are used in object classification. In this section, we explore a possible reason for this behaviour.

Two key features of our convolution layer are: (a) the ability to jointly model both shape and texture information, and (b) handling non-polar (i.e. dense in $\mathbb{B}^3$) objects. However, the dataset used for the ablation study experiment—ModelNet10—contains relatively simpler shapes with uniform texture. Therefore, using more layers (thus more parameters) can cause overfitting on the training set, as our network is able to capture highly discriminative features using a single volumetric convolution layer, which can cause a drop in test accuracy. To verify this, we employ a multi-layer architecture to classify a more challenging dataset, where objects are highly non-polar and textured.

To this end, we sample 1000 3D brain scan images from the large-scale OASIS-3 dataset (Fotenos et al. 2005). OASIS-3 is a compilation of 3D MRI brain scans obtained from over 1000 participants, collected over the course of 30 years. Participants include 609 cognitively normal adults and 489 individuals at various stages of Alzheimer's Disease aged between 42-95yrs. We split the sampled data in to train and test sets, comprising of 800 and 200 scans respectively. In both the train and test sets, we include equal numbers of Alzheimer and normal cases to avoid any bias. Afterwards, we train and test our network on the sampled data with softmax cross entropy as the loss function. Table 4.6 shows the performance against the number of layers used in the network.

In this experiment, we used 16 convolution kernels in each layer. As evident from Table 4.6, increasing the number of convolution layers improve the classification accuracy, up to three layers. Thus, it can be concluded that dense objects with texture

Table 3.6: Ablation study of the proposed architecture on ModelNet10 and SHREC'17 datasets. Here, "+" sign refers to "with" and "−" sign refers to "without".

3D Object Classification	
Method	Accuracy
Final Architecture (FA)	93.8%
FA (2Conv)	92.0%
FA (3Conv)	89.8%
FA − VolCNN + SphCNN	76.2%
Axial symmetry features	60.4%
FA − Adapt. FreqPool + MeanPool	84.2%
FA − Adapt. FreqPool + MaxPool	86.7%
FA − Adapt. FreqPool + FeatureConcat	87.8%
FA − Adapt. FreqPool + MILAPooling [Ilse et al., 2018]	90.3%
FA − Adapt. FreqPool + MILGAPooling [Ilse et al., 2018]	89.8%
FA + WSoftmax [Liu et al., 2017]	92.8%
FA + GASoftmax [Liu et al., 2017]	90.7%
3D Object Retrieval	
Method	mAP
FA (Cosine Similarity)	0.452
FA (Euclidean Distance)	0.386
FA (KL Divergence)	0.320
FA (Bhattacharyya Distance)	0.354

Table 3.7: Performance of multi-layer architectures for highly non-polar and textured shape classification. Our model shows an improvement with higher number of layers.

Model	**Accuracy**
Ours (1 Conv layer)	69.4%
Ours (2 Conv layers)	78.8%
Ours (3 Conv layers)	**83.2%**
Ours (4 Conv layers)	82.8%

Table 3.8: Performance comparison on local object-part movement resulting in global non-rigid deformations. Accuracies are reported for the ModelNet10 dataset. Performance drop under global deformations is shown in blue. Our approach demonstrates minimal performance drop under totally random deformations which signifies the strong invariance behaviour of proposed approach.

	Original Shape	Rot + Radial trans	Random
Ours	93.8%	91.4% (↓2.4)	88.5% (↓5.3)
Ours without FreqPool	82.8%	78.3% (↓4.5)	61.4% (↓21.4)
VoxNet	90.4%	43.8% (↓46.6)	42.1% (↓48.3)

allow our network to showcase its full capacity.

3.6.6 Equivariance to local pattern movements

For the translation and rotation of local feature patterns, which results in non-rigid deformations of the global shape, our proposed convolution operator ensures equivariance. In this experiment, we evaluate the robustness of our proposed network to such movements of local feature patterns. To this end, we radially translate and rotate local feature patterns in 3D and compare the behaviour of our approach with a traditional spatial-domain convolution method (Maturana and Scherer [2015]). Furthermore, we also test our approach with an even more difficult set of 'random' movements of local patterns.

Initially, we get the heat kernel signature of the 3D shape with 20 eigen vectors and a time stamp of 10. Afterwards, we get the vertices associated with the highest 10% of the heat response and cluster them using DBSCAN algorithm Ester et al. [1996]. We then find the centroid of each cluster, and get the 50 closest sample points to each centroid to obtain a set of sample point clusters. We then move each cluster independently and classify the final set of points using networks already trained on ModelNet10. The results are shown in Table 3.8.

As illustrated in Table 3.8, our network is robust to both random and rotational + translational movements of local patterns. Furthermore, when we remove the frequency pooling after the convolution layer, and connect the fully connected layer directly to the response of the convolution, the network becomes less robust to random movements of local patterns. Overall, even with highly challenging severe deformations, we note that the proposed approach does not experience a significant drop in the accuracy, compared to the spatial-domain convolution based approach. This behaviour signifies the strong invariance capability of proposed convolution operator.

3.6.7 Robustness against information loss

One critical requirement of a 3D object classification model is to be robust against information loss. To demonstrate the effectiveness of our proposed features in this

aspect, we randomly remove data points from the objects in validation set, and evaluate model performance. The results are illustrated in Fig. 3.13. The model shows no performance loss until 20% of the data points are lost, and only gradually drops to an accuracy level of 66.8 at a 50% data loss. This implies that the proposed model is robust against data loss and can work well for incomplete shapes.

3.6.8 Approximation Accuracy of 3D Zernike moments calculation approach

In Sec. 3.3.3, we proposed an alternative method to calculate 3D Zernike moments (Eq. 3.22, 3.23), instead of the conventional approach (Eq. 3.16). We hypothesized that moments obtained using the former has a closer resemblance to the original shape, due to the impact of finite number of frequency terms. In this section, we demonstrate the validity of our hypothesis through experiments. To this end, we compute moments for the shapes in the validation set of ModelNet10 dataset using both approaches, and compare the mean reconstruction error defined as: $\frac{1}{T}\sum_t^T \left\| f(t) - \sum_n \sum_l \sum_m \Omega_{n,l,m} Z_{n,l,m}(t) \right\|$, where T is the total number of points and $t \in \mathbb{B}^3$. Fig. 3.14 shows the results. In both approaches, the mean reconstruction error decreases as n increases. However, our approach shows a significantly low mean reconstruction error of 0.0467% at $n = 6$ compared to the conventional approach, which has a mean reconstruction error of 0.56% at same n. This result also justifies the utility of Zernike moments for modeling complex 3D shapes.

3.7 Comparison with Invariant Approaches

In the literature, the 3D shape analysis techniques with invariance properties have been proposed for both continuous surfaces and discrete point clouds. For the former representations, a Riemannian metric was proposed for parameterized 3D surfaces that is invariant to shape re-parametrizations [Kurtek et al., 2010]. In our case, invariance to re-parametrization group is less practical as we are working with discretized 3D shapes. Further, incorporating such distance metrics and parametrizations of real-world 3D shapes within deep feature learning models is a fairly challenging and largely unsolved problem. For the case of point clouds, permutation invariance has been studied for deep convolutional [Qi et al., 2017a] and graph convolutional networks in [Maron et al., 2018; Wang et al., 2018a]. The above works show that achieving permutation invariance is relatively simple in deep architectures.

Graph based approaches have been proposed to work on non-Euclidean topologies and are thus suitable to operate on 3D surfaces [Bronstein et al., 2017]. An input surface is converted to a graphical representation (e.g., polygon mesh) and converted to spectral domain where convolution is performed [Bruna et al., 2013; Defferrard et al., 2016; Boscaini et al., 2015; Henaff et al., 2015]. A different set of methods first reduce the complexity of input data by projecting them in a parametric 2D representation space and then apply convolutions to learn features. Finally, [Masci et al., 2015; Monti et al., 2017; Boscaini et al., 2016] perform convolution within local surface patches and thus provide invariance to surface deformations. However, the

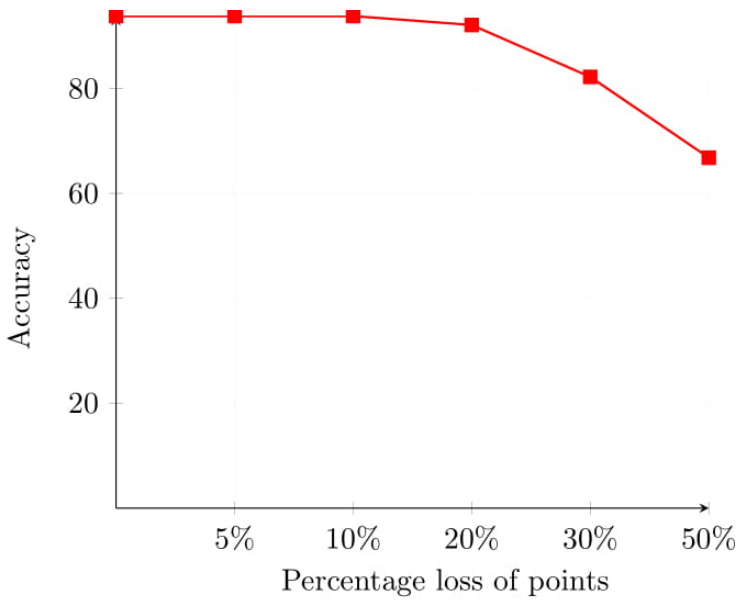


Figure 3.13: The robustness of the proposed model against missing data. The accuracy drop is less than 30% at a high data loss rate of 50%.

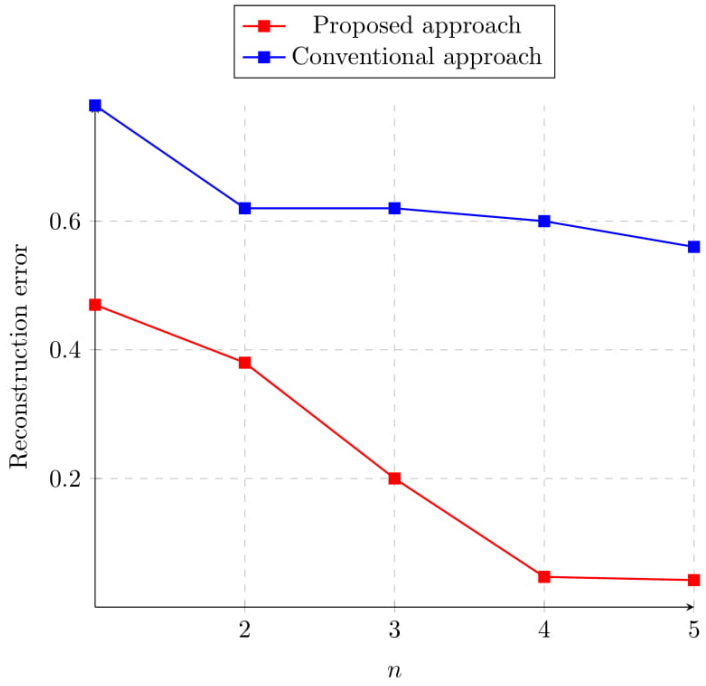


Figure 3.14: The mean reconstruction error Vs 'n'. Our Zernike frequencies computation approach has far less error than the conventional approach.

desirable invariance to deformations is generally dictated by the end-task and may not always be desirable since significant deformations can change object functionality, affordance and semantics [Su et al., 2018]. As we explain in the next paragraph, our fully learnable network allows task-dependent learning of invariance to shape deformations. Further, all of the above approaches learn representations on the shape surface or its 2D transformed version and do not consider the volumetric nature of 3D shapes.

In comparison to above mentioned approaches, we propose a novel convolutional operator in $\mathbb{B}^3$ that is suitable for volumetric shapes. Our proposed convolution operator is equivariant to isometric transformations (rotation, translation), and is also robust to non-isometric variations such as deformations (radial translations of local object parts) and shape articulations (scaling and local part rotations) (as shown in Sec. 3.6.6). Deep learning based solutions that can achieve invariance to all types of deformations are relatively less explored[1] and, to the best of our knowledge, ours is the first roto-translation equivariant convolution operator inside the unit-ball. The advantage of our approach over graph based invariant models is the ability to learn representations on volumetric shapes. As an example, a recent work [Maron et al., 2018] investigates invariance and equivariance for graph networks but only considering the linear layers (not the convolution ones) in the 2D case. The extension of our proposed convolution operator to arbitrary graphs and all possible deformations is an interesting research problem but beyond the scope of current work.

It is noteworthy that the end-to-end representation learning in our case automatically enforces invariance to deformations and articulations depending on the end-task. In comparison, the traditional approaches [Carrière et al., 2015; Reininghaus et al., 2015] for topological data analysis propose hand-crafted descriptors (based on persistence diagrams) that are invariant to only certain classes of deformations (intrinsic and extrinsic isometries). As a result, these descriptors are relatively less generalizable and their manual design offers less flexibility for new problems. We have conducted an experiment in this regard, where ModelNet10 shapes are first deformed by random movements of local object parts and their feature representations are used for final classification (Sec. 3.6.6). We achieve a classification accuracy quite close to that of original shapes, showing that the deformed and articulated shapes are mapped close to the original unaltered 3D shapes in the learned feature space.

3.8 Chapter summary

Equivariant representation learning is an important form of inductive bias that can help the deep networks to discover underlying symmetries of data. In this chapter, we derive a novel '*volumetric convolution*' operation using 3D Zernike polynomials, that can learn feature representations in $\mathbb{B}^3$ while preserving equivariance over the $\mathbb{SO}3$ group. We develop the underlying theoretical foundations for volumetric convolution and demonstrate that it can be efficiently computed and implemented using low-cost

[1] We refer the reader to [Cohen et al., 2018a] for an excellent review on group equivariant CNNs.

matrix multiplications. Furthermore, we propose a novel, fully differentiable method to measure the axial symmetry of a function in $\mathbb{B}^3$ around an arbitrary axis, using 3D Zernike polynomials and demonstrate one possible use case by proposing a simple hand-crafted descriptor. Finally, using volumetric convolution as a building tool, we propose an experimental architecture, that gives competitive results over state-of-the-art with a relatively shallow network, in 3D object recognition and retrieval tasks. In this experimental architecture, we introduce a novel frequency pooling layer, which can learn frequency bands in which the most discriminative features lie. One drawback of the current volumetric convolution operator is that 3D Zernike polynomials loose their orthogonality when a 3D translation is applied. This prevents volumetric convolution from achieving automatic translation invariance. Therefore one immediate extension to this work would be to investigate novel orthogonal and complete basis polynomials in a unit ball, which preserves its orthogonality when translated. Such polynomials would make it possible to achieve translation invariance more efficiently—compared to the proposed method—as then, the conversion from the spatial domain to spectral domain at each translation of the kernel is not necessary.

Chapter 4

Blended Convolution and Synthesis for Efficient Discrimination of 3D shapes.

The human world is three-dimensional, therefore optimally understanding and interpreting 3D data is an important research problem. In the last chapter, we investigated a novel convolution-based feature extraction method for 3D data that is equivariant to 3D translation and rotation. In this chapter, we will explore how to further inject inductive bias to CNNs in order to address several issues that are entailed with 3D data. Specifically, there are two main issues pertinent to 3D data: *(a)* 3D point clouds and rasterized voxel based representations encode redundant information thereby making inter-class discrimination difficult, *(b)* 3D convolutions generally operate in Euclidean space, whereas real-world 3D data lie on a non-Euclidean manifold. The representations thus learned fail to encode the true geometric structure of input shapes. The availability of low-cost 3D sensors and their vast applications in autonomous cars, medical imaging and scene understanding demands a fresh look towards solving the above-mentioned challenges.

Existing representation learning schemes for 3D shape description either operate on voxels [Brock et al., 2016; Wu et al., 2016] or point clouds [Qi et al., 2017b; Klokov and Lempitsky, 2017; Cheraghian et al., 2020, 2019b; Cheraghian and Petersson, 2019; Cheraghian et al., 2019a, 2020]. The voxelized data representations are highly sparse, thus prohibiting the design of large-scale deep CNNs. Efficient data structures such as Octree [Meagher, 1982] and Kdtree [Bentley, 1975] have been proposed to solve this problem, however neural networks based representation learning on these tree-based indexing structures is an open research problem [Riegler et al., 2017]. In comparison, point clouds offer an elegant, simple and compact representation for each point (x, y, z). Additionally, they can be directly acquired from the 3D sensors, e.g., low-cost structured light cameras. On the down side, their irregular structure and high point redundancy pose a serious challenge for feature learning.

We note that recent attempts on direct feature learning from point clouds assume a simplistic pipeline (see Fig. 4.1) that mainly aims to extract better global features considering all points [Qi et al., 2017a,b; Klokov and Lempitsky, 2017; Li et al.,

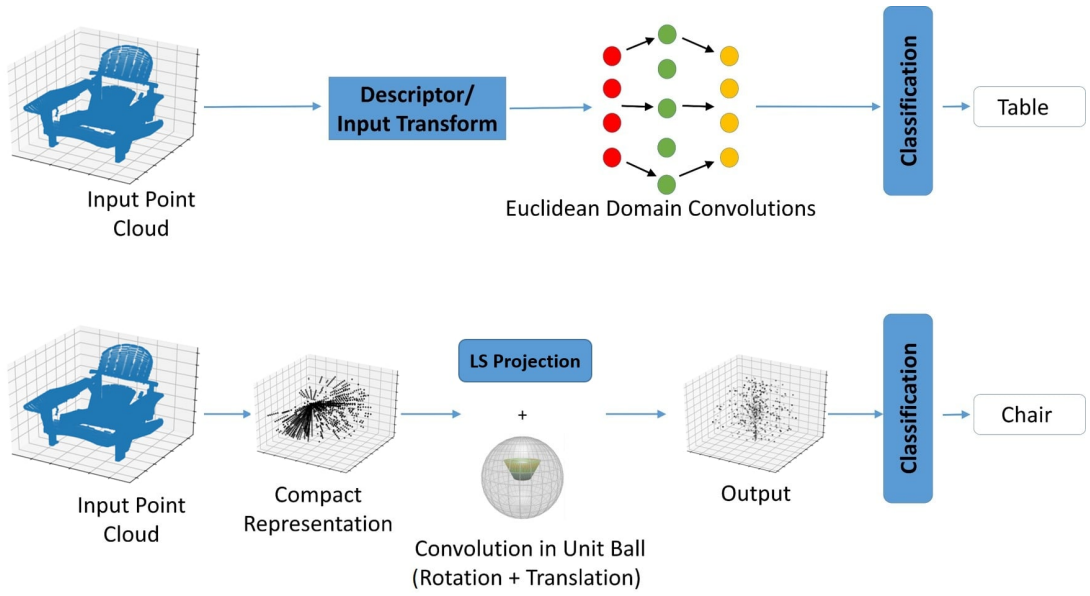


Figure 4.1: High-level comparison of our approach (*bottom)* with the traditional approaches [Qi et al., 2017a; Su et al., 2015; Qi et al., 2017b; Klokov and Lempitsky, 2017; Li et al., 2018b] (*top*). We transform an input shape into a compact representation and project it onto a discriminative latent space to capture more discriminative features, before performing convolution in $\mathbb{B}^3$ with roto-translational kernels. Our novel convolution operator has a clear advantage over existing works that only work with Euclidean geometries. This results in a light-weight and highly efficient network design with significantly lower number of layers.

2018b]. However, all these approaches lack the capacity to work on non-Euclidean geometries and have no inherent mechanism to deal with the high redundancy of point clouds. In this work, we propose an integrated solution, called Blended Convolution and Synthesis (BCS), to address the above-mentioned problems. BCS can effectively deal with the irregular, permutation-invariant and redundant structure of point clouds. Our solution has two key aspects. *First,* we map the input 3D shape into a more discriminative 3D space. We posit that raw 3D point clouds are sub-optimal to be directly used as input to classification models, due to redundant information. This property hampers the classification and retrieval performance by adding an extra overhead to the network, as the network should then disregard redundant features purely using convolution. In contrast, we initially synthesize a more discriminative shape by projecting the original shape to a latent space using a newly derived set of functions which are complete in the unit ball ($\mathbb{B}^3$). The structure of this latent shape is governed by the loss function, and therefore, is optimized to pick up the most discriminative features. This step reduces the number of convolution layers significantly, as shown experimentally in Sec. 6.6. *Second,* we propose a new convolution operation that works on non-Euclidean typologies i.e., inside the unit ball ($\mathbb{B}^3$). We derive a novel set of complete functions within $\mathbb{B}^3$ that perform convolution

in the spectral domain.

Furthermore, since our network operates on the '*spectral domain*', it provides multiple advantages compared to competing models that operate in Euclidean domains: 1) A highly compact and structured representation of 3D objects, which addresses the problem of redundancy and irregularity. Effectively, a 3D shape is represented as a linear combination of complete-orthogonal functions, which allows only a few coefficients to encode shape information, compared to spatial domain representations. 2) Convolution is effectively reduced to a multiplication-like operator which improves computational efficiency, thereby significantly reducing the number of FLOPS. 3) A theoretically sound way to treat non-Euclidean geometries, which enables the convolution to achieve translational and rotational *equivariance*; and 4) Scalability to large-sized shapes with bounded complexity.

Most importantly, existing methods which perform convolution in the spectral domain [Cohen et al., 2018b; Esteves et al., 2018b; Ramasinghe et al., 2019a] use spherical harmonics or Zernike polynomials to project 3D functions to the spectral domain for performing convolution. The aforementioned function spaces entail certain limitations, e.g.: 1) 'Spherical harmonics' only operate on the surface of the unit sphere, which causes critical information loss for non-polar shapes. 2) 'Zernike polynomials' cause the convolution to achieve only 3D rotational movement of the kernel. In contrast, our newly derived polynomials can handle non-polar shapes, while achieving both 3D rotational and translational movements of the convolution kernel as theoretically proved in Sec. 4.3.2. Recently, Jiang et al. [2019] proposed a novel Fourier transform mechanism to optimally sample non-uniform data signals defined on different topologies to spectral domain without spatial sampling error. This allows CNNs to analyze signals on mixed topologies, regardless of the architecture. However, their spectral transformation does not specifically focus on computational efficiency and equivariance properties, as ours. On the other hand, Huang et al. [2019a] proposed a model which can directly segment textured 3D meshes, by extracting features from high-resolution signals on geodesic neighborhoods of surfaces. In contrast, our model consumes point clouds and we propose a lightweight convolution operator, which extracts useful features for 3D classification.

The main contributions of this work are:

- A novel approach to obtain a learned 3D shape descriptor, which enhances the convolutional feature extraction process, by projecting the input 3D shape into a latent space, using newly derived functions in $\mathbb{B}^3$.
- Develop the theory of a novel convolution operation, which allows both 3D rotational and 3D translational movements of the kernel.
- Derive formulae to perform discriminative latent space projection of the input shape and 3D convolution in a single step, thereby making our approach computationally efficient.
- Implement the proposed latent space projection and convolution as a fully differentiable module which can be integrated into any end-to-end learning

architecture, and developing a shallow experimental network which produces results on par with state-of-the-art while being computationally efficient.

4.1 Related Work

3D shape descriptors: A 3D shape descriptor is a representation of the structural essence of a 3D shape. A variety of hand-crafted feature descriptors have been proposed in past research efforts. A few key such works are based on light field descriptors [Chen et al., 2003], Fourier transformation [Vranic et al., 2001], eigen value descriptors [Jain and Zhang, 2007], and geometric moments [Elad et al., 2002]. Most recent hand-crafted 3D descriptors are based on diffusion parameters [Bronstein et al., 2010; Rustamov, 2007; Bronstein et al., 2009]. On the other hand, learned 3D shape descriptors have also been popular in the computer vision literature. Litman et al. [2014] propose a supervised bag-of-features (BOF) method to learn a descriptor. Zhu follow an interesting approach, where they first project the 3D shapes into multiple 2D shapes, and then perform training on the 2D shapes to learn a descriptor. Xie et al. [2016] present a hybrid approach which combines both hand-crafted features and deep networks. They first compute a geometric feature vector from the 3D shape, and then employ a deep network on the feature vector to learn a 3D descriptor. Xie et al. [2015] follow a similar approach, where they first calculate heat kernel signatures of 3D shapes and then use two deep encoders to obtain descriptors. Our work is partially similar to this, but has a key difference: instead of computing hand-crafted features as the first step, we do a learned mapping of input 3D shape into a more discriminative 3D space, which allows us to get rid of high intra-class variances exhibited by most 3D shape descriptors. This step provides another advantage since it maximizes the distance between initial shapes, before being fed to convolution layers later.

Orthogonal Moments and 3D Convolution: Generally, orthogonal moments are used to obtain deformation invariant descriptors from structured data. Compared to geometric moments, orthogonal moments are robust to certain deformations such as rotation, translation and scaling. This property of orthogonal moments has been exploited specially in 2D data analysis in the past [Hu, 1962; Lin and Chellappa, 1987; Arbter et al., 1990; Tieng and Boles, 1995; Khalil and Bayoumi, 2001; Suk and Flusser, 1996]. Extension of deformation invariant moments from 2D to 3D also has been explored by many prior works [Guo, 1993; Reiss, 1992; Canterakis, 1999; Flusser et al., 2003]. However, the certain properties of these moments depend on the Hilbert space on which they are defined. For example, orthogonal moments defined in a sphere or a ball exhibit convenient properties to extract rotation invariants, compared to orthogonal moments defined in a cube. These unique properties of orthogonal moments have recently been used to derive convolution operations which allows 3D rotational movements of kernels [Cohen et al., 2018b; Esteves et al., 2018b; Ramasinghe et al., 2019a,c]. However, the moments used in these works do not contain the necessary properties to achieve 3D translation of the kernels, and therefore, we derive a novel set of functions in $\mathbb{B}^3$ to overcome this limitation.

3D Shape Classification and Retrieval: Recent works developed for 3D shape classification and retrieval can be broadly categorized into three classes: 1) hand-crafted feature based [Vranic and Saupe, 2002], [Guo et al., 2016] 2) unsupervised learning based [Wu et al., 2016], [Khan et al., 2018] 3) deep learning based [Qi et al., 2017a,b; Li et al., 2016]. Generally, deep learning based approaches have shown superior results compared to other two categories. However, the aforementioned deep learning architectures operate on Euclidean spaces, which is sub-optimal for 3D shape analysis tasks, although Weiler et al. [2018b] has shown impressive results using SE(3)-equivariant convolutions in the Euclidean domain. In contrast, our network performs convolution on $\mathbb{B}^3$ which allows efficient feature extraction, since 3D rotation and translation of kernels are easier to achieve in this space.

4.2 Preliminaries

We first provide an overview of basic concepts that will be used later in proposed method.

4.2.1 Complete Orthogonal Systems

Orthogonal functions are useful tools in shape analysis. Let Φ_m and Φ_n be two functions defined in some space $\mathbb{S}$. Then, Φ_m and Φ_n are orthogonal over the space $\mathbb{S}$ if and only if,

$$\int_{\mathbb{S}} \Phi_n(X)\Phi_m(X)dX = 0, \ \forall n \neq m. \tag{4.1}$$

Let f be a function defined in space $\mathbb{S}$, and $\{\Phi_m : m \in \mathbb{Z}^+\}$ be a set of orthogonal functions defined in the same space. Then, the set of orthogonal moments of f, with respect to set $\{\Phi_m\}$, can be obtained by $\hat{f}_m = \int_{\mathbb{S}} f(X)\Phi_m(X)^\dagger$ where $\dagger$ denotes the complex conjugate. If a set of functions $\{\Phi_m : m \in \mathbb{Z}^+\}$ is both complete and orthogonal, it can reconstruct $f(X)$ using its orthogonal moments as follows,

$$f(X) = \sum_m \hat{f}_m \Phi_m(X). \tag{4.2}$$

4.2.2 Convolution in Unit Ball $\mathbb{B}^3$

The unit ball ($\mathbb{B}^3$) is the set of points $x \in \mathbb{R}^3$, where $\|x\|<1$. Any point in $\mathbb{B}^3$ can be parameterized using coordinates (θ, ϕ, r), where θ, ϕ, and r are azimuth angle, polar angle, and radius respectively. Performing convolution on 3D shapes in non-linear topological spaces such as the unit ball ($\mathbb{B}^3$) has a key advantage: compared to the Cartesian coordinate system, it is efficient to formulate 3D rotational movements of the convolutional kernel in $\mathbb{B}^3$ [Ramasinghe et al., 2019a]. To this end, both the input 3D shape and the 3D kernel should be represented as functions in $\mathbb{B}^3$. However, performing convolution in the spatial domain is difficult due to non-linearity of $\mathbb{B}^3$ space [Ramasinghe et al., 2019a]. Therefore, it is necessary to first obtain the spectral

representation of the 3D shape and the 3D kernel, with respect to a set of orthogonal and complete functions in $\mathbb{B}^3$, and consequently perform spectral domain convolution.

4.3 Methodology

Here, we present our 'Blended Convolution and Synthesis' layer in detail. First, we construct a set of orthogonal and complete polynomials in $\mathbb{B}^3$. Then, we relax the orthogonality condition of these polynomials, which allows us to project the input shape to a latent space. This projection is a learned process and depends on the softmax cross-entropy between predicted and ground-truth object classes. Therefore, the projected shape is optimized to contain more discriminative properties across object classes. Afterwards, we convolve the latent space shape with roto-translational kernels in $\mathbb{B}^3$ to map it to the corresponding class. Besides, we derive formulae to achieve both projection and convolution in a single step, which makes our approach more efficient.

Below in Section 4.3.1, we explain the learned projection of the object onto a latent space. Then, in Section 4.3.2, we derive our convolution operation, which is able to capture features efficiently using roto-translational kernels.

4.3.1 Learned Mapping for Shape Synthesis

In this section, we explain the projection of 3D point clouds to a discriminative latent space in $\mathbb{B}^3$. First, we derive a set of complete orthogonal functions in $\mathbb{B}^3$. Orthogonal moments obtained using orthogonal functions can be used to reconstruct the original object. However, our requirement here is not to reconstruct the original object, but to map it to a more discriminative shape. Therefore, after deriving the orthogonal functions, we relax the orthogonality condition to facilitate the latent space projection. Furthermore, instead of the input point cloud, we use a compact representation as the input to the feature extraction layer, for efficiency and to leverage the capacity of convolution in $\mathbb{B}^3$. In Section 6.3.3, we explain our compact representation.

4.3.1.1 Compact Representation of Point Clouds

Most 3D object datasets contain point clouds with uniform texture. That is, if the 3D shape is formulated as a function f in $\mathbb{B}^3$, such that for any point on the shape, $f(\theta, \phi, r) = c$, where c is a constant. However, formulating 3D shapes in $\mathbb{B}^3$ has the added advantage of representing both 2D texture and 3D shape information simultaneously [Ramasinghe et al., 2019a]. Therefore, the advantage of convolution in $\mathbb{B}^3$ can be utilized when the input and kernel functions have texture information.

Following this motivation, we convert the uniform textured point clouds into non-uniform textured point clouds using the following approach. First, we create a grid using equal intervals along r, θ, and ϕ. We use $25, 36$, and 18 interval spaces for r, θ, and ϕ, respectively. Then, we bin the point cloud to grid points, which results in a less dense, non-uniform surface valued point cloud. The obtained compact

representation does not contain all the fine-details of the input point cloud. However in practice, it allows better feature extraction using the kernels. A possible reason could be that kernels are also non-uniform textured point clouds with discontinuous space representations, and they can capture better features from non-uniform textured input point clouds when performing convolution in $\mathbb{B}^3$.

4.3.1.2 Derivation of orthogonal functions in $\mathbb{B}^3$

In this section, we derive a novel set of orthogonal polynomials with necessary properties to achieve the translation and rotation of convolution kernels. Afterwards, in Section 4.3.1.4, we relax the orthogonality condition of the polynomials to facilitate latent space projection.

Canterakis [1999] showed that a set of orthogonal functions which are complete in unit ball can take the form $Z_{n,l,m}(r,\theta,\phi) = Q_{nl}(r)Y_{l,m}(\theta,\phi)$, where Q_{nl} is the linear component and $Y_{l,m}(\theta,\phi)$ is the angular component. The variables r, θ and ϕ are radius, azimuth angle and polar angle, respectively. We choose $Y_{l,m}(\theta,\phi)$ to be spherical harmonics, since they are complete and orthogonal in S^2.

For the linear component, we do not use the Zernike linear polynomials as in Canterakis [1999], as they do not contain the necessary properties to achieve the translational behaviour of convolution kernels [Ramasinghe et al., 2019a]. Therefore, we derive a novel set of orthogonal functions, which are complete in $0 < r < 1$, and can approximate any function in the same range. Furthermore, it is crucial that these functions contain necessary properties to achieve the translation of kernels while performing convolution. Therefore, we choose the following function as the base function:

$$f_{nl} = (-1)^l n \sum_{k=0}^{n} \frac{((n-l)r)^k}{k!}. \tag{4.3}$$

It can be seen that,

$$f_{nl} \approx (-1)^l n \exp(r(n-l)), \tag{4.4}$$

as n increases, for small r. Therefore, we use the approximation given in Eq. 4.4 in future derivations. As we show in Section 4.3.2, this property is vital for achieving the translation of kernels. Next, we orthogonalize $f_{nl}(r)$ to obtain a new set of functions $Q_{nl}(r)$. Consider the orthogonality

$$\int_{\mathbb{B}^3} Z_{n,l,m} Z_{n',l',m'} = 0, \forall n \neq n', l \neq l', m \neq m'. \tag{4.5}$$

If we consider only the linear component, the orthogonality condition should be

$$\int_0^1 Q_{n,l} Q_{n',l'} r^2 dr = 0, \forall n \neq n', l \neq l'. \tag{4.6}$$

Therefore, $Q_{n,l}$ should be orthogonal with respect to the weight function $w(r) = r^2$. We define,

$$Q_{nl}(r) = f_{nl}(r) - \sum_{k=0}^{n-1} \sum_{m=0}^{k} C_{nlkm} Q_{km}(r) \tag{4.7}$$

where $n \geq 0, n \geq l \geq 0$ and C_{nlkm} is a constant. Since Q_{nl} should be an orthogonal set, the inner product between any two different Q_{nl} functions is zero. Therefore, we obtain,

$$\langle Q_{nl}, Q_{n'l'} \rangle = \langle f_{nl}, Q_{n'l'} \rangle - \sum_{k=0}^{n-1} \sum_{m=0}^{k} C_{nlkm} \langle Q_{km}, Q_{n'l'} \rangle$$

Since $\langle Q_{nl}, Q_{n'l'} \rangle = 0$, we get:

$$C_{nln'l'} = \frac{\langle f_{nl}, Q_{n'l'} \rangle}{\| Q_{n'l'} \|^2}. \tag{4.8}$$

Following this process, we can obtain the set of orthogonal functions Q_{nl} for $n \geq 0, n \geq l$. The derived polynomials up to $n = 5, l = 5$ are shown in Table. 4.1. In Section 4.3.1.3, we prove the completeness property of the derived functions.

Table 4.1: The derived Q_{nl} polynomials up to $n = 5$, $m = 5$.

Polynomial	Expression
Q_{00}	0
Q_{10}	$1. + 2x$
Q_{11}	$-1. - 1x$
Q_{20}	$-9.79 - 10.65x + 9x^2$
Q_{21}	$5.29 + 6.29x - 4x^2$
Q_{22}	$-1.99 - 3.63x + x^2$
Q_{30}	$-123.58 - 158.11x + 87.46x^2 + 32x^3$
Q_{31}	$70.26 + 89.41x - 50.31x^2 - 13.5x^3$
Q_{32}	$15.86 + 22.27x - 11.06x^2 - 0.5x^3$
Q_{33}	$-768.81 - 1006.25x + 512.65x^2 + 139.10x^3 + 104.16x^4$
Q_{40}	$-35.86 - 46.15x + 25.59x^2 + 4x^3$
Q_{41}	$422.87 + 550.70x - 287.81x^2 - 73.52x^3 - 42.66x^4$
Q_{42}	$-768.81 - 1014.25x + 480.65x^2 + 73.77x^3 + 13.5x^4$
Q_{43}	$-776.81 - 1034.25x + 454.65x^2 + 50.43x^3 - 2.66x^4$
Q_{44}	$-768.81 - 1022.25x + 464.65x^2 + 56.43x^3 + 0.16x^4$
Q_{50}	$-3683.18 - 4855.97x + 2342.20x^2 + 509.59x^3 + 340.36x^4 + 324x^5$
Q_{51}	$1960.80 + 2578.79x - 1263.64x^2 - 280.02x^3 - 167.77x^4 - 130.20x^5$
Q_{52}	$-981.80 - 1286.88x + 643.53x^2 + 141.74x^3 + 72.23x^4 + 42.66x^5$
Q_{53}	$463.12 + 604.69x - 309.13x^2 - 64.52x^3 - 25.87x^4 - 10.12x^5$
Q_{54}	$-208.26 - 272.17x + 140.81x^2 + 25.87x^3 + 7.44x^4 + 1.33x^5$
Q_{55}	$91.29 + 122.33x - 61.70x^2 - 9.53x^3 - 2.07x^4 - 0.04x^5$

4.3.1.3 Completeness in $\mathbb{B}^3$

In this section, we prove the completeness in $\mathbb{B}^3$ for the set of functions $\{Q_{nl}\}$ derived in Section 4.3.1.2.

Condition 1: *Consider the orthogonal set* $\{p_n\}$ *defined in* $L^2[0,1]$. *Then,* $\{p_n\}$ *is complete in space* $L^2[0,1]$ *if and only if there is no non-zero element in* $L^2[0,1]$ *that is orthogonal to every* $\{p_n\}$.

To show that f_{nl} is complete over $L^2[0,1]$, we first prove the completeness of the set $\{\Phi_n\}$, which is obtained by orthogonalizing the set $\{1, x, x^2, x^3, ...\}$. Let $\Psi(x)$ be an element in $L^2[0,1]$, which is orthogonal to every element of $\{1, x, x^2, x^3, ...\}$. Then, suppose the following relationship is true:

$$\langle \Psi, e^{2\pi ikx} \rangle = \sum_{n=0}^{\infty} \frac{(2\pi ikn)^n}{n!} \langle \Psi, x^n \rangle = 0, \tag{4.9}$$

where k is a constant. However, we know that $\{e^{2\pi ikx}\}_{k=0}^{k=\infty}$ is the complex exponential Fourier basis, and is both complete and orthogonal. Therefore, if Eq. 4.9 is true, $\Psi = 0$, which gives us the result, i.e., $\langle \Psi, x^n \rangle = 0 \equiv \Psi = 0$. Equivalently, since $\{\Phi_n\}$ is obtained by orthogonalization of $\{1, x, x^2, x^3, ...\}$, $\langle \Psi, \{\Phi_n\} \rangle = 0 \equiv \Psi = 0$. Hence, according to Condition 1, $\{\Phi_n\}$ is complete in $L^2[0,1]$.

Next, we consider the set $Q_{n,l}$. Since $Q_{n,l}$ is orthogonalized using the basis functions in Eq. 4.21, it is enough to show that f_{nl} is complete over $L^2[0,1]$. Let Θ be a function defined in $L^2[0,1]$. Then, suppose the following relationship is true:

$$\langle \Theta, f_{n,l} \rangle = (-1)^l n \sum_{k=0}^{n} \frac{((n-l)^k}{k!} \langle \Theta, r^k \rangle = 0. \tag{4.10}$$

For Eq. 4.10 to be true, $\langle \Theta, r^k \rangle = 0$ for $k = \{0, 1, 2, ...\}$. But we showed that this condition is satisfied if and only if $\Theta = 0$. Therefore, $\langle \Theta, f_{n,l} \rangle = 0, \forall n \geq l \geq 0 \equiv \Theta = 0$. Hence, $f_{n,l}$ is complete in $L^2[0,1]$.

4.3.1.4 Relaxation of orthogonality of functions in $\mathbb{B}^3$

Computing $C_{nln'l'}$ using Eq. 4.8 ensures the orthogonality of Q_{nl}. Since Q_{nl} and Y_{lm} are both orthogonal and complete, projecting the input shape f onto the set of functions $Z_{nlm}, n \geq l \geq m \geq 0$, enables us to reconstruct f by:

$$f(\theta, \phi, r) = \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \Omega_{n,l,m}(f) Z_{n,l,m}(\theta, \phi, r), \tag{4.11}$$

where spectral moment $\Omega_{n,l,m}(f)$ can be obtained using

$$\Omega_{n,l,m}(f) = \int_0^1 \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi, r) Z_{n,l,m} r^2 \sin\phi \, dr d\phi d\theta.$$

Representing f in spectral terms, as in Eq. 4.11, enables easier convolution in spectral space, as derived in Section 4.3.2.

However, we argue that since 3D point clouds across different object classes contain redundant information, projecting the point clouds in to a more discriminative latent space can improve classification accuracy. Our aim here is to reduce redundant

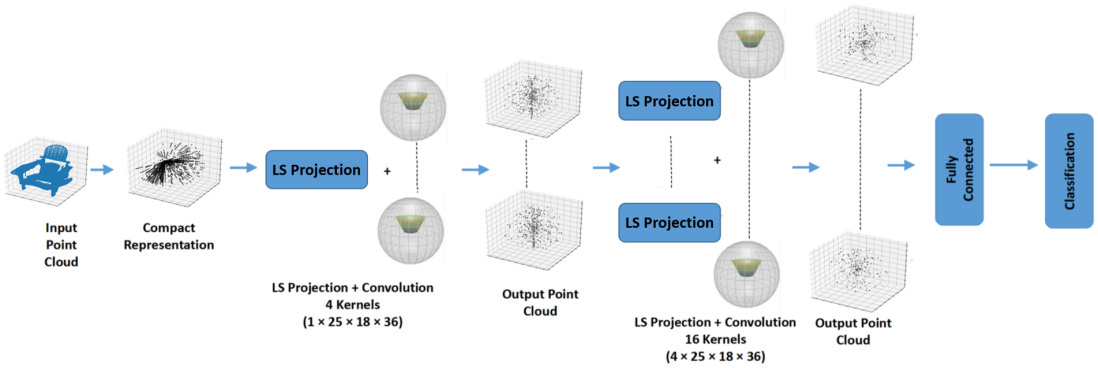


Figure 4.2: The overall CNN architecture. Our proposed design is a light-weight model, comprising of only three weight layers. Our networks aims to achieve a compact latent representation and volumetric feature learning via convolutions in $\mathbb{B}^3$.

information and noise from the input point clouds and map it to a more discriminative point cloud, which concentrates on discriminative geometric features. Therefore, we make $C_{nln'l'}$ trainable, which allows the latent space projection $\hat{f}$ of the input shape f as follows:

$$\hat{f}(\theta,\phi,r) = \sum_{n=0}^{\infty}\sum_{l=0}^{n}\sum_{m=-l}^{l} \hat{\Omega}_{n,l,m}(f)\hat{Z}_{n,l,m}(\theta,\phi,r), \tag{4.12}$$

where spectral moment $\hat{\Omega}_{n,l,m}(f)$ can be obtained using,

$$\hat{\Omega}_{n,l,m}(f) = \int_0^1\int_0^{2\pi}\int_0^{\pi} f(\theta,\phi,r)\hat{Z}^{\dagger}_{n,l,m} r^2 \sin\phi \, dr d\phi d\theta,$$

where

$$\hat{Z}_{n,l,m}(\theta,\phi,r) = \hat{Q}_{nl}(r)Y_{lm}(\theta,\phi) \tag{4.13}$$

and

$$\hat{Q}_{nl}(r) = f_{nl}(r) - \sum_{k=0}^{n-1}\sum_{m=0}^{k} W_{nlkm}\hat{Q}_{km}(r). \tag{4.14}$$

Here, the set $\{W_{nlkm}\}$ denotes trainable weights. Note that since the final orthogonal function is a product of the linear and the angular parts, making both functions learnable is redundant.

4.3.2 Convolution of functions in $\mathbb{B}^3$

Let the north pole be the y axis of the Cartesian coordinate system and the kernel is symmetric around y. Let $f(\theta,\phi,r)$, $g(\theta,\phi,r)$ be the functions of object and kernel

respectively. Then, convolution of functions in $\mathbb{B}^3$ is defined by:

$$\begin{aligned} & f * g(\alpha, \beta, r')\langle f(\theta, \phi, r), T'_r\{\tau_{(\alpha,\beta)}(g(\theta, \phi, r))\}\rangle \\ &= \int_0^1 \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi, r) T'_r\{\tau_{(\alpha,\beta)}(g(\theta, \phi, r))\} \sin\phi \, d\phi d\theta dr, \end{aligned} \tag{4.15}$$

where $\tau_{(\alpha,\beta)}$ is an arbitrary rotation that aligns the north pole with the axis towards the (α, β) direction (α and β are azimuth and polar angles respectively) and T'_r is translation by r'.

To achieve both latent space projection and convolution in $\mathbb{B}^3$ in single step, we present the following theorem.

Theorem 4.1. *Suppose $f, g : X \longrightarrow \mathbb{R}^3$ are square integrable functions defined in $\mathbb{B}^3$ so that $\langle f, f\rangle < \infty$ and $\langle g, g\rangle < \infty$. Further, suppose g is symmetric around the north pole and $\tau(\alpha, \beta) = R_y(\alpha)R_z(\beta)$ where $R \in \mathbb{SO}(3)$ and T'_r is translation of each point by r'. Then,*

$$\begin{aligned} & \int_0^1 \int_0^{2\pi} \int_0^{\pi} P\{f(\theta, \phi, r)\} T'_r\{\tau_{(\alpha,\beta)}(g(\theta, \phi, r))\} \sin\phi \, d\phi d\theta dr \\ & \approx \frac{4\pi}{3} \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \langle f_{nl}(r), Q_{n'l}(r)\rangle (e^{(n-l)r'} - e^{(n'-l)r'}) \\ & \quad \hat{\Omega}_{n,l,m}(f) \hat{\Omega}_{n,l,0}(g) Y_{l,m}(\theta, \phi), \end{aligned} \tag{4.16}$$

where, $\hat{\Omega}_{n,l,m}(f), \hat{\Omega}_{n,l,0}(g)$ and $Y_{l,m}(\theta, \phi)$ are $(n, l, m)^{th}$ spectral moment of f, $(n, l, 0)^{th}$ spectral moment of g, and spherical harmonics function, respectively. $P\{\cdot\}$ is the projection to a latent space, $\tau(\alpha, \beta) = R_y(\alpha)R_z(\beta)$ where $R \in \mathbb{SO}(3)$ and T_r is translation of each point by r.

Proof. The input function f is projected to the latent space shape $\hat{f}$ by,

$$\hat{f}(\theta, \phi, r) = \sum_{n=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \hat{\Omega}_{n,l,m}(f) \hat{Z}_{n,l,m}(\theta, \phi, r), \tag{4.17}$$

where spectral moment $\hat{\Omega}_{n,l,m}(f)$ can be obtained using,

$$\hat{\Omega}_{n,l,m}(f) = \int_0^1 \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi, r) \hat{Z}^{\dagger}_{n,l,m} r^2 \sin\phi \, dr d\phi d\theta. \tag{4.18}$$

and,

$$\hat{Z}_{n,l,m}(\theta, \phi, r) = \hat{Q}_{nl}(r) Y_{lm}(\theta, \phi), \tag{4.19}$$

where,

$$\hat{Q}_{nl}(r) = f_{nl}(r) - \sum_{k=0}^{n-1} \sum_{m=0}^{k} W_{nlkm} \hat{Q}_{km}(r), \tag{4.20}$$

$$f_{nl} = (-1)^l n \sum_{k=0}^{n} \frac{((n-l)r)^k}{k!}, \tag{4.21}$$

and,

$$Y_{l,m}(\theta,\phi) = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\phi) e^{im\theta}, \tag{4.22}$$

where $\phi \in [0,\pi]$ is the polar angle, $\theta \in [0,2\pi]$ is the azimuth angle, $l \in \mathbb{Z}^+$ is a non-negative integer, $m \in \mathbb{Z}$ is an integer, $|m| < l$, and $P_l^m(\cdot)$ is the associated Legendre function,

$$P_l^m(x) = (-1)^m \frac{(1-x^2)^{m/2}}{2^l l!}\frac{d^{l+m}}{dx^{l+m}}(x^2-1)^l. \tag{4.23}$$

In Eq. 4.20, the set $\{W_{nlkm}\}$ denotes trainable weights. Using this result, we can rewrite $f * g(r',\alpha,\beta)$ as,

$$\begin{aligned} f * g(r,\alpha,\beta) = \langle \sum_{n=0}^{\infty}\sum_{l=0}^{n}\sum_{m=-l}^{l} \hat{\Omega}_{n,l,m}(f)\hat{Z}_{n,l,m}(\theta,\phi,r), \\ T_{r'}\{\tau_{(\alpha,\beta)}(\sum_{n'=0}^{\infty}\sum_{l'=0}^{n}\sum_{m'=-l'}^{l'} \hat{\Omega}_{n',l',m'}(g)\hat{Z}_{n',l',m'}(\theta,\phi,r)))\}\rangle_{\mathbb{B}^3} \end{aligned} \tag{4.24}$$

Using the properties of inner product, this can be rewritten as,

$$\begin{aligned} f * g(r',\alpha,\beta) = \sum_{n=0}^{\infty}\sum_{l=0}^{n}\sum_{m=-l}^{l}\sum_{n'=0}^{\infty}\sum_{l'=0}^{'n}\sum_{m'=-l'}^{l'} \hat{\Omega}_{n,l,m}(f)\hat{\Omega}_{n',l',m'}(g) \\ \langle \hat{Z}_{n,l,m}(\theta,\phi,r), T_{r'}\{\tau_{(\alpha,\beta)}(\hat{Z}_{n',l',m'}(\theta,\phi,r))\}\rangle_{\mathbb{B}^3} \end{aligned} \tag{4.25}$$

Consider the term,

$$\begin{aligned} \Gamma &= \langle \hat{Z}_{n,l,m}(\theta,\phi,r), T_{r'}\{\tau_{(\alpha,\beta)}(\hat{Z}_{n',l',m'}(\theta,\phi,r)))\}\rangle_{\mathbb{B}^3} \\ &= \langle \hat{Q}_{nl}(r)Y_{lm}(\theta,\phi), T_{r'}\{\tau_{(\alpha,\beta)}(\hat{Q}_{n'l'}(r)Y_{l'm'}(\theta,\phi))\}\rangle_{\mathbb{B}^3} \end{aligned} \tag{4.26}$$

Γ can be decomposed into its angular and linear components as,

$$\Gamma = \int_0^1 \hat{Q}_{nl}(r)T_{r'}\{\hat{Q}_{n'l'}(r)\}r^2dr \int_0^{2\pi}\int_0^{\pi} Y_{lm}(\theta,\phi)\tau_{(\alpha,\beta)}(Y_{l'm'}(\theta,\phi))\}sin\phi d\phi d\theta. \tag{4.27}$$

First, consider the angular component,

$$Ang(\Gamma) = \int_0^{2\pi}\int_0^{\pi} Y_{lm}(\theta,\phi)\tau_{(\alpha,\beta)}(Y_{l'm'}(\theta,\phi))\}sin\phi d\phi d\theta. \tag{4.28}$$

Since $g(\theta,\phi,r)$ is symmetric around y, using the properties of spherical harmonics, Eq. 4.28 can be rewritten as,

$$Ang(\Gamma) = \int_0^{2\pi}\int_0^{\pi} Y_{lm}(\theta,\phi)\sum_{m''=-l'}^{l'} Y_{l',m''}D^{l'}_{m''0}(\alpha,\beta)\}sin\phi d\phi d\theta \tag{4.29}$$

where $D^{l'}_{mm'}$ is the Wigner-D matrix. But $D^{l'}_{m''0} = Y_{l',m''}$, and hence,

$$Ang(\Gamma) = \sum_{m''=-l'}^{l'} Y_{l',m''}(\alpha,\beta) \int_0^{2\pi} \int_0^{\pi} Y_{lm}(\theta,\phi) Y_{l',m''}(\theta,\phi)\} sin\phi d\phi d\theta \tag{4.30}$$

Since spherical harmonics are orthogonal,

$$Ang(\Gamma) = C_{ang} Y_{l,m}(\alpha,\beta), \tag{4.31}$$

where C_{ang} is a constant. Consider the linear component of Eq. 4.27. It is important to note that for simplicity, we derive equations for the orthogonal case and use the same results for non-orthogonal case. In practice, this step does not reduce accuracy.

$$Lin(\Gamma) = \int_0^1 \hat{Q}_{nl}(r) T_{r'}\{\hat{Q}_{n'l'}(r)\} r^2 dr. \tag{4.32}$$

Since $\hat{Q}_{nl}(r)$ is a linear combination of $f_{nl} \approx (-1)^l n \exp(r(n-l))$, it is straightforward to see that,

$$Q_{nl}(r+r') = f_{nl}(r)\exp((n-l)r') - \sum_{k=0}^{n-1}\sum_{m=0}^{k} C_{nlkm} Q_{km}(r)\exp(k-m)r'. \tag{4.33}$$

Also, we have derived that $l = l'$ from the result in Eq. 4.31. Applying this result and Eq. 4.33 to Eq. 4.32 gives,

$$\langle Q_{nl}(r+r'), Q_{n'l}(r)\rangle = \langle f_{nl}(r+r'), Q_{n'l}(r)\rangle - \sum_{k=0}^{n-1}\sum_{m=0}^{k} C_{nlkm}\langle Q_{km}(r+r'), Q_{n'l}(r)\rangle, \tag{4.34}$$

$$\langle Q_{nl}(r+r'), Q_{n'l}(r)\rangle = \langle f_{nl}(r), Q_{n'l}(r)\rangle e^{(n-l)r'} - \sum_{k=0}^{n-1}\sum_{m=0}^{k} C_{nlkm}\langle Q_{km}(r)e^{(k-m)r'}, Q_{(n'l}(r)\rangle. \tag{4.35}$$

Since Q_{km} and $Q_{n'l}$ are orthogonal,

$$\langle Q_{nl}(r+r'), Q_{n'l}(r)\rangle = \langle f_{nl}(r), Q_{n'l}(r)\rangle e^{(n-l)r'} - C_{nln'l} e^{(n'-l)r'} ||Q_{n'l}||^2. \tag{4.36}$$

But since for orthogonal case, $C_{nln'l'} = \frac{<f_{nl}, Q_{n'l'}>}{||Q_{n'l'}||^2}$,

$$\langle Q_{nl}(r+r'), Q_{n'l}(r)\rangle = \langle f_{nl}(r), Q_{n'l}(r)\rangle e^{(n-l)r'} - \langle f_{nl}(r), Q_{n'l'}(r)\rangle e^{(n'-l)r'}, \tag{4.37}$$

$$\langle Q_{nl}(r+r'), Q_{n'l}(r)\rangle = \langle f_{nl}(r), Q_{n'l}(r)\rangle (e^{(n-l)r'} - e^{(n'-l)r'}). \tag{4.38}$$

Combining Eq. 4.31 and Eq. 4.38 for Eq. 4.25 and choosing the normalization constant

to be $\frac{4\pi}{3}$ (since the integration is over unit ball) gives,

$$f * g(r', \alpha, \beta) \approx \frac{4\pi}{3} \sum_{n=0}^{\infty} \sum_{n'=0}^{\infty} \sum_{l=0}^{n} \sum_{m=-l}^{l} \langle f_{nl}(r), Q_{n'l}(r) \rangle \\ (e^{(n-l)r'} - e^{(n'-l)r'}) \hat{\Omega}_{n,l,m}(f) \hat{\Omega}_{n',l,0}(g) Y_{l,m}(\alpha, \beta). \tag{4.39}$$

□

4.3.3 Network Architecture

Our experimental architecture consists of two convolution layers and a fully connected layer. We employ four kernels in the first convolution layer and 16 kernels in the second convolution layer, each followed by group normalization [Wu and He, 2018] and a ReLU layer. The experimental architecture is illustrated in Figure 4.2. We use $n = 5$ for implementing Eq. 4.15 and softmax cross-entropy loss as the objective function during training. For training, we use a two step process. First, we train polynomial weights using a learning rate of 10^{-5}, and then train kernel weights using a learning rate of 0.01. We used the Adam optimizer for calculating gradients with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$, where parameters refer to the usual notation. We use $20k$ iterations to train polynomials weights and $30k$ iterations to train kernel weights. We use a single GTX 1080Ti GPU for training and the model takes around 30 minutes to complete a single epoch during training on ModelNet10 dataset.

4.4 Experiments

We evaluate the proposed methodology on 3D object classification and 3D object retrieval tasks using recent datasets: ModelNet10, ModelNet40, McGill 3D, SHREC'17 and OASIS. We also conduct a thorough ablation study to demonstrate the effectiveness of our derivations and design choices.

4.4.1 3D Object Classification Performance

A key feature of our proposed pipeline is the projection of the input 3D shapes into a more discriminative latent shape, before feeding them into convolution layers. One critical advantage of this step is that original subtle differences across object classes are magnified in order to leverage the feature extraction capacity of convolution layers. Therefore, the proposed network should be able to capture more discriminative features in the lower layers, and provide better classification results with a smaller number of layers, compared to other state-of-the-art works which directly extract features from the original shape. To illustrate this, we present a model depth vs accuracy analysis on ModelNet10 and ModelNet40 in Table 4.2, and compare the effectiveness of our network with other comparable state-of-the-art approaches.

Method	Modality	Views	#Layers	ModelNet10	ModelNet40
VoxNet (IROS'15) [Maturana and Scherer, 2015]	Volume	-	-	92.0%	83.0%
3DGAN (NIPS'16) [Wu et al., 2016]	Volume	-	-	91.0%	83.3%
3DShapeNet (CVPR'15) [Wu et al., 2015]	Volume		4-3DConv + 2FC	83.5%	77%
VRN ((NIPS'16)) [Brock et al., 2016]	Volume	-	45Conv	93.6%	91.3 %
GIFT ((CVPR'16)) [Bai et al., 2016]	RGB	64	-	92.4%	83.1%
Pairwise ((CVPR'16)) [Johns et al., 2016]	RGB	12	23Conv	92.8%	90.7%
MVCNN (ICCV'16) [Su et al., 2015]	RGB	12	60Conv + 36FC	-	90.1%
MHBN (CVPR'18) [Yu et al., 2018c]	RGB	6	78Conv + 18FC	95.0%	**94.7%**
DeepPano (SPL'15) [Shi et al., 2015]	RGB		4Conv + 3FC	85.5%	77.63%
ECC (CVPR'17) [Simonovsky and Komodakis, 2017]	Points	-	4Conv + 1FC	90.0%	83.2%
Kd-Networks (ICCV'17) [Klokov and Lempitsky, 2017]	Points	-	15KD	93.5%	91.8%
SO-Net (CVPR'18) [Li et al., 2018b]	Points		11FC	**95.7%**	93.4%
PointNet (CVPR'17) [Qi et al., 2017a]	Points	-	5Conv + 2STL	-	89.2%
LP-3DCNN (CVPR'19) [Kumawat and Raman, 2019]	Points	-	15Conv + 3FC	-	92.1%
Ours	Points	-	2Conv + 1FC	94.2%	91.8%

Table 4.2: Model accuracy vs depth analysis on ModelNet10 and ModelNet40 datasets.

Method	**FLOPS** (inference)	**ModelNet40**
PointNet [Qi et al., 2017a]	14.70B	89.2%
SpecGCN [Wang et al., 2018]	17.79B	92.1%
PCNN [Atzmon et al., 2018]	4.70B	92.3%
PointNet++ [Qi et al., 2017b]	26.04B	91.9%
3DmFV-Net [Ben-Shabat et al., 2017]	16.89B	91.6%
PointCNN [Li et al., 2018c]	25.30B	92.2%
DGCNN [Wang et al., 2018a]	44.27B	**93.5%**
Ours	**1.31**B	91.8%

Table 4.3: Our model complexity is much lower compared to state-of-the-art 3D classification models. The FLOPS (inference time) comparisons are reported according to Li et al. [2018c] settings with 16 batch size.

State-of-the-art work can be mainly categorized into three types: volume based, RGB based and Points based. Volume based methods generally rely on volumetric representation of the 3D shape such as voxels. VoxNet [Maturana and Scherer, 2015] shows the best performance among volume based models, with an accuracy of 92.0% in ModelNet10 and 83.0% in ModelNet40, which is lower than our model's accuracy. It is interesting to see that 3DShapeNets [Wu et al., 2015], and VRN [Brock et al., 2016] have significantly more layers compared to our model, although accuracies are lower. In general, our model performs better and has a lower model depth compared to volume based methods.

RGB based models generally follow the projection of the 3D shape into 2D representations, as an initial step for feature extraction. We perform better than all the RGB based methods, except for MHBN [Yu et al., 2018c], which has accuracies 95.0% and 94.7% over ModelNet10 and ModelNet40 respectively. However, MHBN contains six views and for each view they employ a VGG-M network for initial feature extraction. This results in a significantly complex setup, which contains 96 trainable layers. In contrast, our model uses a single view and three trainable layers. Generally, RGB based models use multiple views, pre-trained deep networks and ensembled models, which results in increased model complexity. In contrast, our model use a single view and does not use pre-trained models, and achieves the second highest performance compared to RGB based models.

Point based models directly consume point clouds. Our model achieves the second best performance in this category, the highest being SO-NET [Li et al., 2018b]. However, SO-NET contains 11 fully connected layers, while our model only contains three layers. Our model is able to outperform the other point based setups, although their model depths are larger.

Overall, our model achieves a performance mark comparable to the best models, with a much shallower architecture. Our model contains the lowest number of trainable layers compared to all the models. This analysis on ModelNet10 and

ModelNet40 clearly reveals the efficiency and better feature extraction capacity of our approach. Table 8.7 depicts the computational efficiency of BCS compared to state-of-the-art. With just 1.31B FLOPs, we outperform the closest contender PCNN [Atzmon et al., 2018] by a significant 3.39B margin.

4.4.2 3D Object Retrieval Performance

In this section, we compare the performance of our approach in 3D object retrieval. We use the McGill 3D dataset and SHREC'17 dataset for our experiments. We first obtain the feature vectors computed by each kernel in the second layer, and concatenate them. Then, we apply an autoencoder on the concatenated vector and retrieve a 1000-dimensional descriptor. Then we measure the cosine similarity between input and target shapes to measure the 3D object retrieval performance. We use the nearest neighbour performance and the evaluation metric. Table 4.4 depicts the results on the McGill Dataset. Out of the six state-of-the-art models compared, our model achieves the best retrieval performance. Table 4.5 illustrates the performance comparison on the SHREC'17 dataset, where our approach gives the second best performance, below Furuya and Ohbuchi [2016]. Figure 4.6 depicts our training curves for polynomial weights and kernel weights. The training curves are obtained for ModelNet10.

4.4.3 Ablation Study

In this section, we conduct an ablation study on our model and discuss various design choices, as illustrated in Figure 4.4. Firstly, we use a single convolution layer instead of two, and achieve an accuracy of 74.2% over ModelNet10. Then, we investigate the effect of using a higher number of convolution layers. We get accuracies 91.3% and 87.5%, when using three and four convolution layers respectively. Therefore, using two convolution layers yields the best performance. An important feature of our convolution layer is the translation of convolution kernels, in addition to rotation. To evaluate the effect of this, we use only rotating kernels and measure the performance, and achieve an accuracy of 80.2%. Therefore, it can be concluded that having the translational movements of the kernel has caused an accuracy increment of 14%, which is significant. Next, we measure the effect of latent space projection. To this end, we use orthogonal polynomials derived in Equations 4.7-4.8 for convolution, instead of making them learnable. This removes the latent space projection of the input, as the original object is reconstructed using spectral moments. After removing the latent space projection, the accuracy is dropped by 20.3%, which clearly reveals the significance of this feature. Then, we replace our convolution layers with volumetric convolution [Ramasinghe et al., 2019a] layers and spherical convolution layers [Cohen et al., 2018b] and get 88.5% and 77.3% accuracy respectively. This shows that our convolution layer has a better feature extraction capacity compared to other convolution operations. One key reason behind this can be the translational movements of our kernels and the combined latent space projection step, which the aforementioned convolution methods lack.

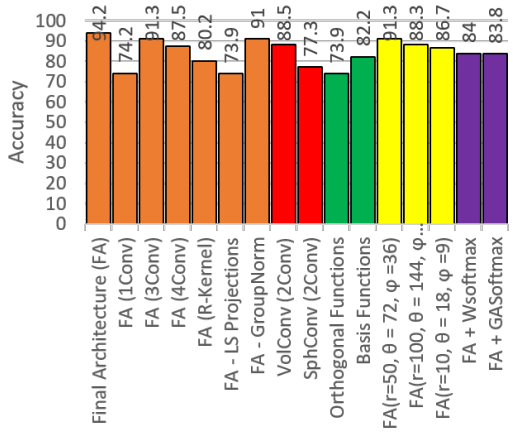


Figure 4.3: Ablation study on ModelNet10 in 3D object classification.

Table 4.4: 3D object retrieval results comparison with state-of-the-art on McGill Dataset.

Method	Accuracy
Bashiri et al. [2019] (arxiv'19)	0.9646%
Zeng et al. [2018] (IET'18)	0.981%
Han et al. [2018] (IP'18)	0.8827%
Tabia et al. [2014] (CVPR'14)	0.977%
Papadakis et al. [2008] (3DOR'w)	0.957%
Lavoué [2012] (TVC'14)	0.925%
Xie et al. [2015] (CVPR'15)	0.988%
Ours	**0.990%**

Moreover, we test our model using basis functions in Eq. 4.21 as the projection functions, instead of learnable functions. Also, we again test the model using orthogonal functions. In both cases, the performance is lower compared to learnable functions. Furthermore, instead of soft-max cross entropy, we use WSoftmax [Liu et al., 2017] and GASoftmax [Liu et al., 2017] and achieve only 84.0% and 83.0% respectively. Therefore, using soft-max cross entropy as the loss function is justified. We also evaluate the effect of sampling density on accuracy. As shown in Figure 4.4, accuracy drops below 94.2%—which is reported by final architecture—when using a denser representation. Similarly, accuracy drops to 86.7% when using $r = 10, \theta = 18, \phi = 9$ as sampling intervals. Therefore, using $r = 25, \theta = 36, \phi = 18$ as in the final architecture seems to be the ideal design choice. We use four different distance measures in the 3D object retrieval task and compare the performance: cosine similarity, Euclidean distance, KL divergence, and Bhattacharya distance. Out of these, cosine similarity yields the best performance, with a mAP of 0.466.

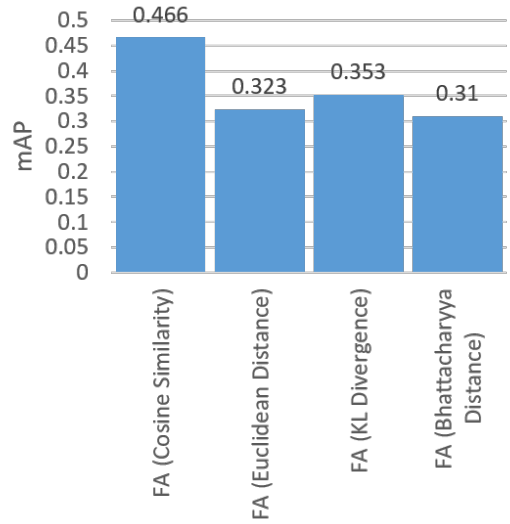


Figure 4.4: Ablation study on SHREC'17 in 3D object retrieval.

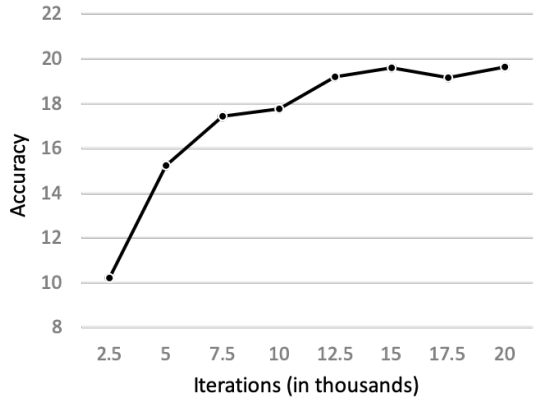


Figure 4.5: Training curves of our architecture on ModelNet10 for polynomial weights.

Figure 4.6: Training curves of our architecture on kernel weights.

Table 4.5: 3D object retrieval results comparison with state-of-the-art on SHREC'17.

Method	mAP
Furuya and Ohbuchi [2016] (BMVC'16)	**0.476**
Esteves et al. [2018b] (ECCV'18)	0.444
Tatsuma and Aono [2009] (TVC'09)	0.418
Bai et al. [2016] (CVPR'16)	0.406
Ours	0.466

Table 4.6: Multi-layer architectures for highly non-polar and textured shape classification. Our model shows an improvement with more layers.

Model	**Accuracy**
Ours (1 Conv layer)	66.3%
Ours (2 Conv layers)	82.7%
Ours (3 Conv layers)	86.7%
Ours (4 Conv layers)	**88.1%**
Ours (5 Conv layers)	87.0%

4.4.4 Classification of Complex Shapes

The proposed convolution layer offers two key advantages: 1) the ability to simultaneously model both shape and texture information, and 2) handling non-polar objects. However, we used ModelNet10 to conduct the ablation study shown in Figure 4.6, which contains relatively simple shapes (i.e. not dense in $\mathbb{B}^3$), and it is clear from the results that the accuracy drops when more than two convolution layers are used. A possible reason for this behaviour is overfitting. Since our convolution layer can capture highly discriminative features from the input functions, using more parameters can cause overfitting on relatively simpler shapes, and thus, a drop in classification accuracy. To test this hypothesis, we conduct an experiment on a more challenging dataset, which contains highly non-polar and textured objects.

In this experiment, we use OASIS-3 dataset [Fotenos et al., 2008] to sample 1000 3D brain scan images. The dataset includes brain scan images from both Alzheimer's disease patients and healthy subjects. A key property of these images is that they have texture information and are highly dense in $\mathbb{B}^3$. Firstly, we split the sampled data in to train and test sets, with 800 and 200 images for each set, respectively. To avoid bias, we include an equal number of Alzheimer cases and healthy cases in both train and test sets. Then, we evaluate different network architectures using the dataset, varying the number of convolution layers. We use cross entropy loss function in this experiment. The results are shown in Table 4.6.

As evident from Table 4.6, the classification accuracy increases with the number of convolution layers, up to four layers. Hence, it can be concluded that more challenging objects allow our model to demonstrate its full capacity.

Table 4.7: Ablation study on the input point cloud density. We sample the input points on a grid (r= 25, θ = 36, ϕ = 18) before feeding to the network.

Original point cloud sampling	**Accuracy**
(r= 250, θ = 200, ϕ = 200)	94.22%
(r= 300, θ = 250, ϕ = 250)	94.21%
(r= 400, θ = 300, ϕ = 300)	**94.23%**
(r= 500, θ = 400, ϕ = 400)	94.20%

4.4.5 Ablation study on input point cloud density

A critical problem associated with directly consuming point clouds, in order to learn features, is the redundancy of information. This property hampers optimal feature learning using neural network based models, by imposing an additional overhead. To verify this, we conduct an ablation study on the density of the input point cloud, and observe the performance variations of our model. The obtained results are reported in Table 4.7. As the results suggest, there is no clear variation of classification performance, although the input sampling density is increased. Therefore, it can be empirically concluded that input point clouds are not optimal to be directly fed to learning networks, due to their inherent redundancy. As a result, significant reduction in their density could still lead to comparable performance with that of the original point cloud.

4.5 Chapter summary

In this chapter, we further extend the equivariant filters derived in chapter 3 to naturally achieve equivariance against the $\mathbb{SE}(3)$ group. We propose a novel approach called 'Blended Convolution and Synthesis' to analyse 3D data, which entails two key operations: (1) learning a 3D descriptor obtained by projecting the input 3D shape into a discriminative latent space and (2) convolving the 3D descriptor in $\mathbb{B}^3$ with roto-translational 3D kernels for extracting features. We derive a novel set of polynomials in $\mathbb{B}^3$, and project the input data into a spectral space using the derived polynomials to join these two operations into a single step. Furthermore, we use a compact representation of the input data to reduce the density of the data distribution and leverage the advantage of convolving functions in $\mathbb{B}^3$. Finally, we present a light-weight architecture and achieve compelling results in 3D object classification and 3D object retrieval tasks. One critical aspect of our work is that the proposed convolution operator relaxes its orthogonality to achieve the equivariance. Hence, although the proposed operator is efficient compared to the state-of-the-art, it can be further optimized and needs further research in this regard.

Chapter 5

Rethinking Conditional-GAN Training

5.1 Introduction

A key attribute that underpins the remarkable learning ability of humans is intuitive physics (see Chapter 1). In this chapter, using a GAN as a use case, we demonstrate that explicit injection of the prior physics of the problem into models, the performance of machine learning models can be significantly elevated. Although we primarily focus on vision based experiments in this chapter, it should be noted that the derived formulae and developed insights are generic across any task.

Generative adversarial networks (GAN) are a family of deep generative models that learns to model data distribution $\mathcal{Y}$ from random latent inputs $z \sim \mathcal{Z}$ using a stochastic generator function $G : \mathcal{Z} \rightarrow \mathcal{Y}$ [Goodfellow et al., 2014a]. A seemingly natural extension from unconditional GANs to conditional GANs (cGAN) can be achieved via conditioning both the discriminator and the generator on a conditioning signal $x \sim \mathcal{X}$. However, such a straightforward extension can cause the models to disregard the conditioning signal x [Isola et al., 2017; Pathak et al., 2016b; Lee et al., 2019a; Ramasinghe et al., 2020b]. To overcome this unsought behavior, a reconstruction loss is typically added to the objective function to penalise the model when it deviates from x. This approach has been widely adapted for diverse tasks including image-to-image translation [Wang et al., 2018; Isola et al., 2017], style transfer [Zhu et al., 2017a; Junginger et al., 2018] and inpainting [Zeng et al., 2019a; Pathak et al., 2016b; Yang et al., 2017a]. However, in spite of the wide usage, naively coupling the reconstruction and the adversarial objectives entails undesirable outcomes as discussed next.

Many conditional generation tasks are ill-posed (many possible solutions exist for a given input), and an ideal generator should be able to capture *one-to-many* mappings between the input and output domains. Note that the stochasticity of G typically depends on two factors, *first* the randomness of z and *second* the dropout [Srivastava et al., 2014]. However, empirical evidence suggests the composition of reconstruction and adversarial losses leads to limited diversity, despite the random seed z. In fact, many prior works have reported that the generator often tends to ignore z, and learns a deterministic mapping from $\mathcal{X}$ to $\mathcal{Y}$, leaving dropout as the

only source of stochasticity [Isola et al., 2017; Lee et al., 2019a; Pathak et al., 2016b; Ramasinghe et al., 2020b]. Additionally, Shao et al. [2018] and Arvanitidis et al. [2017] demonstrated that from a geometrical perspective, latent spaces of generative models (e.g., cGANs) tend to give a distorted view of the generated distribution, thus, the Euclidean paths on the latent manifold do not correspond to the geodesics (shortest paths) on the output manifold. This hinders many possibilities such as clustering in the latent space, better interpolations, higher interpretability and ability to manipulate the outputs. We show that the foregoing problems can be direct consequences of the conventional training approach. Moreover, the naive coupling of regression loss and the adversarial loss can also hamper the visual quality of the generated samples due to contradictory goals of the two objective functions (see Sec. 5.2.1).

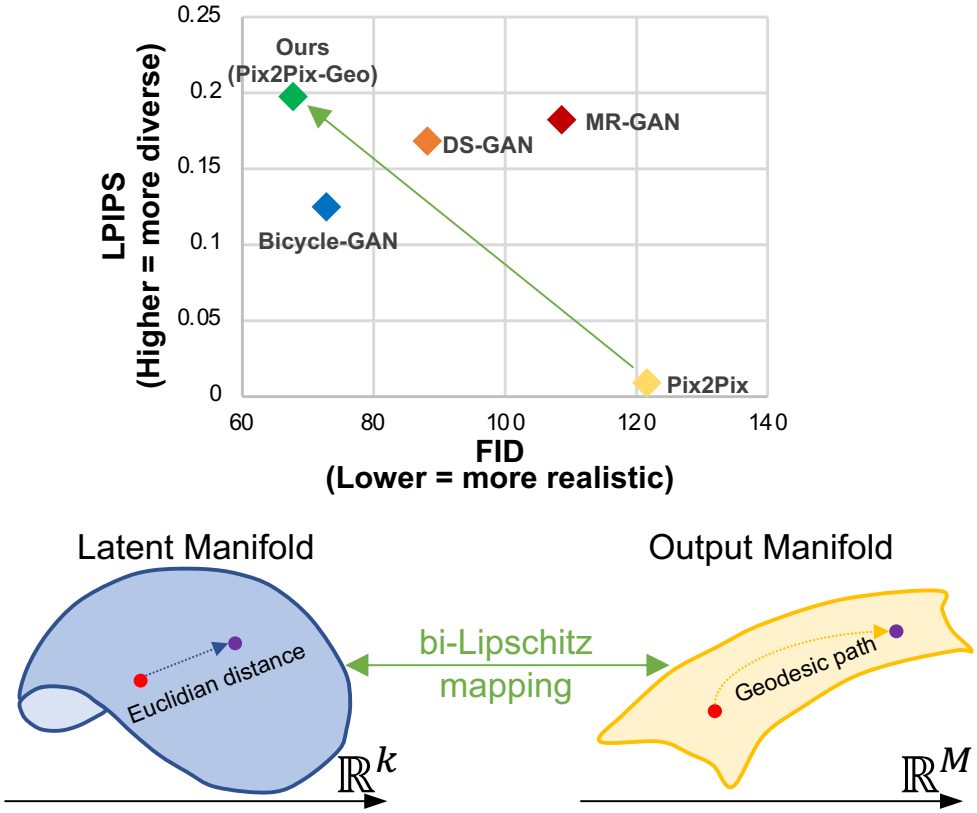


Figure 5.1: *Overview of our approach.* Our training procedure encourages a bi-lipschitz mapping between the latent and generated output manifolds, while mapping the Euclidean shortest paths in the latent manifold to geodesics on the generated output manifold, which allows better diversity and structure. We gain a considerable improvement in both visual quality and the image diversity over our baseline Pix2Pix [Isola et al., 2017], using the same network architecture (*landmark* $\rightarrow$ *faces* image-to-image translation task).

The aforementioned drawbacks have led multi-modal conditional generation approaches to opt for improved objective functions [Yang et al., 2019a; Mao et al.,

2019], and even complex architectures compared to vanilla cGANs [Zhu et al., 2017b; Ramasinghe et al., 2020b; Lee et al., 2019a]. However, in Sec. 5.2, we show that while the existing solutions may improve the diversity and address the loss mismatch, they can also aggravate the topology mismatch and distortion between the latent and output manifolds. In contrast, we argue that these issues are not a consequence of the model capacities of vanilla cGANs [Isola et al., 2017; Pathak et al., 2016b; Mathieu et al., 2015a; Wang et al., 2018], rather a result of sub-optimal training procedures that are insensitive to their underlying geometry. As a remedy, we show that the foregoing problems can be addressed by systematically encouraging a structured bijective and a continuous mapping, i.e., a homeomorphism, between the latent and the generated manifolds. Furthermore, the structure of the latent space can be enhanced by enforcing bi-lipschitz conditions between the manifolds. To this end, we introduce a novel training procedure and an optimization objective to encourage the generator and the latent space to preserve a bi-lipschitz mapping, while matching the Euclidean paths in the latent space to geodesics on the output manifold.

We choose Pix2Pix [Isola et al., 2017], a vanilla cGAN, and modify its training procedure to demonstrate that the proposed mapping improves the realism of the outputs by removing the loss mismatch, enhances the structure of the latent space, and considerably improves the output diversity. As the formulation of our conditional generation approach is generic, we are able to evaluate the modified Pix2Pix model, dubbed *Pix2Pix-Geo*, on a diverse set of popular image-to-image translation tasks. We show that with the modified training approach, our Pix2Pix-Geo significantly improves the prediction diversity of the cGAN compared to the traditional baseline procedure and achieves comparable or better results than the more sophisticated *state-of-the-art* models. Most importantly, our modifications are purely aimed at the optimization procedure, which demands no architectural modifications to vanilla cGANs.

5.2 Motivation

In conditional generative modeling, the ground truth (output) data distribution $\mathcal{Y} \subseteq \mathbb{R}^M$ is conditioned on an input distribution $\mathcal{X} \subseteq \mathbb{R}^d$. Consider the data distribution $\mathcal{Y}_{|x_p} \subset \mathcal{Y}$ conditioned on $x_p \in \mathcal{X}$. Then, the following adversarial objective function is used to optimize the generator G by playing a min-max game against a discriminator D, thereby approximating the distribution $\mathcal{Y}_{|x_p}$,

$$L_{adv} = \min_{G} \max_{D} \mathop{\mathbb{E}}_{y \sim \mathcal{Y}} [\Phi(D(x_p, y))] + \mathop{\mathbb{E}}_{z \sim \zeta} [\Phi(1 - D(G(x_p, z))], \tag{5.1}$$

where Φ is a suitably chosen monotone function, $y \sim \mathcal{Y}$ and $z \in \mathbb{R}^k$ is a latent vector sampled from a prior distribution ζ. It has been widely observed that using the above objective function in isolation, pushes the models to generate samples that are not strongly conditioned on the input signal x_p [Isola et al., 2017; Yang et al., 2019a; Lee et al., 2019a; Zhu et al., 2017b]. Hence, the conventional cGAN loss couples a reconstruction loss L_r (typically ℓ_1 or ℓ_2) with Eq. 5.1. However, as alluded in Sec. 5.1,

this entails several drawbacks: **a)** contradictory goals of the loss components, **b)** conditional mode collapse, and **c)** insensitivity to the underlying manifold geometry. Below, we will explore each of these issues in detail and contrast our method against several recent attempts that have been proposed to resolve them. From this point onwards, our analysis is focused on the conditional setting and we do not explicitly denote the conditioning signal x in our notations, to avoid clutter.

5.2.1 Mismatch b/w adversarial & reconstruction losses

The optimal generator for the adversarial loss is,

$$G^* = \underset{G}{\operatorname{argmin}} \Big(\mathbf{JSD} \Big[p_g(\bar{y}) \| p_d(y) \Big] \Big), \tag{5.2}$$

where **JSD** is the Jensen–Shannon divergence, y is the ground-truth and $\bar{y} = G(z)$ is the generated output.

Now, let us consider the expected ℓ_1 loss, $\mathbb{E}_{y,z} \big| y - \bar{y}(z) \big|$. Then,

$$\mathbb{E}_{y,z} \big| y - \bar{y}(z) \big| = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \big| y - \bar{y}(z) \big| \, p(y) p(z|y) dz dy. \tag{5.3}$$

To find the minimum of the above, we find the value where the subderivative of the $\bar{y}(z)$ equals to zero as,

$$\frac{d}{d\bar{y}} [\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \big| y - \bar{y}(z) \big| \, p(y) p(z|y) dy dz] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} -\text{sign}(y - \bar{y}(z)) p(y) p(z|y) dz dy = 0. \tag{5.4}$$

$$\int_{-\infty}^{\bar{y}} \int_{-\infty}^{\infty} -\text{sign}(y - \bar{y}(z)) p(y) p(z|y) dz dy + \int_{\bar{y}}^{\infty} \int_{-\infty}^{\infty} -\text{sign}(y - \bar{y}(z)) p(y) p(z|y) dz dy = 0. \tag{5.5}$$

$$\int_{-\infty}^{\bar{y}} \int_{-\infty}^{\infty} p(y) p(z|y) dz dy = \int_{\bar{y}}^{\infty} \int_{-\infty}^{\infty} p(y) p(z|y) dz dy. \tag{5.6}$$

Since z is randomly sampled, with enough iterations $p(z) = p(z|y)$. Then,

$$\int_{-\infty}^{\bar{y}} p(y) dy \int_{-\infty}^{\infty} p(z) dz = \int_{\bar{y}}^{\infty} p(y) \int_{-\infty}^{\infty} p(z) dz, \tag{5.7}$$

$$\int_{-\infty}^{\bar{y}} p(y) dy = \int_{\bar{y}}^{\infty} p(y) dy, \tag{5.8}$$

which means that the probability mass to left of $\bar{y}$ is equal to the probability mass to the right of $\bar{y}$. Therefore, $\bar{y}$ is the median of the distribution $p(y)$. Hence, unless $p_d(y)$ is unimodal with a sharp peak, the optimal generator for the ℓ_1 loss does not equal G^*. With a similar approach, it can be shown that ℓ_2 concentrates p_g near the average of the ground truth distribution. Hence, these contradictory goals of L_r and

L_{adv} force the model to reach a compromise, thereby settling in a sub-optimal position in the parameter space.

Now, consider a function f such that $f(z) = y$ and $p(f(z)) = p_d$. Then, the corresponding cumulative distribution is,

$$F(y) = p(f(z) \leq y). \tag{5.9}$$

Therefore, $p(f(z))$ can be obtained as,

$$p(f(z)) = \frac{\partial}{\partial y_1} \cdots \frac{\partial}{\partial y_M} \int_{\{z^* \in \mathbb{R}^k | f(z^*) \leq f(z)\}} p(z) d^k z. \tag{5.10}$$

According to Eq. 5.10, f should be differentiable almost everywhere with a positive definite $\mathbf{J}_f^T \mathbf{J}_f$, where $\mathbf{J}_f$ is the Jacobian of f. Recall the Rademacher theorem,

Theorem 1: Let $\mathcal{Z}$ be an open subset of $\mathbb{R}^k$ and $g : \mathcal{Z} \rightarrow \mathbb{R}^M$ a lipschitz function. Then, g differentiable almost everywhere (with respect to the Lebesgue measure λ). That is, there is a set $E \subset \mathcal{Z}$ with $\lambda(\mathcal{Z}/E) = 0$ and such that for every $z \in E$ there is a linear function $L_z : \mathbb{R}^k \rightarrow \mathbb{R}^M$ with

$$\lim_{z^* \rightarrow z} \frac{g(z) - g(z^*) - L_z(z^* - z)}{|z^* - z|} = 0. \tag{5.11}$$

Recall that our loss function enforce a bilipschitz mapping between the manifolds with a positive definite metric tensor, hence, G^{-1} and G is differentiable almost everywhere. That is, given enough flexibility, G converges to f almost surely, i.e., $\mathbf{JSD}[p_g(\bar{y})||p_d(y)] \approx 0$. Hence, our adversarial loss and the other loss components are not contradictory. This argument is empirically backed by our experiments, as we show that the realism of the outputs of the Pix2Pix [Isola et al., 2017] model can be significantly improved using the proposed method. Both Bicycle-GAN [Zhu et al., 2017b] and MR-GAN [Lee et al., 2019a] remove this loss mismatch using a bijective mapping and by matching the moments of the generated and target distributions, respectively. However, their training procedures can disrupt the structure of the latent space (see Sec. 5.2.3).

5.2.2 Conditional mode collapse

(Conditional) mode collapse is a commonly observed phenomenon in cGANs [Isola et al., 2017; Lee et al., 2019a; Pathak et al., 2016b; Ramasinghe et al., 2020b]. In this section, we discuss how the traditional training procedure may cause mode collapse and show that the existing solutions tend to derange the structure of the latent manifold.

Definition 5.1. *[Yang et al., 2019a]. A mode $\mathcal{H}$ is a subset of $\mathcal{Y}$ s.t. $max_{y \in \mathcal{H}} \left\| y - y^* \right\| < \alpha$ for an output y^* and $\alpha > 0$. Then, at the training phase, z_1 is attracted to $\mathcal{H}$ by ϵ from an optimization step if $\|y^* - G_{\theta(t+1)}(z_1)\| + \epsilon < \|y^* - G_{\theta(t)}(z_1)\|$, where $\theta(t)$ are the parameters of G at time t.*

Proposition 5.1. *[Yang et al., 2019a]. Suppose z_1 is attracted to $\mathcal{H}$ by ϵ. Then, there exists a neighbourhood $\mathcal{N}(z_1)$ of z_1, such that z is attracted to $\mathcal{H}$ by $\epsilon/2, \forall z \in \mathcal{N}(z_1)$. Furthermore, the radius of $\mathcal{N}(z_1)$ is bounded by an open ball of radius r where the radius is defined as,*

$$r = \epsilon \left(4 \inf_{z} \left\{ max(\tau(t), \tau(t+1)) \right\} \right)^{-1}, \tag{5.12}$$

where $\tau(t) = \frac{\left\| G_{\theta(t)}(z_1) - G_{\theta(t)}(z) \right\|}{\|z_1 - z\|}$.

Proposition 5.1 yields that by maximizing $\tau(t)$ at each optimization step, one can reduce the effect of mode collapse. Noticeably, the traditional training approach does not impose such a constraint. Thus, $\|z_1 - z\|$ can be arbitrary large for a small change in the output and the model is prone to mode collapse. As a result, DSGAN [Yang et al., 2019a], MS-GAN [Mao et al., 2019] and MR-GAN [Lee et al., 2019a] (implicitly) aim to maximize τ. Although maximizing τ improves the diversity of the model, it also causes an undesirable side-effect, as discussed next.

5.2.3 Loss of structure b/w output & latent manifolds

A *sufficiently smooth* generative model $G(z)$ can be considered as a surface model [Gauss, 1828]. This has enabled analyzing *latent variable generative models* using Riemannian geometry [Arvanitidis et al., 2020; Wang et al., 2020a; Shao et al., 2018; Khrulkov and Oseledets, 2018]. Here, we utilize the same perspective: a generator can be considered as a function that maps low dimensional latent codes $z \in \mathcal{M}_z \subseteq \mathbb{R}^k$ to a data manifold $\mathcal{M}_y$ in a higher dimensional space $\mathbb{R}^M$ where $\mathcal{M}_z$ and $\mathcal{M}_y$ are Riemannian manifolds, i.e., z encodes the intrinsic coordinates of $\mathcal{M}_y$. Note that increasing τ in an unconstrained setting does not impose any structure in the latent space. That is, since the range of $\left\| G(z_1) - G(z) \right\|$ is arbitrary in different neighbourhoods, stark discontinuities in the output space can occur, as we move along $\mathcal{M}_z$. Further note that Bicycle-GAN also does not impose such continuity on the mapping. Thus, the distance between two latent codes on $\mathcal{M}_z$ may not yield useful information such as the similarity of outputs. This is a significant disadvantage, as we expect the latent space to encode such details. Interestingly, if we can induce a continuous and a bijective mapping, i.e., a homeomorphism between $\mathcal{M}_y$ and $\mathcal{M}_z$, while maximizing τ, the structure of the latent space can be preserved to an extent.

However, a homeomorphism does not reduce the distortion of $\mathcal{M}_z$ with respect to $\mathcal{M}_z$. In other words, although the arc length between z_1 and z is smoothly and monotonically increasing with the arc length between $G(z_1)$ and $G(z)$ under a homeomorphism, it is not bounded. This can cause heavy distortions between the manifolds. More formally, maximizing τ encourages maximizing the components of the Jacobian $\mathbf{J}^{d \times k} = \frac{\partial G}{\partial z}$ at small intervals. If G is sufficiently smooth, the Riemannian metric $\mathbf{M} = \mathbf{J}^T \mathbf{J}$ can be obtained, which is a positive definite matrix that varies

smoothly on the latent space. Further, by the Hadamard inequality,

$$\det(\mathbf{M}) \leq \prod_{i=0}^{k} \|\mathbf{J}_i\|^2, \tag{5.13}$$

where $\mathbf{J}_i$ are the columns of $\mathbf{J}$. This leads to an interesting observation. In fact, $\det(\mathbf{M})$ can be seen as a measure of distortion of the output manifold with respect to the latent manifold. Therefore, although maximizing τ acts as a remedy for mode collapse, even under a homeomorphism, it can increase the distortion between $\mathcal{M}_z$ and $\mathcal{M}_y$.

In conditional generation tasks, it is useful to reduce the distortion between the manifolds. Ideally, we would like to match the Euclidean paths on $\mathcal{M}_z$ to geodesics on $\mathcal{M}_y$, as it entails many advantages (see Sec. 5.1). Consider a small distance Δz on $\mathcal{M}_z$. Then, the corresponding distance in $\mathcal{M}_y$ can be obtained using Taylor expansion as,

$$G(\Delta z) = \mathbf{J}\Delta z + \Theta(\|\Delta z\|) \approx \mathbf{J}\Delta z, \tag{5.14}$$

where $\Theta(\|\Delta z\|)$ is a function which approaches zero more rapidly than Δz. It is evident from Eq. 5.14 that the corresponding distance on $\mathcal{M}_y$ for Δz is governed by $\mathbf{J}$. Ideally, we want to constrain $\mathbf{J}$ in such a way that small Euclidean distances Δz encourage the output to move along geodesics in $\mathcal{M}_y$. However, since random sampling does not impose such a constraint on $\mathbf{J}$, the traditional training approach and the existing solutions fail at this. Interestingly, it is easy to deduce that geodesics avoid paths with high distortions [Gallot et al., 1990]. Recall that minimizing τ along optimization curves reduces the distortion of $\mathcal{M}_y$, thus, encourages Δz to match geodesics on $\mathcal{M}_y$. However, minimizing τ can also lead to mode collapse as discussed in Sec. 5.2.2.

Although the above analysis yields seemingly contradictory goals, one can achieve both by establishing a bi-lipschitz mapping between $\mathcal{M}_y$ and $\mathcal{M}_z$, as it provides both an upper and a lower-bound for τ. Such a mapping between $\mathcal{M}_z$ and $\mathcal{M}_y$ provides a soft bound for $\det(\mathbf{M})$, and prevents mode collapse while preserving structure of the latent manifold.

Remark 1: *An ideal generator function should be homeomorphic to its latent space. The structure of the latent space can be further improved by inducing a bi-lipschitz mapping between the latent space and generator function output.*[1]

Based on the above Remark, we propose a training approach that encourages a structured bi-lipschitz mapping between the latent and the generated manifolds and show that in contrast to the existing methods, the proposed method is able to address all three issues mentioned above.

5.3 Discussion on Related works

Conditional generative modeling. Generative modeling has shown remarkable progress since the inception of Variational Autoencoders (VAE) [Kingma and Welling,

[1]Note that every bi-lipschitz mapping is a homeomorphism.

2013] and GANs [Goodfellow et al., 2014a]. Consequently, the conditional counterparts of these models have dominated the conditional generative tasks [Isola et al., 2017; Zhang et al., 2016; Ramasinghe et al., 2019d; Bao et al., 2017; Lee et al., 2018; Zeng et al., 2019b]. However, conditional generation in multimodal spaces remain challenging, as the models need to exhibit a form of stochasticity in order to generate diverse outputs. To this end, Zhu et al. [2017b] proposed a model where they enforce a bijective mapping between the outputs and the latent spaces. Yang et al. [2019a], Mao et al. [2019], and Lee et al. [2019a] introduced novel objective functions to increase the distance between the samples generated for different latent seeds. Chang et al. [2019] used separate variables that can be injected at the inference to change the effects of loss components that were used during the training. In contrast, VAE based methods aim to explicitly model the latent probability distribution and at inference, diverse samples are generated using different latent seeds. However, typically, the latent posterior distribution of the VAE is approximated by a Gaussian, hence, the ability to model more complex distributions is hindered. As a solution, Maaløe et al. [2016] suggested using auxiliary variables to hierarchically generate more complex distributions, using a Gaussian distribution as the input. Normalizing Flows [Rezende and Mohamed, 2015] are similar in concept, where the aim is to generate more complex posterior distributions hierarchically. They apply a series of bijective mappings to an initial simple distribution, under the condition that the Jacobian of these mappings are easily invertible.

Geometrical analysis of generative models. Recent works have discovered intriguing geometrical properties of generative models [Arvanitidis et al., 2017; Shao et al., 2018; Arvanitidis et al., 2020]. These works apply post-train analysis on the models and confirm that Euclidean paths in the latent space do not map to geodesics on the generated manifold. In contrast, we focus on preserving these properties while training the model. In another direction, Wang et al. [2020a] introduced a loss function that forces the real and generated distributions to be matched in the topological feature space. They showed that by using this loss, the generator is able to produce images with the same structural topology as in real images. Similarly, Khrulkov and Oseledets [2018] proposed a novel performance metric for GANs by comparing geometrical properties of the real and generated data manifolds. Different to our work, these methods do not ensure homeomorphism between the latent and generated manifolds.

5.4 Methodology

Our approach is motivated by three goals. **R1**) A bi-lipschitz mapping needs to be established between $\mathcal{M}_z$ and $\mathcal{M}_y$ as,

$$\frac{1}{C} d_{\mathcal{M}_z}(z^p, z^q) \leq d_{\mathcal{M}_y}(\phi^{-1}(z^p), \phi^{-1}(z^q)) \leq C d_{\mathcal{M}_z}(z^p, z^q), \tag{5.15}$$

where $d_{\cdot}(\cdot)$ is the geodesic distance in the denoted manifold, z^p and z^q are two latent codes, and C is a constant. Further, $\phi : \mathcal{M}_y \rightarrow \mathcal{M}_z$ is a continues global chart map

with its inverse ϕ^{-1}. **R2**) Euclidean distances in $\mathcal{M}_z$ should map to geodesics in $\mathcal{M}_y$ for better structure. **R3**) The geodesic distance between two arbitrary points on $\mathcal{M}_y$ should correspond to a meaningful metric, i.e., pixel distance (note the loss mismatch issue is implicitly resolved by **R1**). In Sec. 5.4.1, we explain our training procedure.

5.4.1 Geodesics and global bi-lipschitz mapping

In this Section, we discuss the proposed training procedure in detail. Consider a map $\gamma_{\mathcal{M}_z} : I \to \mathcal{M}_z$, that parameterizes a curve on $\mathcal{M}_z$ using $t \in I \subset \mathbb{R}$. Then, there also exists a map $(G \circ \gamma_{\mathcal{M}_z}) \equiv \gamma_{\mathcal{M}_y} : I \to \mathcal{M}_y$. If $\gamma_{\mathcal{M}_y}$ is a geodesic, this mapping can be uniquely determined by a $p \in \mathcal{M}_y$ and an initial velocity $V \in T_p\mathcal{M}_y$, where $T_p\mathcal{M}_y$ is the tangent space of $\mathcal{M}_y$ at p as shown below: [2].

Let $\mathcal{M}$ be a manifold and $\gamma : I \to \mathcal{M}$ be a geodesic satisfying $\gamma(t_0) = p$, $\dot{\gamma}(t_0) = V$, where $p \in \mathcal{M}$, $V \in T_p\mathcal{M}$ and $I \subset \mathbb{R}$. $T_p\mathcal{M}$ is the tangent bundle of $\mathcal{M}$. Let us choose coordinates (x^i) on a neighborhood U of p, s.t. $\gamma(t) = (x^1(t), x^2(t), \ldots, x^n(t))$. For γ to be a geodesic it should satisfy the condition,

$$\ddot{x}^k + \dot{x}^i(t)\dot{x}^j(t)\Gamma^k_{ij}(x(t)) = 0, \tag{5.16}$$

where Eq. 5.16 is written using Einstein summation. Here, Γ are Christoffel symbols that are functions of the Riemannian metric. Eq. 5.16 can be interpreted as a second-order system of ordinary differential equations for the functions $x^i(t)$. Using auxiliary variables $v_i = \dot{x}^i$, it can be converted to an equivalent first-order system as,

$$\dot{x}^k(t) = v^k(t), \tag{5.17}$$

$$\dot{v}^k(t) = -v^i(t)v^j(t)\Gamma^k_{ij}(x(t)). \tag{5.18}$$

On the other hand, existence and uniqueness theorems for first-order ODEs ensure that for any $(p, V) \in U \times \mathbb{R}^n$, there exists a unique solution $\eta : (t_0 - \epsilon, t_0 + \epsilon) \to U \times \mathbb{R}^n$, where $\epsilon > 0$, satisfying the initial condition $\eta(t_0) = (p, V)$.

Now, let us define two geodesics, $\gamma, \beta : I \to \mathcal{M}$ in an open interval with $\gamma(t_0) = \beta(t_0)$ and $\dot{\gamma}(t_0) = \dot{\beta}(t_0)$. By the above mentioned uniqueness theorem, they agree on some neighborhood of t_0. Let α be the supremum of numbers b s.t. they agree on $[t_0, b]$. If $\alpha \in I$, then using continuity it can be seen that, $\gamma(\alpha) = \beta(\alpha)$ and $\dot{\gamma}(\alpha) = \dot{\beta}(\alpha)$. Then, by applying local uniqueness in a neighborhood of α, the curves agree on a slightly larger interval, which is contradiction. Hence, arguing the similarity to the left of t_0, it can be seen that the curves agree on all I.

This is a useful result for us, as we can obtain a unique point $p' \in \mathcal{M}_y$ only by defining an initial velocity and following $\gamma_{\mathcal{M}_y}$ for a time T (note that we do not consider the highly unlikely scenario where two geodesics may overlap exactly at $t = T$).

[2] V depends on p and hence the dependency of the mapping $\gamma_{\mathcal{M}p}$ on p does need to be explicitly denoted.

To find the geodesic between two points on a Riemannian manifold, ideally, $\gamma_{\mathcal{M}_z}$ should be constrained as,

$$\begin{aligned}\ddot{\gamma}_{\mathcal{M}_z} = \quad & -\tfrac{1}{2}\mathbf{M}^{-1}\Bigg[2(\mathbf{I}_k \otimes \dot{\gamma}^T_{\mathcal{M}_z})\tfrac{\partial \text{vec}(\mathbf{M})}{\partial \gamma_{\mathcal{M}_z}}\dot{\gamma}_{\mathcal{M}_z} \\ & - \left[\tfrac{\partial \text{vec}(\mathbf{M})}{\partial \gamma_{\mathcal{M}_z}}\right]^T (\dot{\gamma}_{\mathcal{M}_z} \otimes \dot{\gamma}_{\mathcal{M}_z})\Bigg],\end{aligned} \tag{5.19}$$

where $\mathbf{M}^{k\times k} = \mathbf{J}^T_{\phi^{-1}}\mathbf{J}_{\phi^{-1}}$ is the metric tensor, $\mathbf{J}_{\phi^{-1}}$ is the Jacobian, and $\otimes$ is the outer product [Arvanitidis et al., 2017]. This approach is expensive, as it requires calculating the Jacobians in each iteration and moreover, causes unstable gradients. However, in practice, an exact solution is not needed, hence, we adapt an alternate procedure to encourage $\gamma_{\mathcal{M}_y}$ to be a geodesic, and use Eq. 5.19 only for evaluation purposes in Sec. 6.6. Since geodesics are locally length minimizing paths on a manifold, we encourage the model to minimize the curve length $L(\gamma_{\mathcal{M}_y}(t))$ on $\mathcal{M}_y$ in the range $t = [0, T]$. $L(\gamma_{\mathcal{M}_y}(t))$ can be measured as follows:

$$\begin{aligned}L(\gamma_{\mathcal{M}_y}(t)) &= \int_0^1 \left\| \frac{\partial G \circ \gamma_{\mathcal{M}_z}(t)}{\partial t} \right\| dt, \\ &= \int_0^1 \left\| \frac{\partial G \circ \gamma_{\mathcal{M}_z}(t)}{\partial \gamma_{\mathcal{M}_z}(t)} \frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t} \right\| dt.\end{aligned} \tag{5.20}$$

Eq. 5.20 can be expressed using the Jacobian $\mathbf{J}_{\phi^{-1}}$ as,

$$= \int_0^1 \left\| \mathbf{J}_{\phi^{-1}} \frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t} \right\| dt, = \int_0^1 \sqrt{\left[\mathbf{J}_{\phi^{-1}} \frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t}\right]^T \mathbf{J}_{\phi^{-1}} \frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t}} dt. \tag{5.21}$$

Since $\mathbf{M}^{k\times k} = \mathbf{J}^T_{\phi^{-1}}\mathbf{J}_{\phi^{-1}}$,

$$= \int_0^1 \sqrt{\left[\frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t}\right]^T \mathbf{M} \frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t}} dt.$$

Considering small $\Delta t = \frac{T}{N}$,

$$\approx \sum_{i=0}^{N} \sqrt{\left[\frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t}\right]^T \mathbf{M} \frac{\partial \gamma_{\mathcal{M}_z}(t)}{\partial t}} \Delta t = \sum_{i=0}^{N-1} \sqrt{\dot{z}_i^T \mathbf{M} \dot{z}_i} \Delta t. \tag{5.22}$$

Further, $\left\|G(\Delta z)\right\| = \Delta z^T \mathbf{M} \Delta z > 0, \forall \Delta z > 0$, i.e., $\mathbf{M}$ is positive definite (since $\frac{dG}{dz} \neq 0$, which is discussed next). Hence, by Hadamard inequality (Eq. 5.13), it can be seen that we can minimize the $\frac{\partial G}{\partial \mathbf{z}}$, in order for $L(\gamma_{\mathcal{M}_y}(t))$ to be minimized. But on the other hand, we also need $\gamma_{\mathcal{M}_y}(T) = y$. Therefore, we minimize $\frac{\partial G}{\partial z}$ at small

intervals along the curve by updating the generator at each $t_i = i\Delta t$,

$$\begin{aligned}\mathcal{L}_{gh}(t_i, z_{t_i}, y, x) =& \|[\alpha(t_i) \cdot y - (1 - \alpha(t_i)) \cdot G(z_{t_0}, x)] \\ & - G(z_{t_i}, x)\|,\end{aligned} \tag{5.23}$$

where $i = 0, 1, \ldots, N$, and $\alpha(\cdot)$ is a monotonic function under the conditions $\alpha(0) = 0$ and $\alpha(T) = T$. Another perspective for the aforementioned procedure is that the *volume element* ϵ of $\mathcal{M}_y$ can be obtained as $\epsilon = \sqrt{|\det(\mathbf{M})|}dz$. Therefore, $\det(\mathbf{M})$ is a measure of the distortion in $\mathcal{M}_y$ with respect to $\mathcal{M}_z$ and geodesics prefer to avoid regions with high distortions. The procedure explained so far encourages a bi-lipschitz mapping as in Eq. 5.15 and satisfies **R1**.

Proof. The loss $\mathcal{L}_{gh}$ in Sec. 5.4.1 forces $G(z)$ to be smooth and $\det(\frac{\partial(G)}{dz}) > 0$, hence, $\det(\mathbf{G}) > 0$. Let $T(\cdot)$ denote the unit tangent bundle of a given manifold. Then, the map $df : T\mathcal{M}_z \to T\mathcal{M}_y$ is also smooth. Therefore, the function $h(p) = |df(p)|$, $p \in T\mathcal{M}_z$ is continuous too. Let $1/C$ and K denote its minimum and maximum, respectively. Therefore, for every unit speed piecewise-smooth path $\gamma : [a, b] \to \mathcal{M}_z$, the length of its image in $\mathcal{M}_y$ is,

$$L(G \circ \gamma) = \int_a^b \left\| \frac{\partial(G \circ \gamma)}{dt} \right\| dt. \tag{5.24}$$

Further,

$$\frac{1}{C} \int_a^b \left\| \frac{\partial \gamma}{dt} \right\| dt < L(G \circ \gamma) < K \int_a^b \left\| \frac{\partial \gamma}{dt} \right\| dt. \tag{5.25}$$

If $C < K$,

$$\frac{1}{K} \int_a^b \left\| \frac{\partial \gamma}{dt} \right\| dt < L(G \circ \gamma) < K \int_a^b \left\| \frac{\partial \gamma}{dt} \right\| dt. \tag{5.26}$$

On the contrary, if $C \geq K$,

$$\frac{1}{C} \int_a^b \left\| \frac{\partial \gamma}{dt} \right\| \leq L_{\mathcal{M}_y}(G \circ \gamma) \leq C \int_a^b \left\| \frac{\partial \gamma}{dt} \right\|. \implies \frac{1}{C} L_{\mathcal{M}_z}(\gamma) \leq L_{\mathcal{M}_y}(\gamma) \leq C L_{\mathcal{M}_z}(\gamma). \tag{5.27}$$

□

Since the geodesic distances are length minimizing curves on $\mathcal{M}_y$ and $\mathcal{M}_z$, it follows that,

$$\frac{1}{C} d_{\mathcal{M}_z}(z^p, z^q) \leq d_{\mathcal{M}_y}(\phi^{-1}(z^p), \phi^{-1}(z^q)) \leq C d_{\mathcal{M}_z}(z^p, z^q), \tag{5.28}$$

where, $d(\cdot, \cdot)$ are the geodesic distances and C is a constant.

Proof. Consider a geodesic $\gamma_V : I \to \mathcal{M}$, defined in an open interval $I \subset \mathbb{R}$, with an initial velocity $V \in T\mathcal{M}$. Let us also define a curve $\tilde{\gamma}(t) = \gamma_V(ct)$. Then, $\tilde{\gamma}(0) =$

$\gamma_V(0) = p \in \mathcal{M}$. Writing $\gamma_V(t) = (\gamma^1(t), \gamma^2(t), \ldots, \gamma^n(t))$ in local coordinates,

$$\dot{\tilde{\gamma}}(t) = \frac{d}{dt}\gamma_V^i(ct) = c\dot{\gamma}_V^i(ct). \tag{5.29}$$

Further, it follows that $\dot{\tilde{\gamma}} = c\dot{\gamma}(0) = cV$.

Next, let D_t and $\tilde{D}_t$ denote the covariant differentiation operators along γ_V and $\tilde{\gamma}$, respectively. Then,

$$\tilde{D}_t\dot{\tilde{\gamma}}(t) = [\frac{d}{dt}\dot{\tilde{\gamma}}^k(t) + \Gamma_{ij}^k(\tilde{\gamma}(t))\dot{\tilde{\gamma}}^i(t)]\partial_k \tag{5.30}$$

$$= (c^2\ddot{\gamma}^k(ct) + c^2\Gamma_{ij}^k(\gamma_V(ct))\dot{\gamma}_V^i(ct)\dot{\gamma}^j(ct))\partial_k \tag{5.31}$$

$$c^2 D_t\dot{\gamma}(ct) = 0. \tag{5.32}$$

Hence, $\tilde{\gamma}$ is a geodesic, and therefore, $\tilde{\gamma} = \gamma_{cV}$. □

Further, as shown before, it can be shown that the enforced bijective mapping removes the loss mismatch between the adversarial and reconstruction losses, hence, improves the visual quality of the outputs (see Fig. 5.3).

According to **R2**), proposed training mechanism should map Euclidean paths on $\mathcal{M}_z$ to geodesics on $\mathcal{M}_y$. Therefore, we move z along Euclidean paths when minimizing $\mathcal{L}_{gh}$, which also ensures that $\mathcal{M}_z \subseteq \mathbb{R}^k$. Furthermore, we constrain $\dot{z}$ to be a constant, for simplicity. Since we ensure that the distortion of $\mathcal{M}_y$ along the paths of z are minimum, in practice, it can be observed that the Euclidean paths on the latent space are approximately matched to the geodesics on the output manifold (Fig. 5.7).

Further, let $\gamma_V(t)$ be a geodesic curve with an initial velocity V. Then, it can be shown

$$\gamma_{cV}(t) = \gamma_V(ct), \tag{5.33}$$

where c is a constant.

Proof. Consider a geodesic $\gamma_V : I \to \mathcal{M}$, defined in an open interval $I \subset \mathbb{R}$, with an initial velocity $V \in T\mathcal{M}$. Let us also define a curve $\tilde{\gamma}(t) = \gamma_V(ct)$. Then, $\tilde{\gamma}(0) = \gamma_V(0) = p \in \mathcal{M}$. Writing $\gamma_V(t) = (\gamma^1(t), \gamma^2(t), \ldots, \gamma^n(t))$ in local coordinates,

$$\dot{\tilde{\gamma}}(t) = \frac{d}{dt}\gamma_V^i(ct) = c\dot{\gamma}_V^i(ct). \tag{5.34}$$

Further, it follows that $\dot{\tilde{\gamma}} = c\dot{\gamma}(0) = cV$.

Next, let D_t and $\tilde{D}_t$ denote the covariant differentiation operators along γ_V and $\tilde{\gamma}$, respectively. Then,

$$\tilde{D}_t\dot{\tilde{\gamma}}(t) = [\frac{d}{dt}\dot{\tilde{\gamma}}^k(t) + \Gamma_{ij}^k(\tilde{\gamma}(t))\dot{\tilde{\gamma}}^i(t)]\partial_k \tag{5.35}$$

$$= (c^2\ddot{\gamma}^k(ct) + c^2\Gamma_{ij}^k(\gamma_V(ct))\dot{\gamma}_V^i(ct)\dot{\gamma}^j(ct))\partial_k \tag{5.36}$$

Algorithm 1: Training algorithm

sample inputs $\{x_1, x_2, ..., x_J\} \sim \mathcal{X}$;
sample outputs $\{y_1, y_2, ..., y_J\} \sim \mathcal{Y}$;
for *k epochs* **do**
  for *x in* χ **do**
    $z \sim \mathcal{B}_r^k$ // Sample z from k-ball with a small radius r
    $V \leftarrow \nabla_z \left\| y - G(z_{t_0}) \right\|$
    $t \leftarrow 0$
    for *T steps* **do**
      sample noise: $e \sim \mathcal{N}(0, \epsilon_1)$; $\epsilon_1 1$
      update G: $\nabla_w \mathcal{L}_{gh}(y, z, x, t)$
      update z: $z \leftarrow z + \eta V + e$
      update t: $t \leftarrow t + 1$
    update G: $\nabla_w [\mathcal{L}_{lh}(y, z, x) + \mathcal{L}_R(y, z, x) + \mathcal{L}_{adv}(y, z, x)]$

$$c^2 D_t \dot{\gamma}(ct) = 0. \tag{5.37}$$

Hence, $\tilde{\gamma}$ is a geodesic, and therefore, $\tilde{\gamma} = \gamma_{cV}$.

□

This is an important result, since it immediately follows that $\left\| \dot{z}_{t_0}^1 \right\| > \left\| \dot{z}_{t_0}^2 \right\| \implies L(\gamma_{\dot{z}^1}(T)) > L(\gamma_{\dot{z}^2}(T))$. Following these intuitions, we define $\dot{z} = \nabla_z \left\| y - G(z_{t_0}) \right\|$. This yields an interesting advantage, i.e., $\|\dot{z}\|$ (hence $L(\gamma_{\dot{z}}(T))$) tends to be large for high $\left\| y - G(z_{t_0}) \right\|$, which corresponds to **R3**.

5.4.2 Encouraging the local bijective conditions

The approach described in Sec. 5.4.1 encourages a global bi-lipschitz mapping between $\mathcal{M}_y$ and $\mathcal{M}_z$. However, we practically observed that imposing bijective conditions in local neighborhoods in conjunction leads to improved performance. Thus, we enforce a dense bijective mapping between $\mathcal{M}_y$ and $\mathcal{M}_z$ near $\gamma_{\mathcal{M}_y}(T)$. Let z_T and y be the latent code at $\gamma_{\mathcal{M}_y}(T)$ and the ground truth, respectively. We generate two random sets $\tilde{\mathcal{Z}}$ and $\tilde{\mathcal{Y}}$ using the distribution,

$$\tilde{\mathcal{Z}} = \mathcal{N}(z_T; \epsilon_2) \quad \text{and} \quad \tilde{\mathcal{Y}} = \Psi(y), \tag{5.38}$$

where $\Psi(\cdot)$ applies random perturbations such as brightness, contrast and small noise, and $0 < \epsilon_2 < 1$. One trivial method to ensure that a bijective mapping exists is to apply a loss function $\sum \left\| y_i - G(z_i) \right\|$, $\forall z_i \in \tilde{\mathcal{Z}}, y_i \in \tilde{\mathcal{Y}}$ to update the generator. However, we empirically observed that the above loss function unnecessarily applies a hard binding between the perturbations and the generated data. Therefore, we minimize the KL-distance between G and $\tilde{\mathcal{Y}}$ up to second order moments. One possible way to achieve this is to model each pixel as a univariate distribution (App 5.4.3). However in this case, since the generator cannot capture the correlations between different spatial locations, unwanted artifacts appear on the generated data. Therefore, we treat G and

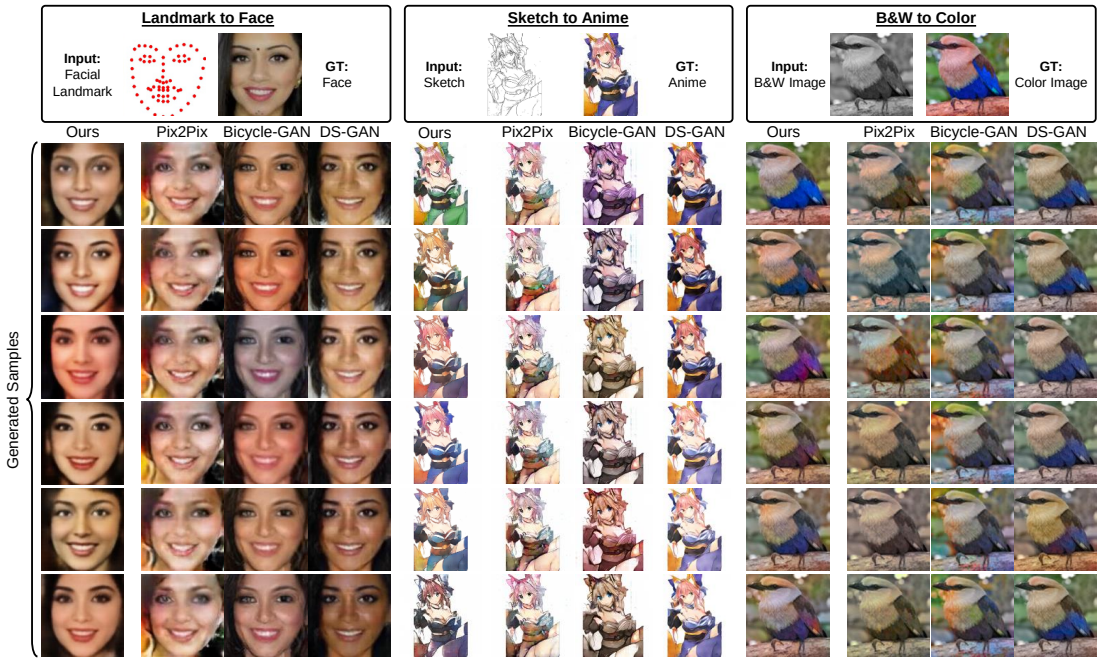


Figure 5.2: *Qualitative comparison with state-of-the-art cGANs on three challenging tasks.* We compare our proposed model with the baseline Pix2Pix [Isola et al., 2017], Bicycle-GAN [Zhu et al., 2017b] and DS-GAN [Yang et al., 2019a]. It can be seen that samples generated by our model are clearly more diverse (e.g., , color and subtle structural variation) and realistic (e.g., , shape and color) compared to other models in all tasks. Note that our model has the same architecture as Pix2Pix.

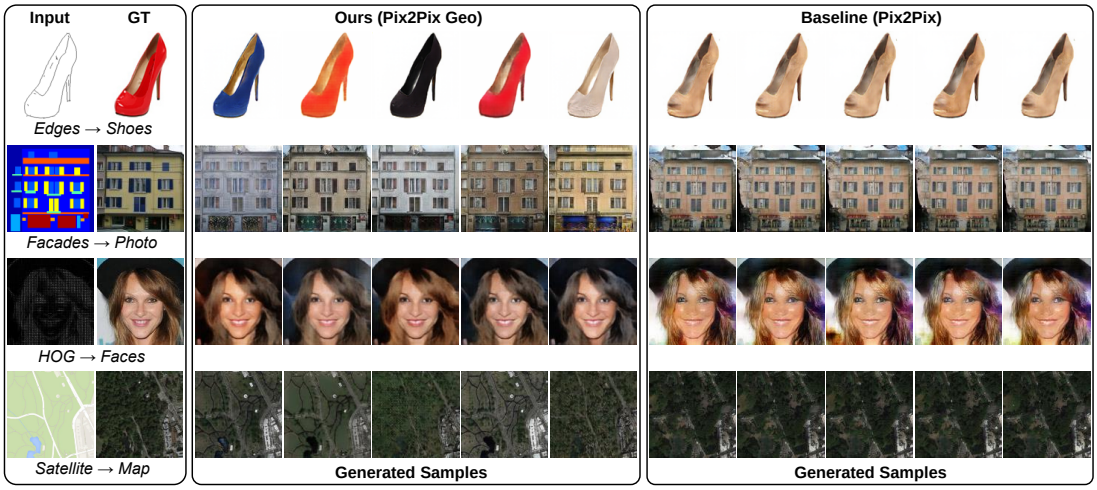


Figure 5.3: *Qualitative comparisons with baseline Pix2Pix [Isola et al., 2017] model.* Our proposed model consistently generates diverse and realistic samples compared to its baseline Pix2Pix model.

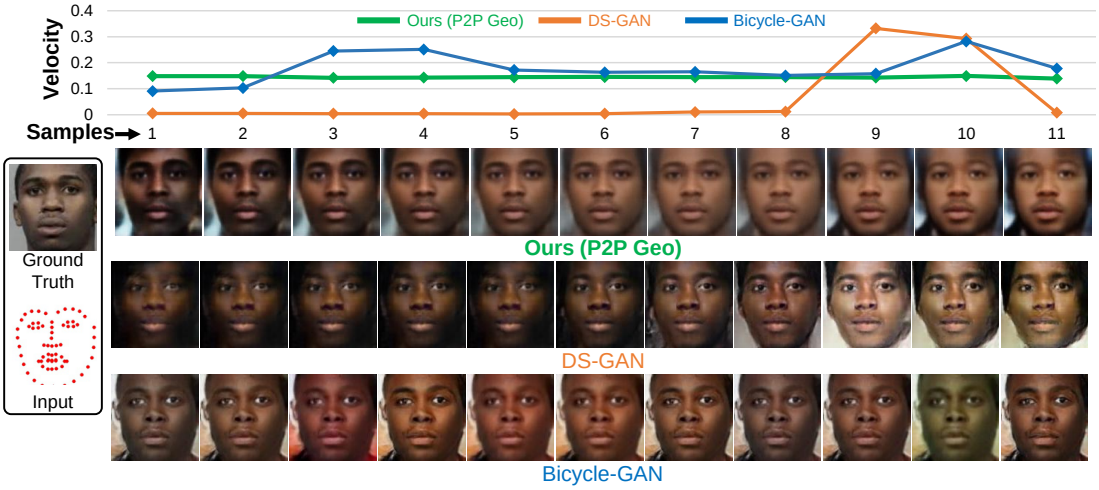


Figure 5.4: *A visual example of interpolation along an Euclidean shortest path on the latent manifold. Top row:* the velocity $V = \sqrt{\dot{z}\mathbf{M}\dot{z}}$ change on $\mathcal{M}_y$ across the samples. *Bottom three rows:* the corresponding interpolated samples in Bicycle-GAN, DS-GAN, and P2P Geo (Ours). As evident, our model exhibits a smooth interpolation along with an approximately constant velocity on $\mathcal{M}_y$ compared to the other networks, implying that our model indeed tends to move along geodesics. The total standard deviations of the V for 100 random interpolations for Bicycle-GAN, DS-GAN, and P2P Geo (Ours) are 0.056 0.067, and 0.011, respectively.

$\tilde{\mathcal{Y}}$ as M-dimensional multivariate distributions (M = image height $\times$ image width). Then, the KL-distance between the distributions up to the second order of moments can be calculated using the following equation,

$$\begin{aligned}\mathcal{L}_{lh}(y,z,x) =& \frac{1}{2}\Bigg[\log\frac{|\Sigma_{G^*}|}{|\Sigma_{\tilde{\mathcal{Y}}}|} - M + \mathrm{tr}(\Sigma_G^{-1}\Sigma_{\tilde{\mathcal{Y}}}) \\ &+ (\mu_G - \mu_{\tilde{\mathcal{Y}}})^T\Sigma_G^{-1}(\mu_G - \mu_{\tilde{\mathcal{Y}}})\Bigg],\end{aligned} \tag{5.39}$$

where Σ and μ denote the correlation matrices and the means.

5.4.3 Univariate distributions

Minimizing the information loss between two distributions can be interpreted as minimizing the Kullback–Leibler (KL) distance between the two distributions. KL-distance between two distribution is defined as,

$$KL(P||Q) = \int p(x)\log\left[\frac{p(x)}{q(x)}\right]dx. \tag{5.40}$$

If we approximate an arbitrary density Q in $\mathbb{R}^n$ with a Gaussian distribution, it can be shown that the parameters which minimize the KL-distance between Q and a given density P are exactly the same as minimizing the distance between P and Q up to the second moment. Therefore, we approximate P and Q with Gaussian distributions and minimize the KL distance between them.

Now, consider two Gaussian distributions, P and Q.

$$\begin{aligned}
KL(P||Q) &= \int \Big[\log(P(x)) - \log(Q(x))\Big] P(x)dx \\
&= \int \Bigg[-\frac{1}{2}\log(2\pi) - \log(\sigma_P) - \frac{1}{2}(\frac{x-\mu_P}{\sigma_P})^2 \\
&+ \frac{1}{2}\log(2\pi) + \log(\sigma_Q) + \frac{1}{2}(\frac{x-\mu_Q}{\sigma_Q})^2 \Bigg] \frac{1}{\sqrt{2\pi\sigma_P}} \exp\Bigg[-\frac{1}{2}(\frac{x-\mu_P}{\sigma_P})^2 \Bigg] dx \\
&= \int \Bigg[\log(\frac{\sigma_Q}{\sigma_P}) + \frac{1}{2}((\frac{x-\mu_Q}{\sigma_Q})^2 - (\frac{x-\mu_P}{\sigma_P})^2) \Bigg] \frac{1}{\sqrt{2\pi\sigma_P}} \exp\Bigg[-\frac{1}{2}(\frac{x-\mu_P}{\sigma_P})^2 \Bigg] dx \\
&= \mathop{\mathbb{E}}_{P}\Bigg[\log(\frac{\sigma_Q}{\sigma_P}) + \frac{1}{2}((\frac{x-\mu_Q}{\sigma_Q})^2 - (\frac{x-\mu_P}{\sigma_P})^2) \Bigg] \\
&= \log(\frac{\sigma_Q}{\sigma_P}) + \frac{1}{2\sigma_Q^2}\mathop{\mathbb{E}}_{P}[(x-\mu_Q)^2] - \frac{1}{2} \\
&= \log(\frac{\sigma_Q}{\sigma_P}) + \frac{\sigma_P^2 + (\mu_P - \mu_Q)^2}{2\sigma_Q^2} - \frac{1}{2}.
\end{aligned} \tag{5.41}$$

5.4.4 Multivariate distribution

Consider two Gaussian distributions, P and Q in $\mathbb{R}^n$,

$$P(x) = \frac{1}{(2\pi)^{n/2}\det(\Sigma_P)^{1/2}} \exp\Bigg[-\frac{1}{2}(x-\mu_P)^T\Sigma_P^{-1}(x-\mu_P) \Bigg], \tag{5.42}$$

$$Q(x) = \frac{1}{(2\pi)^{n/2}\det(\Sigma_Q)^{1/2}} \exp\Bigg[-\frac{1}{2}(x-\mu_Q)^T\Sigma_Q^{-1}(x-\mu_Q) \Bigg]. \tag{5.43}$$

KL distance between the two distributions,

$$
\begin{aligned}
KL(P||Q) &= \mathop{\mathbb{E}}_{P}\Big[\log P - \log Q\Big] \\
&= \frac{1}{2}\mathop{\mathbb{E}}_{P}\Big[- \log\det\Sigma_P - (x-\mu_P)^T\Sigma_P^{-1}(x-\mu_P) + \log\det\Sigma_Q + (x-\mu_Q)^T\Sigma_Q^{-1}(x-\mu_Q)\Big] \\
&= \frac{1}{2}\Big[\log\frac{\det\Sigma_Q}{\det\Sigma_P}\Big] + \frac{1}{2}\mathop{\mathbb{E}}_{P}\Big[- (x-\mu_P)^T\Sigma_P^{-1}(x-\mu_P) + (x-\mu_Q)^T\Sigma_Q^{-1}(x-\mu_Q)\Big] \\
&= \frac{1}{2}\Big[\log\frac{\det\Sigma_Q}{\det\Sigma_P}\Big] + \frac{1}{2}\mathop{\mathbb{E}}_{P}\Big[- \mathrm{tr}((\Sigma_P^{-1}(x-\mu_P)(x-\mu_P)^T) + \mathrm{tr}(\Sigma_Q^{-1}(x-\mu_Q)(x-\mu_Q)^T)\Big] \\
&= \frac{1}{2}\Big[\log\frac{\det\Sigma_Q}{\det\Sigma_P}\Big] + \frac{1}{2}\mathop{\mathbb{E}}_{P}\Big[- \mathrm{tr}(\Sigma_P^{-1}\Sigma_P) + \mathrm{tr}(\Sigma_Q^{-1}(xx^T - 2x\mu_Q^T + \mu_Q\mu_Q^T))\Big] \\
&= \frac{1}{2}\Big[\log\frac{\det\Sigma_Q}{\det\Sigma_P}\Big] - \frac{1}{2}M + \frac{1}{2}\mathrm{tr}(\Sigma_Q^{-1}(\Sigma_P + \mu_P\mu_P^T - 2\mu_Q\mu_P^T + \mu_Q\mu_Q^T)) \\
&= \frac{1}{2}(\Big[\log\frac{\det\Sigma_Q}{\det\Sigma_P}\Big] - M + \mathrm{tr}(\Sigma_Q^{-1}\Sigma_P) + \mathrm{tr}(\mu_P^T\Sigma_Q^{-1}\mu_P - 2\mu_P^T\Sigma_Q^{-1}\mu_Q + \mu_Q\Sigma_Q^{-1}\mu_Q)) \\
&= \frac{1}{2}(\Big[\log\frac{\det\Sigma_Q}{\det\Sigma_P}\Big] - M + \mathrm{tr}(\Sigma_Q^{-1}\Sigma_P) + (\mu_Q-\mu_P)^T\Sigma_Q^{-1}(\mu_Q-\mu_P)).
\end{aligned}
\tag{5.44}
$$

However, using the above loss (Eq. 5.39) in its original form yields practical obstacles: for instance, the correlation matrices have the dimension $M \times M$, which is infeasible to handle. Therefore, following Achlioptas [2001], we use a random projection matrix $R^{M\times h}; hM$ to project the images to a $h-$dimensional space, where $R_{i,j} \sim p(x)$; $p(\sqrt{3}) = \frac{1}{6}, p(0) = \frac{2}{3}, p(-\sqrt{3}) = \frac{1}{6}$ (we empirically justify this reduction method using an ablation study in Sec. 5.5.4). Moreover, numerically calculating $|\Sigma|$ and Σ^{-1} causes unstable gradients which hinders the generator optimization. We address this issue by adapting the approximation technique proposed in Boutsidis et al. [2017]:

$$
\log(|\Sigma|) \approx -\sum_{i=1}^{N}\frac{\mathrm{tr}(C^i)}{i}, \tag{5.45}
$$

where $C = \mathbf{I} - \Sigma$. Further, Σ^{-1} can be calculated as,

$$
V_{i+1} = V_i(3\mathbf{I} - \Sigma V_i(3\mathbf{I} - \Sigma V_n)), i = 1, 2, \ldots, N, \tag{5.46}
$$

where Li et al. [2011] proved that $V_i \to \Sigma^{-1}$ as $i \to \infty$, for a suitable approximation of V_0. They further showed that a suitable approximation should be $V_0 = \alpha\Sigma^T$, $0 < \alpha < 2/\rho(\Sigma\Sigma^T)$, where $\rho(\cdot)$ is the spectral radius. Our final loss function $\mathcal{L}_{total}$

consists of four loss components:

$$\mathcal{L}_{total} = \beta_0 \mathcal{L}_{gh} + \beta_1 \mathcal{L}_{lh} + \beta_2 \mathcal{L}_r + \beta_3 \mathcal{L}_{adv}, \tag{5.47}$$

where $\beta_0 \dots \beta_3$ are constant weights learned via cross-validation. Algorithm 1 shows overall training scheme.

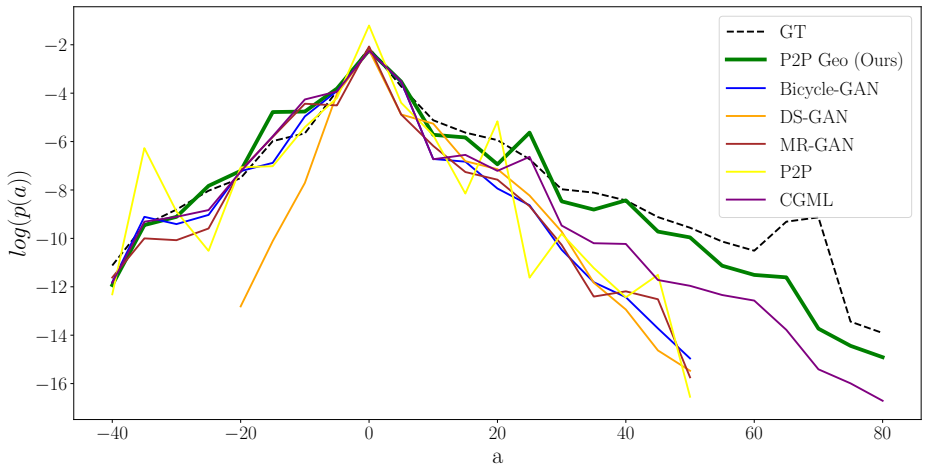


Figure 5.5: Colour distribution comparison on $BW \rightarrow$ color dataset on a-plane in Lab color space. Our model exhibits the closest color distribution compared to the ground truth. Furthermore, our model is able to generate rare colors which implies more diverse colorization.

5.5 Experiments

In this section, we demonstrate the effectiveness of the proposed training scheme using qualitative and quantitative experiments. First, we illustrate the generalizability of our method by comparing against the state-of-the-art methods across a diverse set of image-to-image translation tasks. Then, we explore the practical implications of geometrically structuring the latent manifold. Finally, we conduct an ablation study to compare the effects of the empirical choices we made in Sec. 8.1. In all the experiments, we use Pix2Pix [Isola et al., 2017] as our model architecture, and use the same model trained using the traditional training approach as the main baseline. We use the official implementation of other comparable methods to benchmark their performance against ours. For a fair comparison, we use their pre-trained models wherever available, otherwise train their model from scratch, strictly following the authors' instructions to the best of our ability.

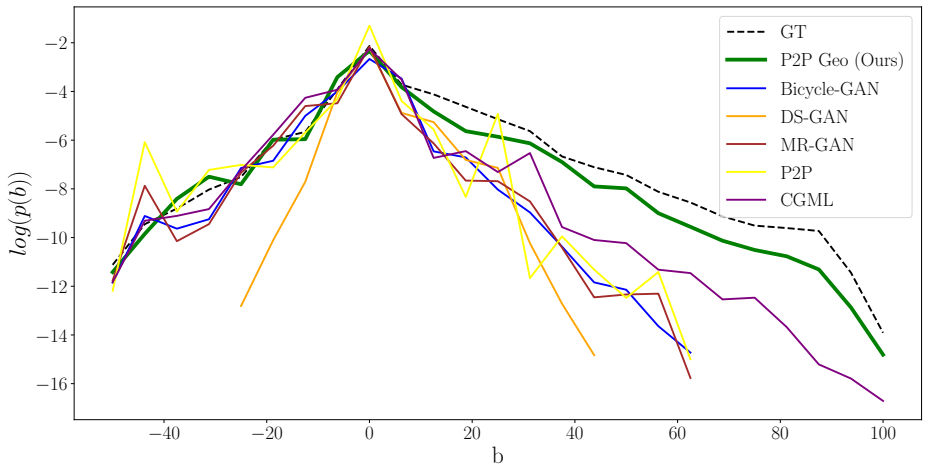


Figure 5.6: Colour distribution comparison on $BW \rightarrow$ color dataset on b-plane in Lab color space. Our model exhibits the closest color distribution compared to the ground truth. Furthermore, our model is able to generate rare colors which implies more diverse colorization.

5.5.1 Hyper-parameters and datasets

We use 100 iterations with $\alpha = 0.1$ to calculate the inverse of matrices using Eq. 5.46 and 20 iterations to calculate the log determinant using Eq. 5.45. Further, 10 time steps are used for $\mathcal{L}_{gh}$, and z_{t_0} is sampled from a $\mathcal{B}^{64}_{0.01}$. For training, we use the Adam optimizer with hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$. All the weights are initialized using a random normal distribution with 0 mean and 0.5 standard deviation. The weights of the final loss function are,

$$\mathcal{L}_{total} = 100.0\mathcal{L}_{gh} + 0.01\mathcal{L}_{lh} + 100.0\mathcal{L}_{R} + \mathcal{L}_{adv}, \tag{5.48}$$

All these values are chosen empirically. For $facades \rightarrow photo$, $map \rightarrow photo$, $edges \rightarrow shoes$, $edges \rightarrow bags$, and $night \rightarrow day$, we use the same stadard datasets used in Pix2Pix [Isola et al., 2017]. For the $landmarks \rightarrow faces$, $hog \rightarrow faces$, $BW \rightarrow color$, and $sketch \rightarrow anime$ experiments, we use the UTKFace dataset [Zhang and Qi, 2017], CelebHQ dataset [Lee et al., 2020], STL dataset [Coates et al., 2011], and *Anime Sketch Colorization Pair* dataset [Kim] provided in Kaggle, respectively.

5.5.2 Image-to-image translation

We compare our method against state-of-the-art models that focus on multimodal image-to-image translation. Fig. 5.2 shows the qualitative results on $landmarks \rightarrow faces$, $sketch \rightarrow anime$ and $BW \rightarrow color$. As evident, our training mechanism increases the diversity and the visual quality of the baseline P2P model significantly, and shows

better performance compared to other models. Fig. 5.3 shows qualitative comparison against the baseline. Table 8.9 depicts the quantitative results. As shown, our model exhibits a higher diversity and a higher realism on multiple datasets. In all the cases, we outperform our baseline by a significant margin. Fig. 5.5 and Fig. 5.6 compare color distribution in *BW2color* task.

5.5.3 Geometrical interpretations

A key implication of our training scheme is that the Euclidean shortest paths on $\mathcal{M}_z$ map to geodesics on $\mathcal{M}_y$, which preserves better structure. We conduct an experiment to empirically validate the aforementioned attribute. First, we travel along Euclidean paths on $\mathcal{M}_z$ and measure the corresponding curve length L_E on the data manifold. Second, we calculate the actual geodesic distance L_G between the same two points on $\mathcal{M}_y$ using Eq. 5.19 in discrete intervals. We travel in 10 random directions starting from random initial points, and obtain L_{G_i} for evenly spaced $L_E \in \{10, 20, 30, \dots 90\}$. Then, we obtain set of the means and standard deviations of L_G for the corresponding L_E. Fig. 5.7 illustrates the distribution. As evident, our model exhibits a significantly high overlap with the ideal curve, i.e., , $L_E = \mathbb{E}(L_G)$ compared to DS-GAN and Bicycle-GAN.

A useful attribute of travelling along the geodesics on the output manifold ($\mathcal{M}_y$) is to obtain smooth interpolations, since the geodesics tend to avoid regions with high distortions, i.e., rapid changes. However, Euclidean shortest paths in the latent spaces ($\mathcal{M}_z$) of cGANs often do not correspond to geodesics on the $\mathcal{M}_y$. Therefore, in order to travel along geodesics, it is required to numerically obtain the geodesic paths using Eq. 5.19, which requires extra computation. In contrast, the proposed training method encourages the generator to map the Euclidean paths on $\mathcal{M}_z$ to geodesics on $\mathcal{M}_y$. Therefore, smooth interpolations can be obtained by traveling between two latent codes in a straight path. To evaluate this, we compare the interpolation results between Bicycle-GAN, DS-GAN and our model. Fig. 5.7 shows a qualitative example, along with a quantitative evaluation. As visible, our model exhibits smooth transition from the starting point to the end point. In comparison, Bicycle-GAN shows abrupt and inconsistent changes along the path. DS-GAN does not show any significant variance in the beginning and shows sudden large changes towards the end. We also quantify this comparison using the velocity on the data manifold: since the curve length on $\mathcal{M}_y$ can be calculated using Eq. 5.22, it is easy to see that the velocity on $\mathcal{M}_y$ can be obtained using $\sqrt{\dot{z}_i^T \mathbf{M} \dot{z}_i}$. Fig. 5.7 further illustrates the change in the aforementioned velocity, corresponding to the given qualitative examples. Our model demonstrates an approximately constant non-zero velocity (note that geodesics have constant velocities), while the other models show sudden velocity changes along the path. Further qualitative results for smooth interpolation are shown in the Fig. 5.16. We did not include CGML in these evaluations. The reason is that the inference procedure of the CGML is fundamentally different from a CGAN. The latent variables are randomly initialized at inference and then guided towards optimal latent codes through a separate path-finding expert module. As a result, unlike CGANs, the entire

latent space is not considered as a low-dimensional manifold approximation of the output space. In other words, interpolation through sub-optimal areas of the latent space does not correspond to meaningful changes in the output. Therefore, we did not use CGML for experiments that evaluate the structure of the latent space.

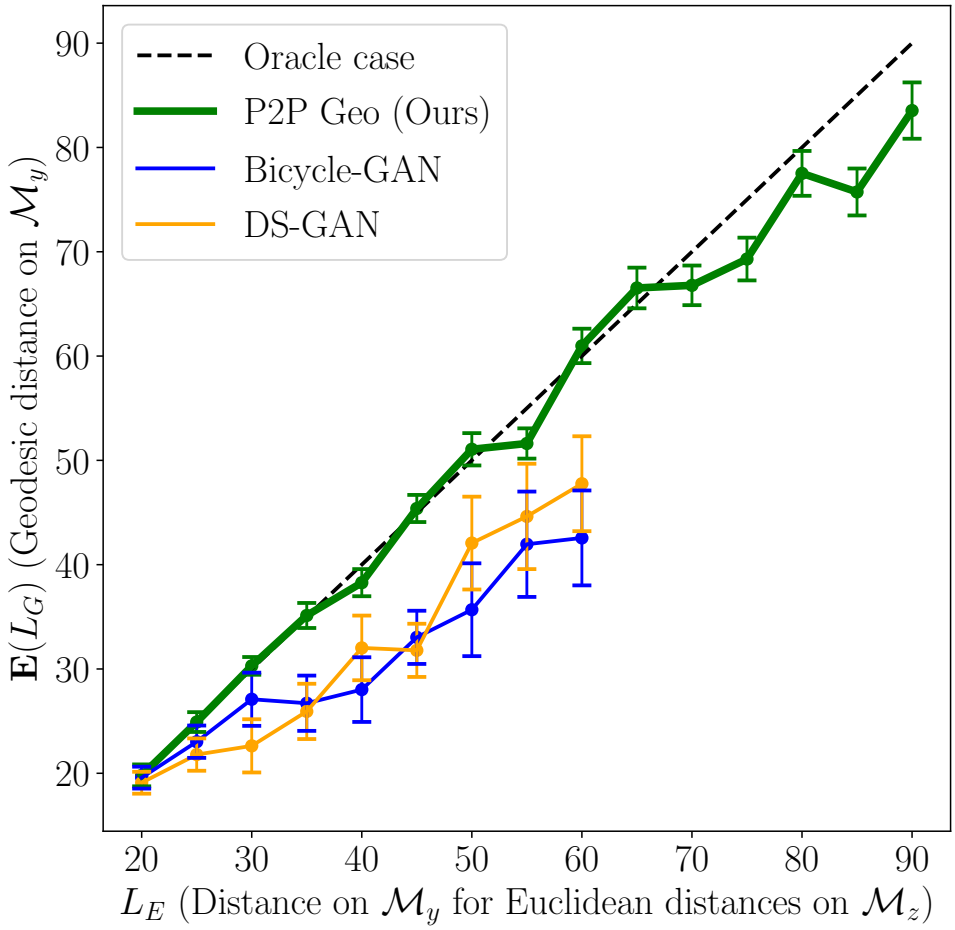


Figure 5.7: *Euclidean path vs. geodesic comparison.* We travel along a Euclidean shortest path on $\mathcal{M}_z$ and measure the corresponding curve distance L_G on $\mathcal{M}_z$ (*lm2faces*). Then, we traverse between the same two points along the numerically calculated geodesic and measure the corresponding curve length L_G. $\mathbb{E}(L_G)$ vs L_E is illustrated with the corresponding standard deviation obtained along 10 random paths. Our model is closer to the oracle case ($L_E = \mathbb{E}(L_G)$). We were not able to obtain distance greater than ~ 60 for DS-GAN and Bicyle-GAN which implies that our model generates more diverse data. Further, Pix2Pix did not produce enough diversity for this comparison.

5.5.4 Ablation study

We conduct an ablation study to compare the different variants of the proposed technique. Table 5.2 depicts the results. First, we compare different distance functions used to calculate $\mathcal{L}_{lh}$. As expected, naive maximization of the distances between the generated samples without any constraints increases the diversity, but reduces the visual quality drastically. Further, we observed unwanted artifacts when modeling each pixel as a univariate distribution, as the model then cannot capture dependencies across spatial locations. Then, we compare different down-sampling methods that can be used for efficient calculation of the correlation matrices, where random projection performed the best. Interestingly, we observed a reduction of the visual quality when the dimension of the latent code is increased. In contrast, the diversity tends to improve with the latter. We chose $\dim(z) = 64$ as a compromise. Finally, we compare the effects of different combinations of the loss components.

5.5.5 Generalizability

To demonstrate the generalizability of the proposed algorithm across different loss functions and architectures, we employ it on three classic networks: Pathak et al. [2016b], Johnson et al. [2016a], and Ronneberger et al. [2015]. These networks use a masked reconstruction loss with the adversarial loss, perception loss from pre-trained networks, and a reconstruction loss, respectively.

Pathak *et al.* This model is used for image inpainting tasks, and is trained by regressing to the ground truth content of the missing area. To this end, the authors utilize a reconstruction loss (L_{rec}) and an adversarial loss (L_{Adv}). Consider a binary mask M where missing pixels are indicated by 1 and 0 otherwise. Then, $L_{rec}(x) = \left\| M \odot (x - G((1 - M) \odot x)) \right\|_2^2$, where x is the input and $\odot$ is the element-wise production. L_{adv} is the usual adversarial loss on the entire output. In order to apply our training algorithm, we replace L_R with L_{rec}.

Johnson *et al.* The primary purpose of this network is neural style transferring, *i.e.*, given a artistic style image and an RGB image, output should construct an image where the content of the RGB image is represented using the corresponding artistic style. The model utilizes an encoder decoder mechanism and consists of four loss components: 1) feature reconstruction loss $\mathcal{L}_{fr}$, 2) style reconstruction loss $\mathcal{L}_{style}$ 3) reconstruction loss and 4) variation regularization loss $\mathcal{L}_{tv}$. The feature reconstruction loss is obtained by passing the generated and ground truth images through a pre-trained VGG-16 and calculating the ℓ_2 loss between the corresponding feature maps. Let the output of the *relu2_2* layer of VGG-16 be denoted as $\phi(\cdot)$. Then,

$$\mathcal{L}_{fr}(y, \bar{y}) = \frac{1}{K} \left\| \phi(y) - \phi(\bar{y}) \right\|_2^2, \tag{5.49}$$

where K is the number of neurons in *relu2_2*.

The style reconstruction loss is similar, except that the inputs to the VGG-16 are the generated image and the style image. Let the output of the j^{th} layer of VGG-16

be $\phi(\cdot)_j$. Further, assume that $\phi(\cdot)_j$ gives C_j dimensional features on a $H_j \times W_j$ grid, which can be reshaped in to a $C_j \times H_j W_j$ matrix ψ_j. Then, $G_j(\cdot) = \psi\psi^T/(C_j H_j W_j)$ and,

$$\mathcal{L}_{style} = \left\| G_j(y) - G_j(\bar{y}) \right\|_F^2, \tag{5.50}$$

where $\|\cdot\|_F$ is the Frobeneus norm. While training, $\mathcal{L}_{style}$ is calculated for *relu1_2*, *relu2_2*, *relu3_3*, and *relu4_3* of the VGG-16.

Reconstruction loss is simply the pixel-wise ℓ_2 loss. They also adapt a total variation loss to encourage spatial smoothness in the output image as,

$$\mathcal{L}_{tv}(\bar{y}) = \sum_{i,j}((\bar{y}_{i,j+1}, -\bar{y}_{i,j})^2 + (\bar{y}_{i+1,j}, -\bar{y}_{i,j})^2). \tag{5.51}$$

In order to apply our training algorithm, we replace L_{Adv} with $\mathcal{L}_{fr}$, $\mathcal{L}_{style}$, and $\mathcal{L}_{tv}$.

Ronneberger *et al.* This model was originally proposed for segmentation of RGB (medical) images and is trained with a soft-max cross-entropy loss between the predicted and target classes. However, we use a pixel-wise reconstruction loss as the objective function to allow multi-modal outputs. Further, we define the task at hand as converting segmentation maps to faces. To impose our training algorithm, we simply remove L_{adv}.

The above networks are designed to capture one-to-one mappings between the inputs and the outputs. Therefore, the only stochasticity in these models is the dropout. Therefore, we concatenate a latent map to the bottle-necks of the networks to improve the stochasticity. Note that simply concatenating the latent maps without our algorithm does not yield diverse outputs as the naive reconstruction losses (which exist in all of the above networks) only converge to a single output mode. Our algorithm increases the diversity of the models and obtain one-to-many mappings with no changes to the architecture (for fair comparison, we concatenate a latent code at the bottlenecks during both the original and proposed training).

5.5.6 Qualitative results

We apply our model on a diverse set of image-to-image translation tasks. Qualitative results are shown in Fig. 5.12, Fig. 5.11, Fig. 5.15, Fig. 5.9, Fig. 5.13, Fig. 5.14, and Fig. 5.10.

5.6 Chapter summary

In this chapter, we improve the performance of cGANSs by inducing inductive bias in the form of geometrical priors. We study vanilla cGANs and provide important insight into the relationship between the training mechanism and the manifold structure of

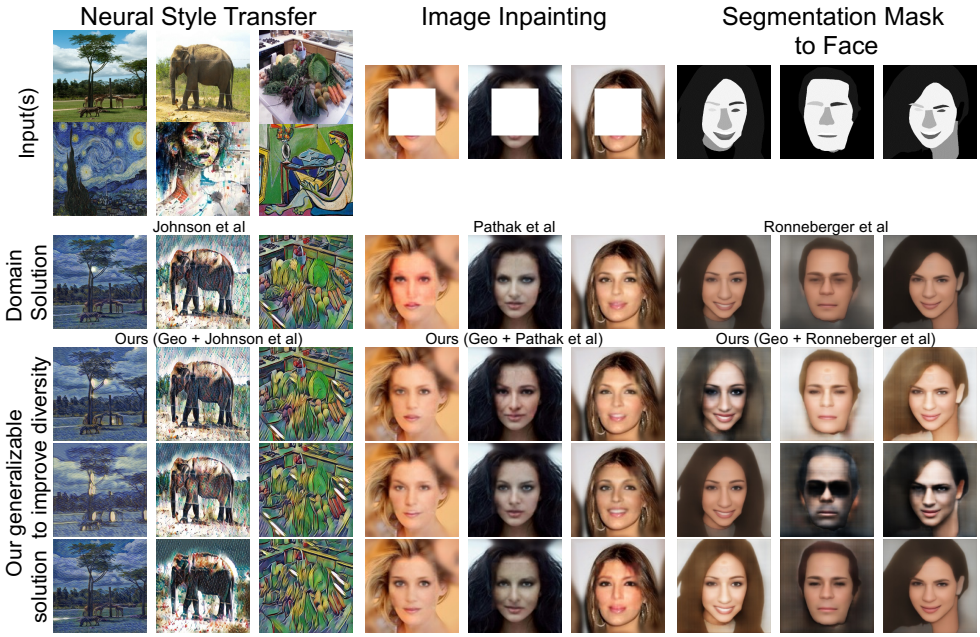


Figure 5.8: We apply our algorithm to three classic networks and obtain increased diversity with no architectural modifications. Note that the original networks only learn one-to-one mappings.

the latent and output spaces. We show that the cGANs, in their basic form, suffer from significant drawbacks in terms of diversity, and the proposed solutions to overcome these problems also are prone to undesired implications. Specifically, we show that the existing solutions are not optimal from a geometrical perspective, and can lead to sub-optimally structured latent space structures. In contrast, we propose a novel training algorithm that can increase both realism and the diversity of the outputs that are generated by cGANs while preserving the structure of the latent manifold. To this end, we enforce a bi-lipschitz mapping between the latent and generated output manifolds while encouraging Euclidean shortest paths on the latent manifold to be mapped to the geodesics on the generated manifold. Moreover, we establish the necessary theoretical foundation and demonstrate the effectiveness of the proposed algorithm at a practical level, using a diverse set of image-to-image translation tasks, where our model achieves compelling results.

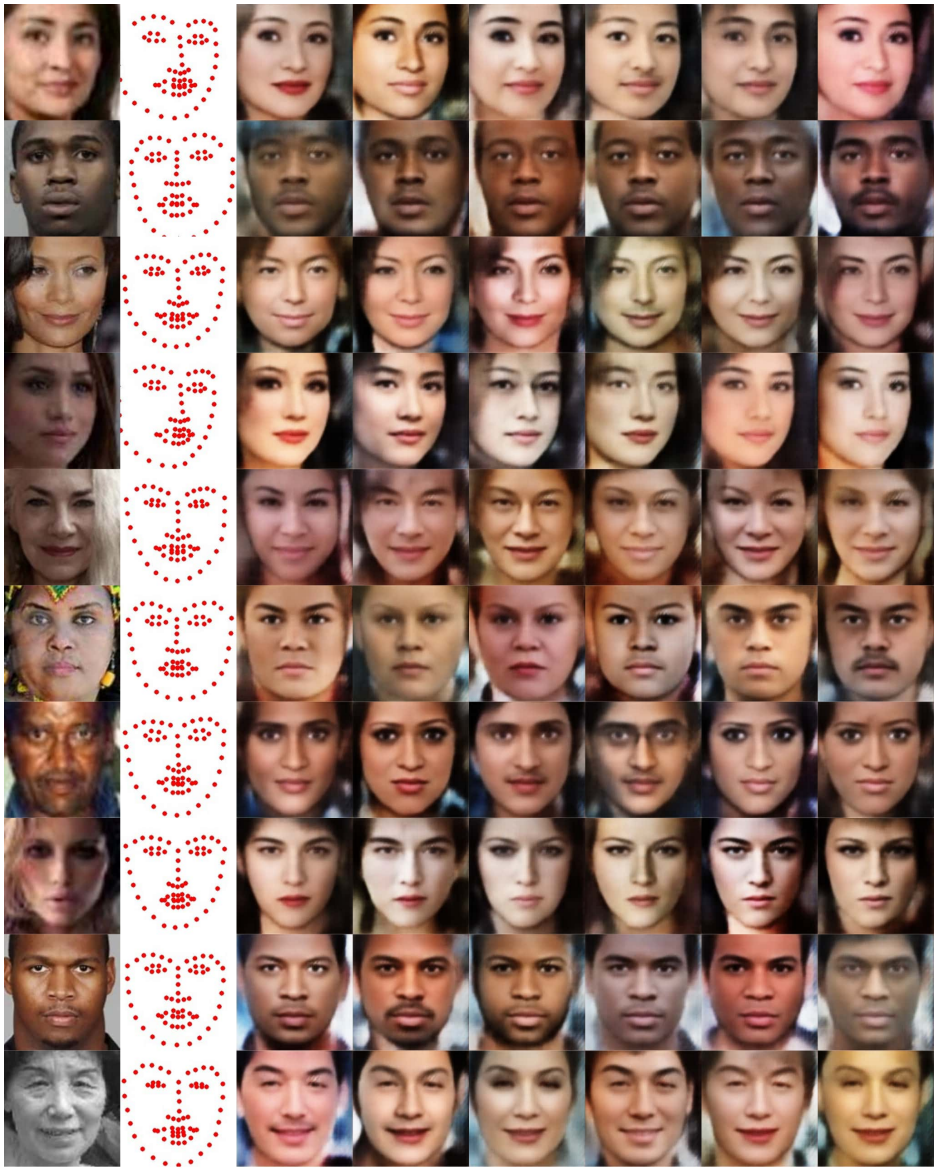


Figure 5.9: Qualitative results from *landmarks* → *faces* task.

Figure 5.10: Qualitative results from *sketch* → *shoes* task.

Figure 5.11: Qualitative results from *hog* → *faces* task. The diversity of the outputs are less in this task, as hog features maps are rich in information.

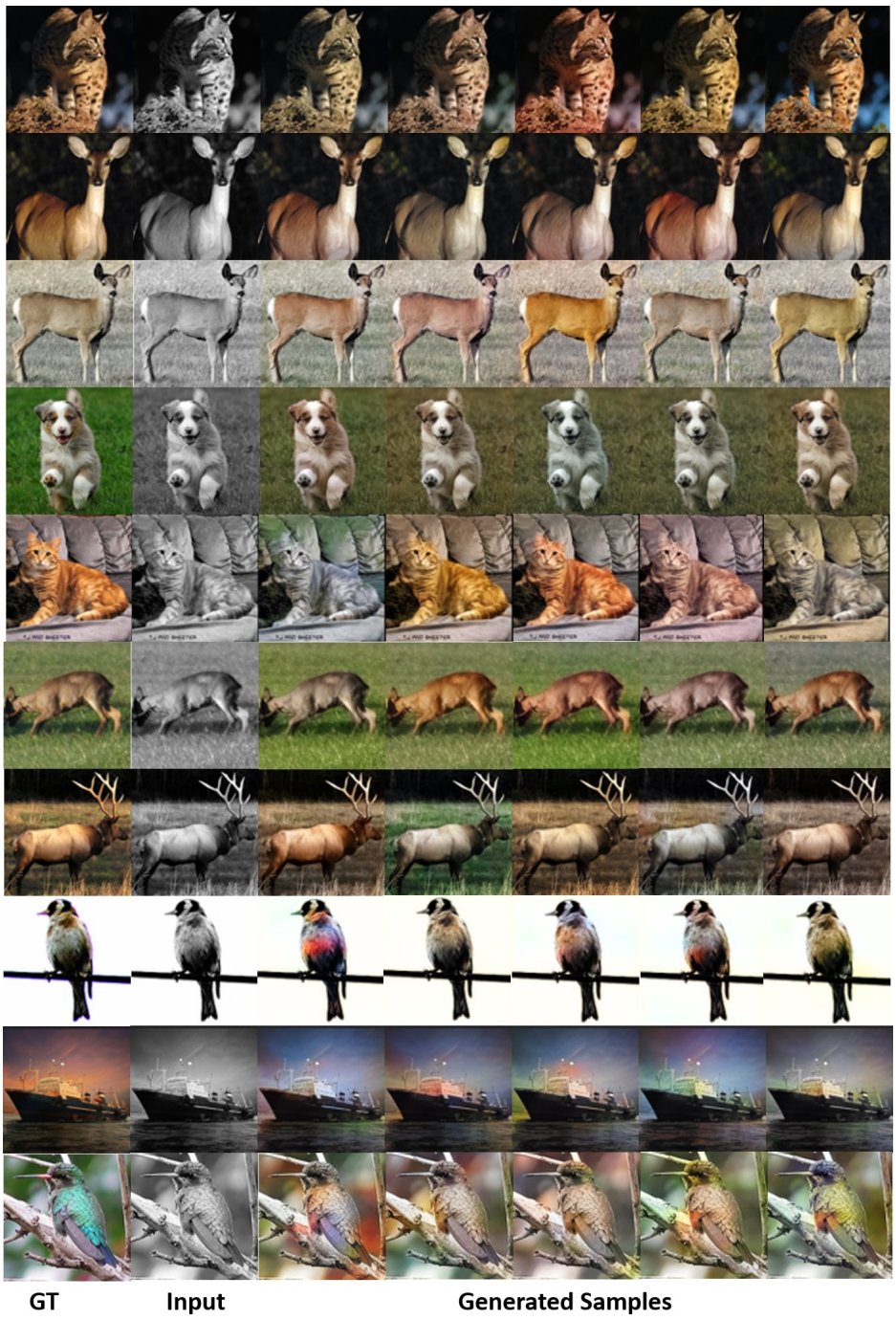


Figure 5.12: Qualitative results from *BW* $\rightarrow$ *color* task.

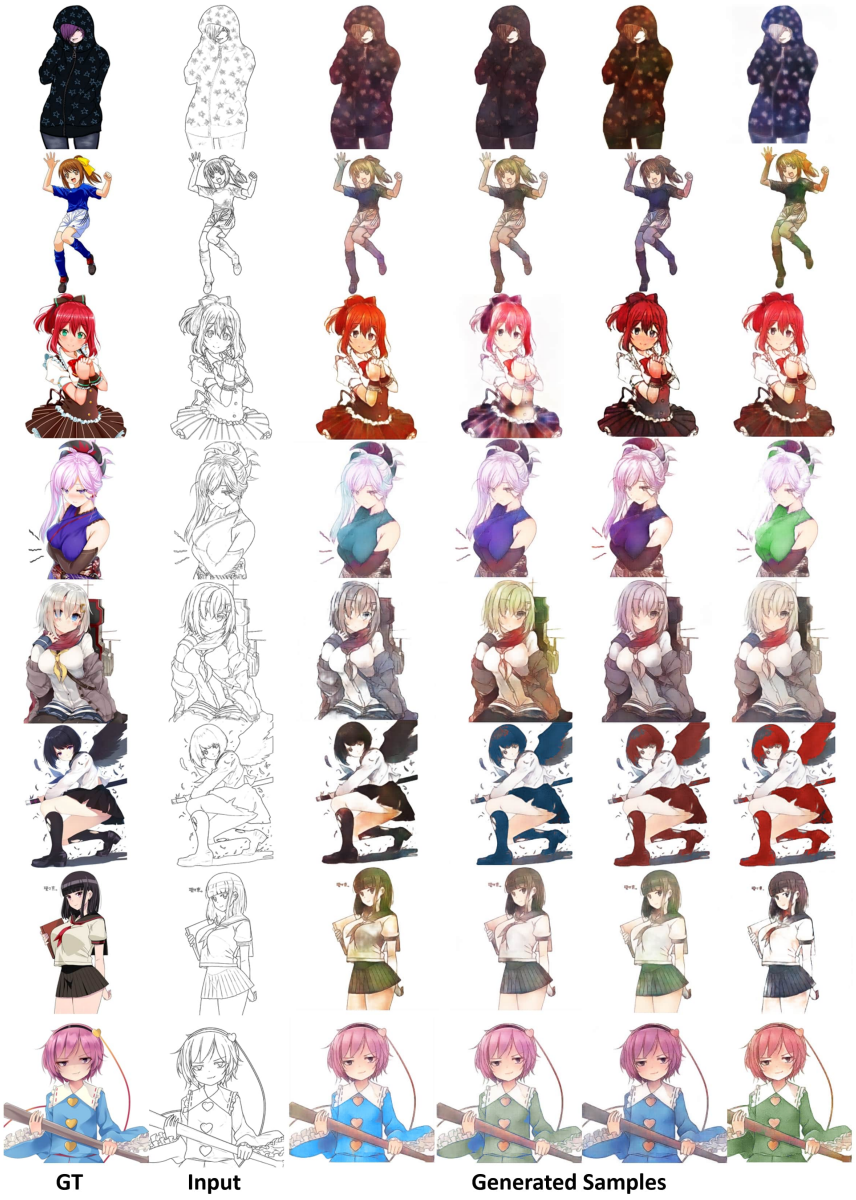


Figure 5.13: Qualitative results from *sketch* → *anime* task.

Figure 5.14: Qualitative results from *sketch* → *bags* task.

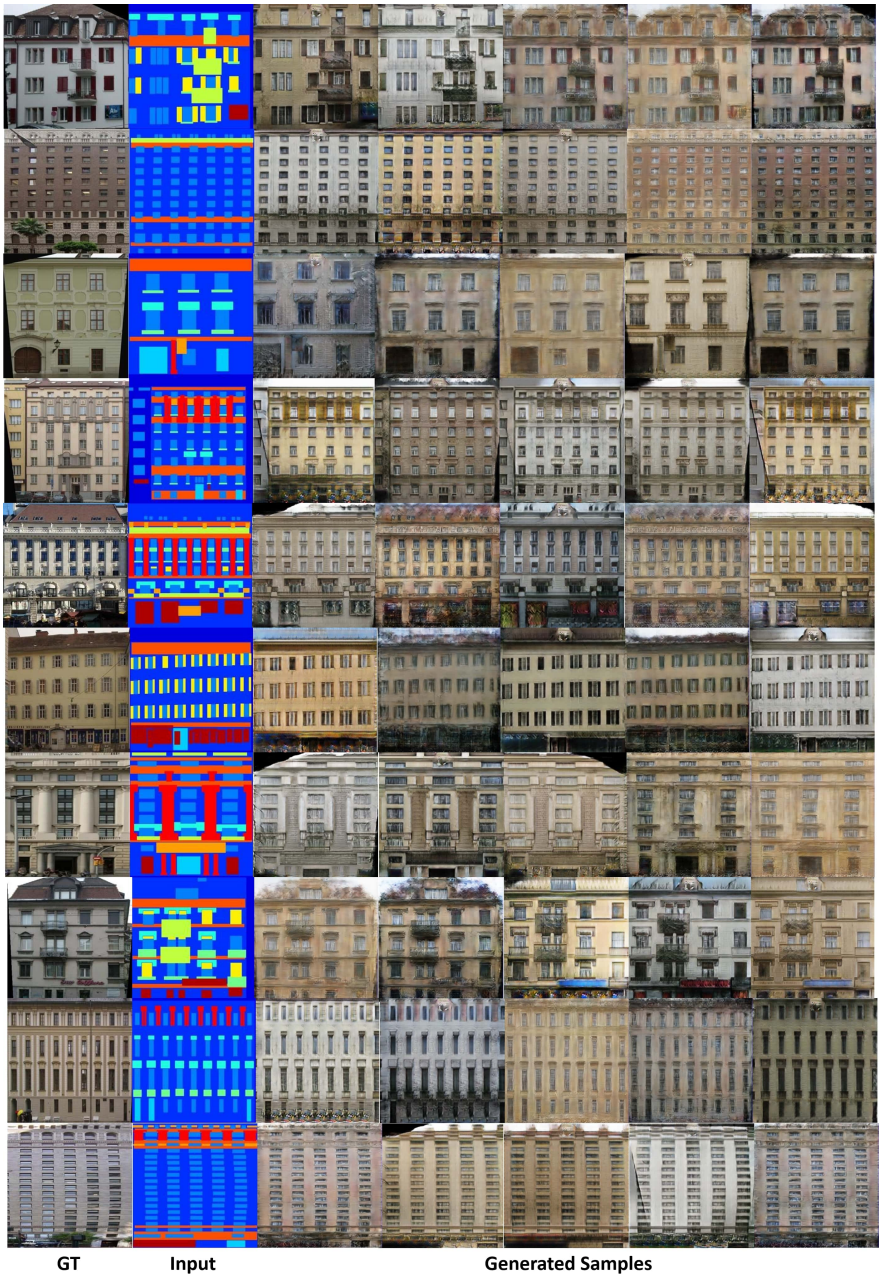


Figure 5.15: Qualitative results from *labels* → *facades* task.

Figure 5.16: Smooth interpolations of our model. Each column represents an interpolation between two latent codes, conditioned on an input. The faces are rotated to fit the space.

Method	*facades2photo*		*sat2map*		*edges2shoes*		*edges2bags*		*sketch2anime*		*BW2color*		*lm2faces*		*hog2faces*		*night2day*	
	LPIP	FID	LPIP	FID	LPIP	FID	LPIP	FID	LPIP	FID	LPIP	FID	LPIP	FID	LPIP	FID	LPIP	FID
Bicycle-GAN	0.142	**58.21**	0.109	54.21	0.139	21.49	**0.184**	22.33	0.026	73.33	0.008	78.13	0.125	72.93	0.065	98.208	**0.103**	**120.63**
DS-GAN	**0.181**	59.43	0.128	**48.13**	0.126	27.44	0.113	26.66	0.006	67.41	0.012	71.56	0.168	88.31	0.061	92.14	0.101	137.9
MR-GAN	0.108	110.31	0.091	108.34	-*	-*	-*	-*	-*	-*	0.015	113.46	0.182	108.72	0.138	155.31	0.098	140.51
CGML	**0.191**	**46.2**	0.143	**42.11**	0.13	**20.38**	**0.19**	20.43	0.05	61.40	0.092	**51.4**	0.190	73.40	0.141	51.33	0.100	127.8
Baseline (P2P)	0.011	92.06	0.014	88.33	0.016	34.50	0.012	32.11	0.001	93.47	0.002	97.14	0.009	121.69	0.021	151.4	0.008	157.3
Ours(P2P Geo)	0.148	63.27	**0.154**	59.41	**0.141**	**20.48**	0.167	**19.31**	**0.086**	**56.11**	**0.092**	**61.33**	**0.197**	**67.82**	**0.156**	**45.31**	0.101	131.8

Table 5.1: *Quantitative comparison with the state-of-the-art on 9 (nine) challenging datasets.* -* denotes the cases where we were not able to make the models converge. A higher LPIP similarity score means more diversity and lower FID score signifies more realism in the generated samples. Our approach gives consistent improvements over the baseline.

Variant type	Model	FID	LPIP
$\mathcal{L}_{lh}$	MMD	66.31	0.188
	2^{nd} moment (univaritate)	117.53	0.201
	Maximizing distance	132.91	**0.232**
	2^{nd} moment (multivariate)	**67.82**	0.197
Downsampling	Mean pool	75.41	0.192
	Max pool	82.42	0.162
	CNN	77.93	0.191
	Random Projection	**67.82**	**0.197**
$\dim(z)$	16	**65.32**	0.172
	32	67.11	0.188
	64	67.82	**0.197**
	128	82.33	0.166
Training loss	$\mathcal{L}_l + \mathcal{L}_{adv}$	91.3	0.051
	$\mathcal{L}_{gh} + \mathcal{L}_l + \mathcal{L}_{adv}$	**63.11**	0.151
	$\mathcal{L}_{lh} + \mathcal{L}_l + \mathcal{L}_{adv}$	91.3	0.055
	$\mathcal{L}_{lh} + \mathcal{L}_{gh} + \mathcal{L}_l + \mathcal{L}_{adv}$	67.82	**0.197**

Table 5.2: *Ablation study.* Ablation study with different variants of our model on *landmark* $\rightarrow$ *faces* dataset reporting FID score (lower = more realistic) and LPIPS (higher = more diverse).

Chapter 6

Robust normalizing flows using Bernstein-type polynomials

6.1 Introduction

In the last chapter, we showed that regularizing generative models using prior physics about the task in hand, can improve their performance. In particular, we used a GAN as a use case. In this chapter, we further validate this proposition by using a different class of generative models: Normalizing flows. Modeling the probability distribution of a set of observations, *i.e.*, generative modeling, is a crucial task in machine learning. It enables the generation of synthetic samples using the learned model and can estimate the likelihood of a data sample. This field has met with great success in many problem domains including image generation [Ho et al., 2019; Kingma and Dhariwal, 2018; Lu and Huang, 2020], audio synthesis [Esling et al., 2019; Prenger et al., 2019], reinforcement learning [Mazoure et al., 2020; Ward et al., 2019], noise modeling [Abdelhamed et al., 2019], and simulating physics experiments [Wirnsberger et al., 2020; Wong et al., 2020]. In the recent past, deep neural networks such as generative adversarial networks (GANs) and variational autoencoders (VAEs) have been widely adopted in generative modeling due to their tremendous success in modeling high dimensional distributions. However, they entail several limitations: 1) exact density estimation of arbitrary data points is not possible, and 2) training can be cumbersome due to aspects such as mode collapse, posterior collapse and high sensitivity to architectural design of the model Kobyzev et al. [2020].

In contrast, normalizing flows (NFs) are a category of generative models that enable exact density computation and efficient sampling. Since the seminal work by Rezende and Mohamed [2015], NFs have been gaining increasing attention from the machine learning community due to the attractive properties mentioned earlier. In the abstract, NFs consist of a series diffeomorphisms that transforms a simple distribution into a more complex one, which in turn allows an analytical density estimation of samples. In the implementation, the Jacobian determinants of these diffeomorphisms should be computed. As the computation of a determinant of an unconstrained $n \times n-$matrix is of $O(n^3)$ complexity, NFs must be designed so that efficient calculation of the Jacobian determinants is possible. To this end, two popular

approaches have been proposed so far: 1) efficient determinant calculation methods such as [Berg et al., 2018; Grathwohl et al., 2018; Lu and Huang, 2020], and 2) *triangular maps* [Jaini et al., 2019; Dinh et al., 2014, 2016]. Here, we focus on the latter.

The key benefit of triangular maps is that their Jacobian matrices are triangular, and hence, the calculation of Jacobian determinants takes only $O(n)$ steps. However, it is not a priori clear whether such a constrained class of maps is expressive enough to produce sufficiently complex transformations. Interestingly, Bogachev et al. [2005] shows that there exists a unique increasing triangular map that transforms one probability density to another (up to null sets). There have been numerous attempts at calculating such maps efficiently. In fact, Jaini et al. [2019] mentions that majority of the existing normalizing flow variants are triangular maps (*e.g.*, autoregressive flows), albeit not being universal, *i.e.*, not being dense in the space of increasing triangular maps. As a result, most of these methods have reverted to the empirical approach of stacking several transformations together, aiming to increase the expressiveness of the composite transformation. Alternatively, there are NFs that use genuinely universal transformations. The majority of such methods employ coupling functions based on polynomials such as sum-of-squares (SOS) polynomials in [Jaini et al., 2019], cubic splines in Durkan et al. [2019a] or rational quadratic splines in Durkan et al. [2019b]. Thanks to known mathematical properties of polynomials being used as building blocks, these models are highly interpretable.

In this Chapter, building on the existing literature, we further investigate this direction and propose a novel, theoretically sound, and universal approach that employs Bernstein-type polynomials for estimating the increasing triangular transformation between densities. The proposed method holds several advantages over previous attempts:

- robustness against initial and round-off errors due to the optimal stability of the Bernstein basis,
- existence of a theoretical upper bound for the error of approximation when the optimal convergence of the trainable parameters occur,
- being able to invert easily and accurately due to the availability of efficient root finding algorithms,
- suitability for approximating compactly supported target densities,
- the ability to increase the expressiveness by increasing the polynomial degree at no cost to the training stability,
- constructive universality proof which gives analytic expressions for approximations of known transformations.

The rest of the Chapter is structured as follows. In section 6.2, we introduce Bernstein-type polynomials and discuss the properties that lead to the advantages mentioned above. Afterwards, in section 6.3 we compare Bernstein-type polynomials with other classes of coupling maps and highlight the suitability of our method.

In section 6.4, we describe briefly how we construct the NF using Bernstein-type polynomials. Finally, in section 6.6, we verify our claims empirically by testing our NF on synthetic and real-world data sets.

6.2 Bernstein polynomials

Bernstein polynomials (of degree n),

$$\binom{n}{k} x^k (1-x)^{n-k},\ 0 \leq k \leq n,\ n \in \mathbb{N}, \tag{6.1}$$

were first introduced by Bernstein in his constructive proof of the Weierstrass theorem in Bernstein [1912]. In fact, given a continuous function $f : [0,1] \to \mathbb{R}$, its degree n Bernstein approximation, $B_n(f) : [0,1] \to \mathbb{R}$, given by

$$B_n(f)(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}, \tag{6.2}$$

is such that $B_n(f) \to f$ uniformly in $[0,1]$ as $n \to \infty$. Moreover, Bernstein polynomials form a basis for degree n polynomials on $[0,1]$. More generally, polynomials of Bernstein-type can be defined as follows.

Definition 6.1. *A degree n polynomial of Bernstein-type is a polynomial of the form*

$$B_n(x) = \sum_{k=0}^{n} \alpha_k \binom{n}{k} x^k (1-x)^{n-k},\ x \in [0,1] \tag{6.3}$$

where α_k, $0 \leq k \leq n$ are some real constants.

Remark 6.1. *Polynomials of Bernstein-type on an arbitrary closed interval $[a,b]$ are defined by composing B_n with the linear map that sends $[a,b]$ to $[0,1]$, $L_{a,b}(x) = \frac{x-a}{b-a}$. So, Bernstein-type polynomials on $[a,b]$ take the form*

$$B_n \circ L_{a,b}(x) = \sum_{k=0}^{n} \alpha_k \binom{n}{k} \frac{(x-a)^k (b-x)^{n-k}}{(b-a)^n}. \tag{6.4}$$

Hereafter, we denote degree n Bernstein-type polynomials by B_n regardless of the domain.

Remark 6.2. *Note that $B_n(a) = \alpha_0$ and $B_n(b) = \alpha_n$. Therefore, one can fix the values of a Bernstein-type polynomial at the end points of $[a,b]$ by fixing α_0 and α_n.*

Due to the structure of Bernstein-type polynomials, they are ideal for constructing triangular flows between compactly supported densities. As we shall see below, one can control their properties like strict monotonicity, range and universality by specifying conditions on the coefficients, and the error of approximation depends on the degree of the polynomials used. As a result, our model is highly interpretable.

6.2.1 Strict monotonicity

In triangular flows, the class of transformations used are expected to be invertible. We can achieve this using increasing functions. Due to the structure of Bernstein-type polynomials, we can easily specify conditions on the coeffecients that guarantee their strict monotonicity. We state this as a theorem:

Theorem 6.1. *Consider the Bernstein-type polynomial in* (6.3)*. Suppose* $\alpha_0 < \alpha_1 < \cdots < \alpha_n$*. Then* B_n *is strictly increasing on* $[0,1]$*.*

Proof. Let $f : [0,1] \to \mathbb{R}$ be a strictly increasing continuous function such that $f(k/n) = \alpha_k$. Let $s < t$ and let Z_i^x, $0 \le i \le n$ and be a sequence of iid Bernoulli(x) for $x = s, t$, defined on the same probability space such that $Z_i^s \le Z_i^t$ via monotone coupling. That is, let $Z_i^s = \mathbf{1}_{U \le s}$ and $Z_i^t = \mathbf{1}_{U \le t}$ where U is a uniform random variables on $[0,1]$ and couple them as follows.

$$\mathbb{P}((Z_i^s, Z_i^t) = (j,k))_{j,k \in \{0,1\}} = \begin{pmatrix} 1-t & t-s \\ 0 & s \end{pmatrix}$$

and $\mathbb{P}(Z_i^s > Z_i^t) = \mathbb{P}(Z_i^s = 1, Z_i^t = 0) = 0$. So, $Z_i^t \ge Z_i^s$ as required.

Then

$$f\left(\sum_{i=0}^{n} Z_i^s/n\right) \le f\left(\sum_{i=0}^{n} Z_i^t/n\right).$$

Consequently,

$$\mathbb{E}\left(f\left(\sum_{i=0}^{n} Z_i^s/n\right)\right) \le \mathbb{E}\left(f\left(\sum_{i=0}^{n} Z_i^t/n\right)\right). \tag{6.5}$$

This is equivalent to $B_n(s) \le B_n(t)$.

If (6.5) is not strict, then $f(\sum_{i=0}^n Z_i^t/n) = f(\sum_{i=0}^n Z_i^s/n)$ almost surely, and therefore, $\sum_{i=0}^n Z_i^t = \sum_{i=0}^n Z_i^s$ almost surely. But this is impossible due to monotone coupling. Therefore, by contradiction, (6.5) is strict as required. □

Remark 6.3. *According to the Theorem 6.1, the strict monotonicity of Bernstein-type polynomials depends entirely on the strict monotinicity of the coefficients* α_k*'s. We saw in Remark 6.2 that* $B_n(a) = \alpha_0$ *and* $B_n(b) = \alpha_n$*. In addition, if we assume that* α_k*'s are increasing, then the range of* B_n *is the interval* $[\alpha_0, \alpha_n]$*. This translates to a significant advantage when training for compactly supported targets because we can achieve any desired range* $[c,d]$ *(the support of the target) by fixing* $\alpha_0 = c$ *and* $\alpha_n = d$ *and let only* α_k*,* $0 < k < n$ *vary.*

Remark 6.4. *The strict monotonicity of the coeffecients is not restrictive. For example, if the required range is* $[c,d]$*, we can take* $\alpha_{n-k} = c + (d-c)(1 + v_0^2 + \cdots + v_k^2)^{-1}$ *where* v_k*'s are real valued. This converts the constrained problem of finding* α_k*'s to an unconstrained one of finding* v_k*'s. Alternatively, we can take* $\alpha_0 = c$ *and* $\alpha_k = |v_1| + \cdots + |v_k|$*, and after each iteration, linearly scale* α_k*'s in such a way that* $\alpha_n = d$*.*

6.2.2 Universality

Whether the NFs are sufficiently expressive or not depends on the universality of the transformations used in their construction. Therefore, in triangular flows, we need to use a class of functions that well-approximates the class of increasing continuous functions. As we shall see below, the class of increasing Bernstein-type polynomials on $[a, b]$ are uniformly dense in the class of increasing continuous functions on $[a, b]$. In fact, we construct this sequence of polynomial approximations explicitly.

First, we recall the density result in Bernstein [1912].

Theorem 6.2 (Bernstein)**.** *Let $f : [0, 1] \to \mathbb{R}$ be given. Then $B_n(f)$ given by Equation 6.2 converges uniformly to f as $n \to \infty$.*

Remark 6.5. *By rescaling, the above theorem holds on any interval $[a, b]$. Moreover, by construction, whenever f is increasing, $B_n(f)$ is increasing. So, it is automatic that increasing Bernstein polynomials on $[a, b]$ are dense in increasing continuous functions, and in particular, increasing differentiable maps on $[a, b]$.* (6.4) *gives us the explicit formula for the degree n polynomial that approximates f.*

Next, to show universality, we need to prove that any increasing differentiable function $f : \mathbb{R} \to \mathbb{R}$ can be well-approximated by Bernstein-type polynomials. This is our next result.

Corollary 6.1. *Let $f : \mathbb{R} \to \mathbb{R}$ be continuous. Then there exists a sequence of Bernstein-type polynomials that converges to f point-wise on $\mathbb{R}$ and converges uniformly to f on any compact set.*

Proof. Choose two positive sequences $\{M_n\}$ and $\{\epsilon_n\}$ such that $M_n \to \infty$ and $\epsilon_n \to 0$. Let $I_n = [-M_n, M_n]$. Then, there exists a Bernstein approximation q_n on I_n (which extends naturally to $\mathbb{R}$) such that

$$\max_{I_n} |f - q_n| \leq \epsilon_n.$$

Then the sequence of Bernstein approximations $\{q_n\}$ converges point-wise to f on $\mathbb{R}$, and this convergence is uniform on each compact interval. □

Remark 6.6. *When f is increasing then q_n is increasing on $[-M_n, M_n]$. Also, when f is $\alpha-$Hölder and $M_n = \log n$, choosing the degree of q_n to be n is sufficient. This will be explained in Remark 6.7.*

6.2.3 Theoretical error bound

The error of approximation of a function f by its Bernstein polynomials has been extensively studied. We state two such result due to Voronovskaya [1932] and Kac [1938] and refer the reader to [Bustamante, 2017, Chapter 4] for their proofs and a detailed account of other error bounds.

Theorem 6.3 (Voronovskaya). *Suppose $f : [0,1] \to \mathbb{R}$ is twice continuously differentiable. Then*

$$B_n(f) - f = \frac{x(1-x)}{2n} f''(x) + o(n^{-1}).$$

and this error cannot be improved by increasing the smoothness of f.

Theorem 6.4 (Kac). *Suppose $f : [0,1] \to \mathbb{R}$ is $\alpha-$Hölder with holder constant L, then*

$$|B_n(f) - f| \leq L \left(\frac{x(1-x)}{2n} \right)^{\alpha/2}.$$

Remark 6.7. *These hold for an arbitrary interval $[a,b]$ with $x(1-x)$ replaced by $(x-a)(b-x)$. Thus, in Corollary 6.1, when f is $\alpha-$Hölder, $I_n = [-\log n, \log n]$ and q_n is the degree n Bernstein approximation, then the error estimate is $L(\log n)^2/n$ where L is the Hölder constant of f and $\lim_{n\to\infty} L(\log n)^2/n = 0$.*

Since we know the error estimate in terms of the degree of the polynomials used, we can improve the optimality of our NF by avoiding unnecessarily high degree polynomials. This keeps the number of trainable parameters under control.

6.2.4 Robustness

Due to the presence of random or systematic errors, sample data generated from experiments are generally a perturbation of their *true* value. These initial errors may get amplified in the NFs unless they are numerically stable. Moreover, the usage of finite-precision arithmetic in the execution causes round-off errors which affect the final outcome. Therefore, a discussion about the robustness of the construction of NFs is in order. In this section, we recall some known results in Farouki and Rajan [1987]; Farouki and Goodman [1996] about the optimal stability of the Bernstein basis. The key idea is that smaller *condition numbers* lead to smaller errors and the Bernstein basis has the optimal condition numbers compared to other polynomial bases.

To illustrate this, let $p(x)$ be a polynomial on $[a,b]$ of degree n expressed in terms of a basis $\{\phi_k\}_{k=0}^n$. That is,

$$p(x) = \sum_{k=0}^{n} c_k \phi_k(x),\ x \in [a,b].$$

Let c_k be randomly perturbed, with perturbations δ_k where the relative error $\delta_k/c_k \in (-\varepsilon, \varepsilon)$. Then the total pointwise perturbation is $\delta(x) = \sum_{k=0}^n \delta_k \phi_k(x)$. So, $|\delta(x)| \leq \sum_{k=0}^n |\delta_k \phi_k(x)| \leq \varepsilon \sum_{k=0}^n |c_k \phi_k(x)| \leq \varepsilon C_\phi(p(x))$, where

$$C_\phi(p(x)) = \sum_{k=0}^{n} |c_k \phi_k(x)|$$

is the condition number for total perturbation with respect to the basis ϕ_k. It is clear that $C_\phi(p(x))$ controls the magnitude of the total perturbation.

The following result compares the condition numbers for total perturbation with respect to two non-negative bases.

Theorem 6.5 (Farouki & Goodman)**.** *Suppose $\phi = \{\phi_k\}_{k=0}^n$ and $\psi = \{\psi_k\}_{k=0}^n$ are non-negative bases for polynomials of degree n on $[a, b]$. Suppose that for each j, ψ_j is a non-negative linear combination of the former, that is, $\psi_j = \sum_{k=0}^n M_{jk}\phi_k$ with $M_{jk} \geq 0$. Then for any polynomial $p(x)$, $C_\phi(p(x)) \leq C_\psi(p(x))$.*

Note that on $[a, b]$ where $0 \leq a < b$, the Bernstein polynomials and the power monomials, $\{1, x, x^2, \ldots, x^n\}$, are non-negative bases. It is true that the latter is a positive linear combination of the former but not vice-versa Farouki and Goodman [1996]. So, we have the following Corollary.

Corollary 6.2. *Let $0 \leq a < b$, $\{\psi_k := x^k\}$ and $\{\phi_k\}_{k=0}^n$ be the power basis and degree n Bernstein basis on $[a, b]$ respectively. Then for an arbitrary polynomial $p(x)$, $C_\phi(p(x)) \leq C_\psi(p(x))$.*

Remark 6.8. *This means that the upper bound on the change in the value of a polynomial caused by a perturbation of coefficients is always smaller in the Bernstein basis than in the power basis.*

A more involved computation yields the condition number

$$\tilde{C}_\phi(x_0) = \left(\frac{m!}{|p^{(m)}(x_0)|} \sum_{k=0}^{n} |c_k \phi_k(x_0)| \right)^{1/m}$$

that controls the computational error for a $m-$fold root x_0 of $p(x)$ in $[a, b]$ Farouki and Rajan [1987]. The following theorem establishes the robustness of the Bernstein basis for root finding.

Theorem 6.6 (Farouki & Rajan)**.** *For an arbitrary polynomial $p(x)$ with a root $x_0 \in [0, 1]$, let $\tilde{C}_\psi(x_0)$ and $\tilde{C}_\phi(x_0)$ denote the condition numbers of the root in the power and the Bernstein bases on $[0, 1]$, respectively. Then $\tilde{C}_\phi(x_0) < \tilde{C}_\psi(x_0)$ for $x_0 \in (0, 1]$ and $\tilde{C}_\phi(0) = \tilde{C}_\psi(0)$.*

Remark 6.9. *In fact, the condition numbers of Bernstein basis on $[a, b]$ are better than the condition numbers for the power basis adapted to $[a, b]$,* i.e., *$(x - a)^k, k = 0, 1, \ldots, n$ Farouki and Rajan [1987]. So, the above theorem does not depend on the choice of the interval $[0, 1]$.*

So, we conclude that compared to the power basis representation of a polynomial, the Bernstein representation (which we use), is systematically more stable against perturbations of coefficients, and the resulting errors when 1) evaluating polynomials and 2) computing roots are lower.

6.2.5 Inversion

It is crucial that the transformations used in the construction of NFs have inverses that are easy to compute. In our setting, this boils down to finding roots of Bernstein-type

polynomials. That is, at each interation, given x we solve for $z \in [0,1]$,

$$\sum_{k=0}^{n} \alpha_k \binom{n}{k} z^k (1-z)^{n-k} = x \tag{6.6}$$

which can be rewritten as

$$\sum_{k=0}^{n} (\alpha_k - x) \binom{n}{k} z^k (1-z)^{n-k} = 0 \tag{6.7}$$

as Bernstein polynomials form partition of unity on $[0,1]$. So, finding inverse images, *i.e.*, solving (6.6) is equivalent to finding solutions to (6.7). Due to our assumption of increasing α_k's, B_n is increasing, and has at most one root on $[0,1]$. The condition $(\alpha_0 - x)(\alpha_n - x) < 0$ (which can be easily checked) guarantees the existence of a unique solution of (6.7), and hence, the invertibility of the original transformation.

Due to the extensive use of Bernstein-type polynomials in computer-aided geometric design, there are several well-established efficient root finding algorithms at our disposal Spencer [1994]. For example, the parabolic hull approximation method in Rajan et al. [1988] is ideal for higher degree polynomials with fewer roots (in our case, just one) and has cubic convergence for simple roots (better than Newton's method). Further, because of the numerical stability described in section 6.2.4, Bernstein-type polynomials minimizes the errors in such root solvers based on floating–point arithmetic.

6.2.6 Examples of Bernstein-type approximations

In this section, we illustrate how to use Bernstein-type polynomials to approximate diffeomorphisms between densities. We restrict our attention to densities on $\mathbb{R}$.

Suppose F and G are the distribution functions of the two probability densities P_z and P_x on the real line. Then the *increasing rearrangement* $f = G^{-1} \circ F$ is the unique increasing transformation that pushes forward P_z to P_x, and this basic construction generalizes to higher dimensions [Villani, 2009, Chapter 1].

Now, we can easily write down transformations between some known densities, and using (6.4), we can explicitly write down their Bernstein-type approximations.

Example 6.1. Uniform$[0,1]$ *to any arbitrary non-zero continuous and compactly supported density P on $[0,1]$:*
Here, $G(x) = \int_0^x P(s)\,ds$, $x \in [0,1]$ is strictly increasing and hence, invertible on $[0,1]$. So, $f(x) = G^{-1}(x)$, and G^{-1} is once continuously differentiable. The degree n Bernstein approximation of f is

$$B_n(f)(x) = \sum_{k=0}^{n} G^{-1}\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}.$$

and $\|B_n(f) - f\|_\infty = O(n^{-1/2})$.

Example 6.2. Kumaraswamy(α, β) *to* Uniform$[0,1]$*:*
Here, $\alpha, \beta > 0$ *and for* $x \in [0,1]$, $F(x) = 1 - (1 - x^{\alpha})^{\beta}$ *Kumaraswamy [1980] and* $G(x) = x$. *Therefore,* $f(x) = F(x)$ *and the degree n Bernstein approximation is*

$$B_n(f)(x) = \sum_{k=0}^{n} F\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}.$$

When $\alpha, \beta \geq 1$, $\|B_n(f) - f\| = O(n^{-1})$.

Example 6.3. Uniform$[0,1]$ *to* Exponential(λ)*:*
Since $F(x) = x, \;\; x \in [0,1]$ *and* $G(x) = 1 - e^{-\lambda x}, \;\; x \geq 0,$

$$f(x) = -\frac{1}{\lambda} \ln(1-x) = \frac{1}{\lambda} \sum_{k=1}^{\infty} \frac{x^k}{k}, \;\; x \in [0,1).$$

Pick an increasing sequence $\{b_n\}$ *such that* $0 < b_n < 1$ *and* $\lim_{n\to\infty} b_n = 1$. *Then, the degree n Bernstein polynomial on* $[0, b_n]$*, given by*

$$q_n(x) = -\frac{1}{\lambda} \sum_{k=0}^{n} \ln\left(1 - \frac{kb_n}{n}\right) \binom{n}{k} \frac{x^k (b_n - x)^{n-k}}{b_n^n}$$

is such that $\lim_{n\to\infty} q_n(x) = f(x)$, $x \in [0,1)$. *This convergence is uniform on each compact subset of* $[0,1)$.

6.3 Theoretical comparison with other methods

Here, we compare and contrast properties of existing NFs and our construction and discuss its advantages as well as limitations. To keep the exposition brief, we restrict our discussion to universal NFs such as neural autoregressive flows (NAF) in Huang et al. [2018b], SOS flows in Jaini et al. [2019] and splines in Durkan et al. [2019a,b].

6.3.1 Approximations and error bounds

In all of these cases and ours, the universality proof is based on the fact that the learnable class of functions is dense in the class of increasing continuous functions. However, the argument we present here (see Corollary 6.1) is constructive. So, as shown in section 6.2.6, we can write down sequences of approximations for (known) transformations between densities. In fact, we can say more. We have a theoretical upper bound for the error of approximation. This is discussed in section 6.2.3. In the case of cubic splines of Durkan et al. [2019a]), it is known that for $k = 1, 2, 3$ and 4, when the transformation is k times continuously differentiable and the bin size is h, the error is $O(h^k)$ [Ahlberg et al., 1967, Chapter 2]. However, we are not aware of any other instance where an error bound is available.

We present an example to show that the error $O(n^{-1})$ in Theorem 6.3 does not necessarily improve when SOS polynomials are used instead.

Example 6.4. Uniform$[0,1]$ *to the* Normal$(0,1)$*:*

$$T(z) = \text{Erf}^{-1}(2z-1) = \sum_{k=0}^{\infty} \frac{\sqrt{2}\pi^{k+\frac{1}{2}} c_k}{2k+1} \left(z - \frac{1}{2}\right)^{2k+1},$$

where $\{c_k\}_{k\geq 0}$ *is a bounded positive sequence Jaini et al. [2019]. This is the power series expansion of* T *at* $z = 1/2$*, and hence, it is unique.*

The SOS approximation $\sum_{k=0}^{n} \frac{\sqrt{2}\pi^{k+1/2} c_k}{2k+1}(z-1/2)^{2k+1}$ *of* T *is only* $O((2n+1)^{-1})$ *accurate on any compact sub-interval of* $(0,1)$*. This is precisely the accuracy one would expect from the degree* $2n+1$ *Bernstein approximation on a compact subinterval of* $(0,1)$*.*

Even though the error bound under optimal convergence is $O(n^{-1})$, and in general, cannot be improved further regardless of how regular the transformation is. We compare the average error we obtain against the theoretical average error in an experiment. See section 6.6.2.

6.3.2 Numerical stability

From Farouki and Rajan [1987], we know that Bernstein-type polynomials are systematically more stable than the polynomials in the power form when determining roots (for example, when inverting) and evaluation (for example, when finding image points). This was discussed in section 6.2.4. In fact, due to Farouki and Goodman [1996] the Bernstein polynomial basis on a given interval is optimally stable, in the sense that no other non-negative basis gives smaller condition numbers for the values or roots of arbitrary polynomials on that interval. As a result, when polynomials are used to construct NFs, i.e., quadratic or cubic splines, SOS polynomials, and etc., Bernstein-type polynomials yield the most robust NF. We verify this experimentally in section 6.6.3.

Inverting splines are easier due to the availability of analytic expressions for roots. However, we have efficient and numerically stable algorithms like convex hull approximation and parabolic hull approximation to find roots of Bernstein-type polynomials. The latter is faster than both the bisection method and the Newton's method Spencer [1994]. This allows us to reduce the cost of numerical inversion in our setting.

6.3.3 Applicability to compact densities

In any NF model, even if the target density is not compactly supported, we can implement the learning procedure by first converting the target density to a density with a support of our choice via a suitable invertible transformation.

Let $B_n : [0,1] \to [a,b]$ be the learnable Bernstein-type polynomial and let $h : [a,b] \to \mathbb{R}$ be a fixed invertible transformation so that h^{-1} transforms the target density to a one supported on $[a,b]$. So, $\alpha_0 = a$ and $\alpha_n = b$. Let $I = \{(\alpha_1, \ldots, \alpha_{n-1}) \mid a <$

$\alpha_1 < \cdots < \alpha_{n-1} < b\}$. Then the optimization problem is

$$\begin{aligned}
&\min_I \mathrm{KL}(P_x \| (h \circ B_n)_* P_z) \\
&= -\max_I \int \log \frac{P_z(B_n^{-1}(h^{-1}\mathbf{x}))}{|\mathbf{J}_{h\circ B_n}(B_n^{-1}(h^{-1}\mathbf{x}))|} \cdot P_x(\mathbf{x})\, d\mathbf{x} \\
&= -\max_I \int \log \frac{P_z(B_n^{-1}(h^{-1}\mathbf{x}))}{|\mathbf{J}_h(h^{-1}\mathbf{x})\mathbf{J}_{B_n}(B_n^{-1}(h^{-1}\mathbf{x}))|} \cdot P_x(\mathbf{x})\, d\mathbf{x} \\
&= -\max_I \int \log \frac{P_z(B_n^{-1}(h^{-1}\mathbf{x}))}{|\mathbf{J}_{B_n}(B_n^{-1}(h^{-1}\mathbf{x}))|} \cdot P_x(\mathbf{x})\, d\mathbf{x} \\
&\qquad + \int \log |\mathbf{J}_h(h^{-1}\mathbf{x})| \cdot P_x(\mathbf{x})\, d\mathbf{x}.
\end{aligned} \tag{6.8}$$

Note that the second integral can be taken outside the max because it is independent of B_n, and hence, a constant that is irrelevant for the optimization. So, wlog we can assume that the target density is compactly supported.

We note that this argument holds true even when we replace B_n with an arbitrary f from a suitable class $\mathfrak{F}$. So, for example, in Durkan et al. [2019a,b] and in our case, the assumption that the range of the transformations used is finite (as opposed to, say, SOS polynomials from $\mathbb{R}$ to $\mathbb{R}$ having infinite range) is not restrictive.

Since we deal with compactly supported targets, in practice, we do not need to construct deep architectures (with a higher number of layers), as we can increase the degree of the polynomials to get a more accurate approximation. Although the same argument seems valid for the other polynomial based methods, a practical problem arises where the higher order polynomials may predict extremely high values initially, which can cause unstable gradients. In contrast, since the range of our transformations can be explicitly controlled by constraining α_0 and α_n (see Remark 6.3), the same problem does not occur. For the same reason of being able to control the range by simply fixing α_0 and α_n, Bernstein-type polynomials are ideal for modeling compactly supported targets. In fact, in most other methods except splines, either there is no obvious way to control the range or the range is infinite.

6.3.4 Intepretability

In general, polynomial based NFs are more interpretable than NAFs because we know how certain properties of polynomials determine properties of the target density. For example, coefficients of SOS polynomials control the first few moments of the target density Jaini et al. [2019]. In the Bernstein case, we can say more.

- If α_k's are increasing, then so is B_n, and the support of the target density is $[\alpha_0, \alpha_n]$.

- The condition numbers C_ϕ and $\tilde{C}_\phi$ given in section 6.2.3 control the error of evaluation and inversion, respectively.

- The error of approximation corresponds to the degree of the Bernstein-type polynomial in a precise way.

Therefore, our model is highly interpretable.

6.4 Bernstein-type Normalizing Flow

Now, we construct normalizing flows using the theoretical framework established in section 6.2 for compactly supported targets. Consider a d-dimensional source $P_z(\mathbf{z})$ and a d-dimensional target $P_x(\mathbf{x})$. Then, the element-wise mapping between the components x_j and z_j is approximated using a Bernstein-type polynomial as $x_j = B_n^j(z_j)$. We obtain the parameters of $B_n^j(z_j)$ using a neural network which is conditioned on $z_{<j}$. This ensures a triangular mapping between the distributions.

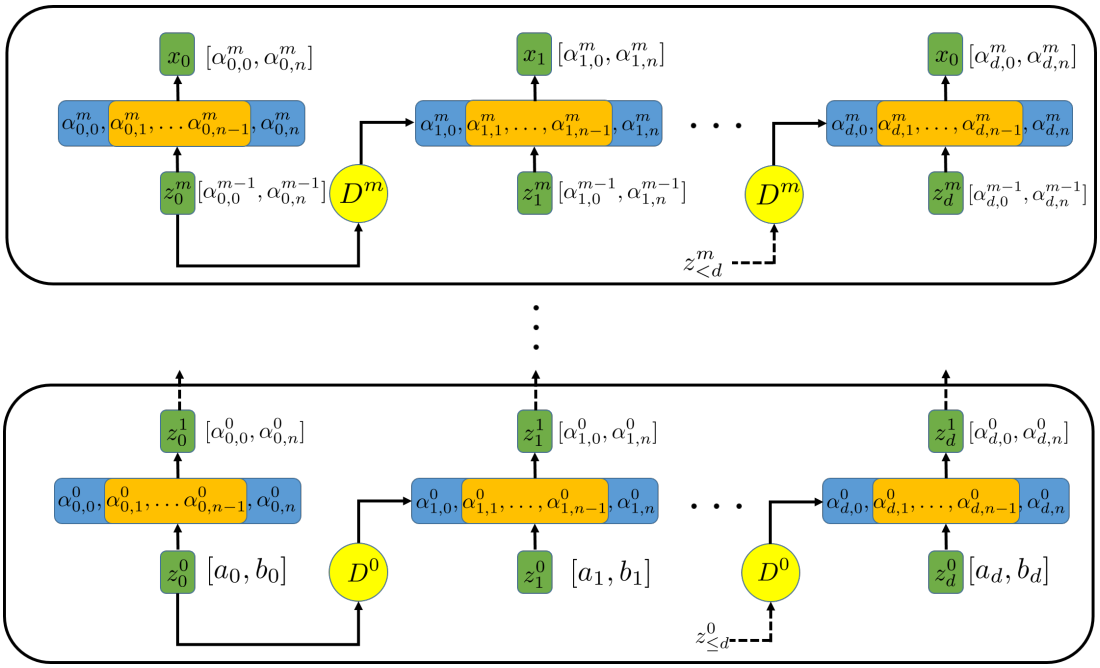


Figure 6.1: Overall Bernstein-NF architecture with $m+1$ layers for d-dimensional distributions. The range of transformations are within brackets and trainable coefficients are in orange boxes.

As per Remark 6.3, we fix α_0 and α_n to be constants, and thus, define the range of each transformation. Moreover, due to Theorem 6.1, α_k's need to be strictly increasing for transformation to be strictly increasing. However, when we convert this constrained problem to an unconstrained one as proposed by Remark 6.4, we may obtain v_k's using the neural network and then calculate α_k's as described there.

For each B_n^j, we employ a fully-connected neural net with three layers to obtain the parameters, except in the case of B_n^0 for which we directly optimize the parameters.

Model	Power	GAS	hepmass	miniboone	bsds300
FFJORD	0.46 ± 0.01	8.59 ± 0.12	-14.92 ± 0.08	-19.43 ± 0.04	157.40 ± 0.19
GLOW	0.42 ± 0.01	12.24 ± 0.03	-16.99 ± 0.02	-10.55 ± 0.45	156.95 ± 0.28
MAF	0.45 ± 0.01	12.35 ± 0.02	-17.03 ± 0.02	-10.92 ± 0.46	156.95 ± 0.28
NAF	0.62 ± 0.01	11.96 ± 0.33	-15.09 ± 0.04	-8.86 ± 0.15	157.73 ± 0.04
BLOCK-NAF	0.61 ± 0.01	12.06 ± 0.09	-14.71 ± 0.38	-8.95 ± 0.07	157.36 ± 0.03
SOS	0.60 ± 0.01	11.99 ± 0.41	-15.15 ± 0.10	-8.90 ± 0.11	157.48 ± 0.41
RQ-NSF (AR)	0.66 ± 0.01	13.09 ± 0.02	-14.01 ± 0.03	-9.22 ± 0.48	157.31 ± 0.28
BERNESTEIN	0.63 ± 0.01	12.81 ± 0.01	-15.11 ± 0.02	-8.93 ± 0.08	157.13 ± 0.11

Table 6.1: Test log-likelihood comparison against the state-of-the-art on real-world datasets (higher is better). Log-likelihoods are averaged over 10 trials in SOS and Bernstein.

Figure 7.2 illustrates a model architecture with $m+1$ layers and degree n polynomials for d-dimensional distributions. Here, there are $(n-1)(m+1)$ variable coeffecients, altogether.

6.5 Hyper-parameters and training details

For optimization, we used the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_1 = 0.999$, $\epsilon = 1 \times 10^{-8}$, where parameters refer to the usual notation. An initial learning rate of 0.01 was used for updating the weights with a decay factor of 10% per 50 iterations. We initialized all the trainable weights using a random standard normal distribution and used maximum likelihood as the objective function for training. We observed that a single layer model with 100 degree polynomials performed well for the real-world data. In contrast, for 2D toy distributions and and images we used higher number of layers (8) with 15 degree polynomials in each layer. For all the experiments, we use a Kumaraswamy distribution with parameters $a = 2$ and $b = 5$ as the base density. However, using a standard normal distribution after converting it to a density on $[0, 1]$ using a nonlinear transformation e.g., $\frac{1+\tanh(z)}{2}$, also yielded similar results.

6.6 Experiments

In this section, we summarise our empirical evaluations of the proposed model based on both real-world and synthetic datasets and compare our results with other NF methods.

6.6.1 Modeling sample distributions

We conducted experiments on four datasets from the UCI machine-learning repository and BSDS300 dataset. Table 6.1 compares the obtained test log-likelihood against recent flow-based models. As illustrated, our model achieves competitive results

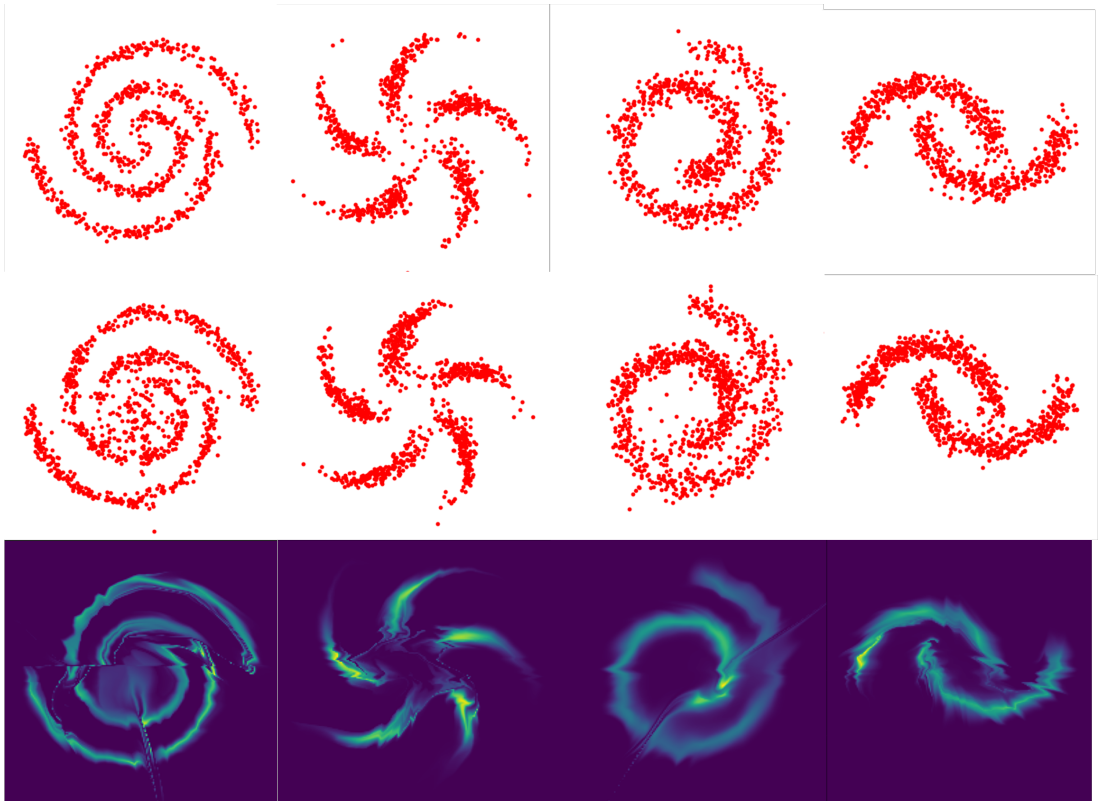


Figure 6.2: Qualitative results for modeling the toy distributions. *From the top row*: ground truth, prediction, and predicted density.

on all of the five datasets. We observe that our model consistently reported a lower standard deviation which may be attributed to the robustness of our model.

We also applied our method to two low-dimensional image datasets, CIFAR10 and MNIST. The results are reported in Table 6.2. Among the methods that do not use multi-scale convolutional architectures, we obtain the optimal results. In addition, we tested our model on several toy datasets (shown in Fig. 6.2). Note that these 2D datasets contain multiple modes, sharp jumps and are not fully supported. So, the densities are not that obvious to learn. Despite the difficulties, our model is able to estimate the given distributions in a satisfactory manner.

6.6.2 Validation of the theoretical error upper-bound

The degree $n\,(\geq 5)$ Bernstein approximation of a function $f \in C^3[0,1]$ has an error upper-bound

$$E_n = n^{-1}\|\rho^2 f^{(2)}\|_\infty + n^{-3/2}\|\rho^3 f^{(3)}\|_\infty$$

where $\rho(x) = \sqrt{x(1-x)}$ [Bustamante, 2017, Chapter 4]. This is an improved version of the Theorem 6.3. Now, we verify this experimentally and show that the observed (average) error is smaller than this theoretical upper-bound. To this end, we use

a Kumarswamy$(2,5)$ distribution as the prior and Uniform$[0,1]$ as the target (see Example 6.2). We can compare the average error, $\int_0^1 |f(x) - q_n(x)|\,dx$ where $f(x) = 1-(1-x^2)^5$ and q_n is the learned degree n Bernstein-type polynomial, and the theoretical error,

$$1.25n^{-1} + 5n^{-3/2} < E_n < 1.25n^{-1} + 5.5n^{-3/2}.$$

The Fig. 6.3 illustrates the desired conclusion.

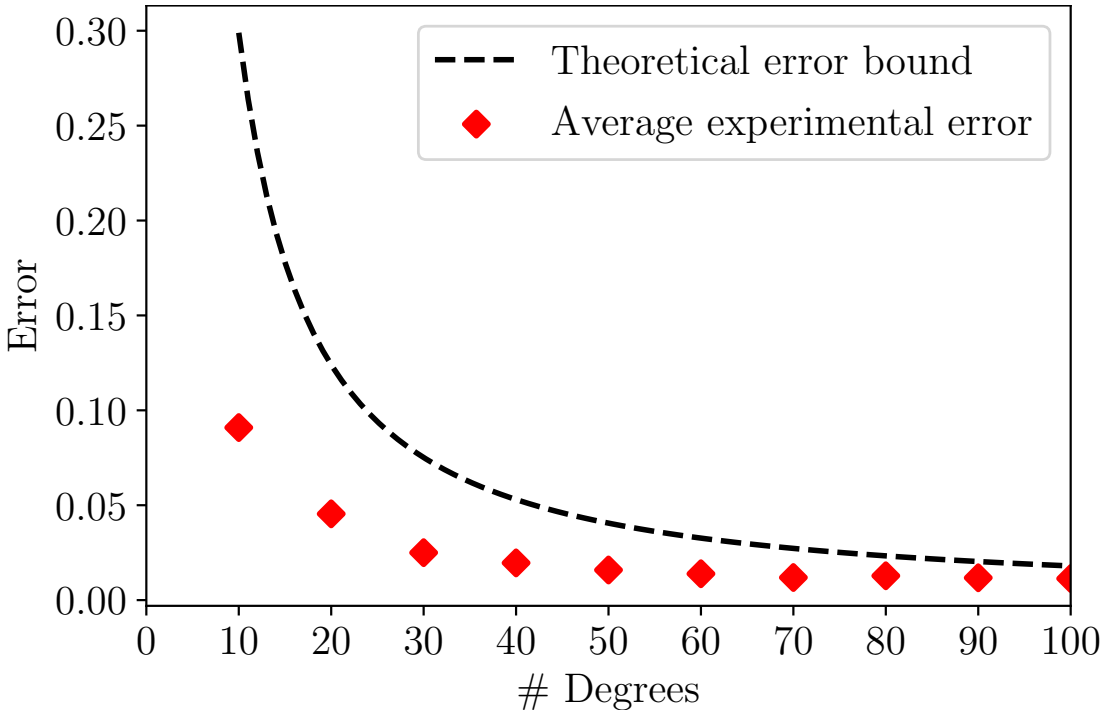


Figure 6.3: Theoretical error bound vs (averaged) experimental error.

In this case, in the NF, we have used a single layer and increased the degree of the polynomial from 10 to 100. The NF model was stable even when the degree 100 polynomial was used. So, this experiment also demonstrates that our model is, in fact, stable even when higher degree polynomials are used (as claimed in Section 6.3.3).

6.6.3 Robustness

In order to experimentally verify that Bernstein-NFs are more numerically stable than other polynomial based NFs (as claimed in Section 6.2.4), we add i.i.d. noise (sampled from a Uniform$[0, 0.01]$) to POWER and GAS datasets, and measure the change in the test log-likelihood as a fraction of the standard deviation. For a fair comparison, we train all the models from the scratch over 5 trials to obtain the standard deviations and the test log-likelihoods. The Table 6.3 illustrates these results. As expected,

Bernetein-type polynomials demonstrate the lowest change in performance, implying the robustness against random initial errors.

Model	MNIST	CIFAR10
Real NVP	-1.06	-3.49
GLOW	-1.05	-3.35
FFJORD	-0.99	-3.40
MADE	-2.04	-5.67
MAF	-1.89	-4.31
SOS	-1.81	-4.18
BERNSTEIN	-1.54	-4.04

Table 6.2: Test log-likelihood comparison against the state-of-the-art on image datasets (higher is better). First three used multi-scale convolutional architectures.

Model	POWER	GAS
RQ-NSF	2.3	5.4
SOS	2.1	1.7
MADE	2.1	4.6
MAF	2.4	4.4
BERNSTEIN	1.1	1.3

Table 6.3: Test log-likelihood drop for random initial errors, as a ratio of the standard deviations obtained using original data.

6.7 Ablation study

We compare the performance of different variants of our model against a simple task, in order to better understand the design choices. To this end, we use a standard normal as the base distribution, and a mixture of five Gaussians with means $= (-5, -2, 0, 2, 5)$, variances = $(1.5, 2, 1, 2, 1)$, and weights 0.2 each, as the target. Fig. 6.4 depicts the results.

Clearly, we were able to increase the expressiveness of the transformation by increasing the degree of the polynomials, as well as the number of layers. However, it is also visible that using an unnecessarily higher degree over-parametrizes the model, and hence, deteriorate the output. As discussed in the main article, we are able to use polynomials with degree as high as 100 in this experiment and others with no cost to the training stability because the training is done for a compactly supported target.

We also examine how the initial base distribution affects the performance. We use a mixture of seven Gaussians with means $= (-7, -5, -2, 0, 2, 5, 7)$, variances = $(1, 1, 2, 2, 2, 1, 1)$, and weights $= (0.8, 0.2, 0.2, 0.6, 0.2, 0.2, 0.8)$, as the target. We used a model with a 100-degree polynomial and a single layer for this experiment. Fig. 6.5

illustrates the results. Although all priors capture the multimodes, when Uniform$[0, 1]$ is used the model was not able to predict that the density is almost zero for large negative values.

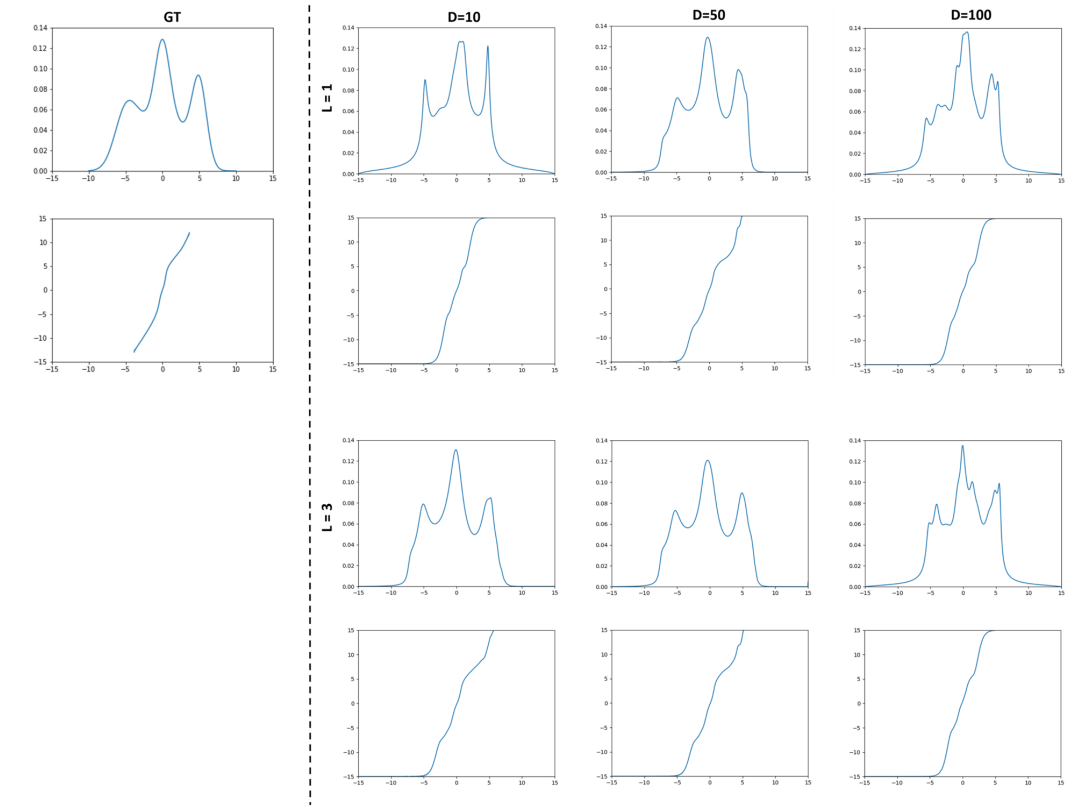


Figure 6.4: Ablation study with different varients of our model. **D** and **L** denotes the degree of the used polynomials and the number of layers, respectively. Corresponding transformation functions are also shown below the predicted densities.

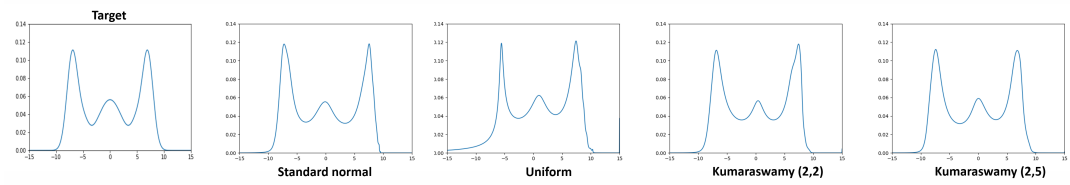


Figure 6.5: Approximation of the target density starting from various initial densities (the initial distributions are noted below the densities).

6.8 Chapter summary

In this Chapter, we proposed a novel method to construct a universal autoregressive normalizing flow with Bernstein-type polynomials as the coupling functions. Our theoretical discussion on Bernstein-type polynomials concluded that they possess properties ideal for constructing normalizing flows, and as a result, our method has several advantages like robustness against initial and round-off errors, higher interpretability, theoretical error bound, and better training stability for higher degree polynomials. This allows us to induce significant inductive bias into the model by controlling the bounds of the target density, and by explicitly designing the model to achieve an upperlimit for the error. Another important aspect of our model is the constructive universality proof, which does not exist for the competing methods. We compared our method to other universal normalizing flows both theoretically and experimentally. Our experiments on real-world and synthetic datasets (including 2D multimodal ones) provided competitive results and show that our theoretical claims hold in practice.

Chapter 7

Efficient high-resolution point cloud generation on unit sphere

Thus far, we showed that instilling task-dependant physics priors into machine learning models can significantly improve their efficacy and performance, across different problem settings. From here onward, we turn our attention towards inducing another significant aspect of the human brain into machine learning systems. We show that the concept of combinatorial generalization — dividing a learning system into smaller modules that specialize on solving a particular task — can be extremely effective in learning representations. In this chapter, we show the efficacy of combinatorial generalization by following a use case of 3D data synthesis.

Point-clouds are a popular 3D representation for real-world objects and scenes. In comparison to other representations such as voxels, mesh and truncated signed distance function (TSDF), point-clouds are often an attractive choice for 3D data because they capture shape details accurately, are computationally efficient to process and can be acquired as a default output from several 3D sensors (e.g., LiDAR). However, point-clouds pose a major challenge for deep networks, particularly the generative pipelines, due to their inherent redundancy and irregular nature (e.g., permutation-invariance).

Due to the complexity of point-clouds, most 3D synthesis approaches are inapplicable. For example, generative approaches designed for voxelized inputs [Wu et al., 2016; Kingma and Welling, 2013; Wu et al., 2015; Xie et al., 2018a; Huang et al., 2017b; Khan et al., 2019], cannot work with the irregular point sets. To overcome this challenge, some recent generative approaches solely focus on point-cloud synthesis. For example, Achlioptas et al. [2017a] use a GAN framework for 3D point-cloud distribution modelling in the data and auto-encoder latent space, Yang et al. [2019b] sample 3D points from a prior spatial distribution and then transform them using an invertible parameterization while Shu et al. [2019]; Valsesia et al. [2018] employ graph-structured networks for point-cloud generation.

All such efforts so far, operate in the 'spatial-domain' (3D Euclidean space) which makes the modelling task relatively difficult due to arbitrary point configurations in 3D space. This leads to a number of roadblocks towards a versatile generative model e.g., considering a fixed set of points [Achlioptas et al., 2017a] and limited scalability

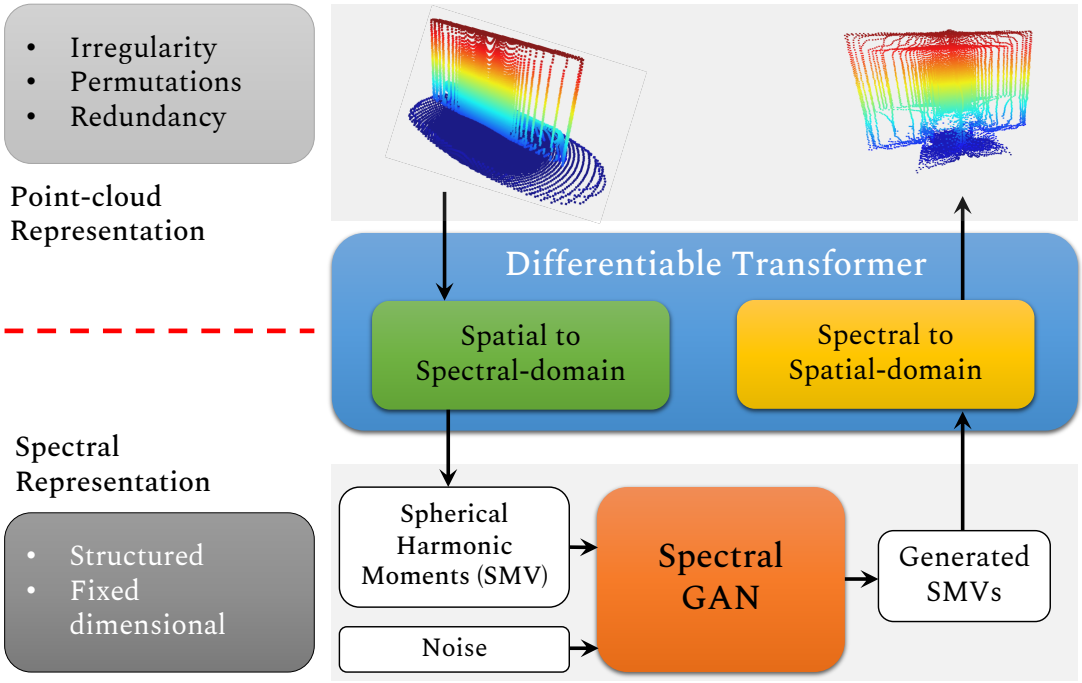


Figure 7.1: *Overview of Spectral-GAN.* Our model operates in the spectral domain using spherical harmonic moment vectors (SMVs). This allows us to avoid the redundancy and irregularity of point-clouds. Using a differentiable transformer, our model can also receive guidance from the spatial domain.

to arbitrary point resolutions [Shu et al., 2019; Valsesia et al., 2018]. As opposed to previous works, we perform generative modelling in the spectral space using spherical harmonic moment vectors (SMVs), which inherently offers a solution to the above mentioned problems. Specifically, generating 3D shapes via spectral representations allows us to compactly represent redundant information in point-clouds, easily scale to high-dimensional point-cloud sets, remain invariant to the permutations in unordered point sets and generate high-fidelity shapes with relatively minimal outliers. Besides, our spectral representation allow us to develop an understanding about the frequency domain functional space of generic 3D objects. Our main contributions are:

- To handle the redundancy and irregularity of point-clouds, we propose the first spectral-domain GAN that synthesizes novel 3D shapes by using a spherical harmonics based representation.
- A fully differentiable transformation from the spectral to the spatial domain and back, thus allowing us to integrate knowledge from well-established spatial models.
- Through both quantitative and qualitative evaluations, we illustrate that Spectral-GAN can generate high-quality 3D shapes with minimal artifacts and can be easily scaled to high-dimensional outputs.
- Our proposed framework learns highly discriminative unsupervised features and can seamlessly perform 3D reconstruction from 2D inputs. Moreover, we show that Spectral-GAN is scalable to high-resolution outputs (40$\times$ resolution increase with just 4$\times$ parameters).

7.1 Related Work

Generative models in spectral-domain: Yang et al. [2017b] and Souza and Frayne [2018] develop methods for MRI reconstruction using GANs, and use Fourier domain information to refine the output. In the former approach, the generator operates in the spatial domain, and spectral information is used to refine the output. The latter approach, in contrast, uses two separate networks in the frequency and spatial domains and adopts the Fourier transform to exchange information between the two. A significant drawback of these approaches is that output resolution is tightly coupled to the network design and thus they lack scalability to high dimensions.

In a different application, Portilla and Simoncelli [2000] present a method to synthesize textures as 2D images based on a complex wavelet transform. They parameterize this operation using a set of statistics computed on pairs of coefficients corresponding to basis functions at adjacent spatial locations, orientations, and scales. However, their approach is not a learning model, which offers less flexibility. Furthermore, Zhu et al. [2018] recently proposed a model that initially processes undersampled input data in the frequency domain and then refines the result in the spatial domain using the inverse Fourier transform. They approximate the inverse Fourier transform using a sequence of connected layers, but one disadvantage is that their transformation has quadratic complexity with respect to the size of the input image. Furthermore,

the above works are limited to 2D and do not study the 3D point-cloud generation problem in spectral domain.

3D GANs in spatial-domain: 3D GANs can be primarily categorized into two types: voxel outputs and point-cloud outputs. The latter typically entails more challenges as point-clouds are unordered and highly irregular in nature.

For voxelized 3D object modeling, several influential methods have been proposed in the literature. Wu et al. [2016] extend the 2D GAN framework to 3D domain for the first time. Following their work, Smith and Meger [2017] use a novel GAN architecture for 3D shape generation by employing Wasserstein distance as the loss function. A recent work by Khan et al. [2019] presents a factorized 3D generative model that sequentially generates shapes in a coarse-to-fine manner. Our approach also uses a two-step procedure–a forward pass and backward pass—to refine a coarse 3D shape, but a key difference here is that they use spatial information to refine the shape, while our method depends on frequency information.

Naive extensions of traditional spatial GANs to 3D point-cloud generation do not produce satisfactory results, due to their inherent properties such as being an unordered, irregularly distributed collection (see Sec. 7.2). Achlioptas et al. [2017a] were the first to use GANs to generate point-clouds. They first convert a point-cloud to a compact latent representation and then train a discriminator on it. Although we also use a compact representation, i.e., , the SMV to train the GAN, SMVs provide a richer representation compared to latent space approximations and theoretically guarantee accurate reconstruction of the 3D point-cloud. Moreover, Valsesia et al. [2018] propose a graph convolution based network to extract localized features from 3D point-clouds, in order to reduce the effect of irregularity. A drawback of their method, however, is the rather high computational complexity of graph convolution, and less scalability with the resolution of the point-cloud. A recent work by Shu et al. [2019] also propose a tree-structured graph convolution network, which is more computationally efficient. The model proposed by Li et al. [2018a] attempts to handle the irregularity of point-clouds using a separate inference model which captures a latent distribution, to deal with the irregularity of point-clouds. In contrast, we effectively reduce the problem to the standard GAN setting by using a fixed-dimensional representation for point-clouds.

7.2 Problem Formulation

An *exchangeable* sequence can be considered as a sequence of random variables $\tilde{X} = \{x_i\}_{i=1}^n$, where the joint probability distribution of $\tilde{X}$ does not vary under position permutations. More formally,

Definition: *For a given finite set $\{x_i\}_{i=1}^n$ of random objects, let $\mu_{x_1,x_2,\ldots,x_n}$ be their joint distribution. This finite set is exchangeable if $\mu_{x_1,x_2,\ldots,x_n} = \mu_{x_{\pi(1)},x_{\pi(2)},\ldots,x_{\pi(n)}}$, for every permutation $\pi : \{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$.* The spatial representation X of a point-cloud is a *set* of d-dimensional vectors, and in cases of Euclidean geometry, typically, $d = 3$. A *set* is a collection of elements without any particular order or a fixed number of

elements and thus, the probability distribution $p(X)$ is an exchangeable sequence. According to the Hewitt and Savage theorem [Hewitt and Savage, 1955], there exists a latent distribution Q such that,

$$p(X) = \int p(x_1, x_2, \ldots, x_n \mid Q)\, p(Q) dQ. \tag{7.1}$$

Eq. 7.1 shows that in order to properly model X as an exchangeable sequence and obtain a distribution $p(X)$, it is necessary to capture the latent representation Q. In other words, it is difficult for a GAN to model X as an exchangeable sequence, only by observing a set of X sequences and estimating the marginal distributions $p(x_i), i \in [1, \ldots, n]$. In this case, the generative model needs to learn the joint probability distribution $p(x_i, Q)$ instead of $p(x_i)$. This makes it challenging to extend traditional GANs to the point-cloud generation problem. A straightforward approach to resolve this is to model point-cloud data as ordered, fixed-dimensional vectors. However, this approach does not hold the integral probability metric (IPM) guarantees of a GAN [Li et al., 2018a].

On the contrary, we propose to model point-cloud data as SMVs, which effectively reduces the problem to the traditional case in two ways: 1) SMVs encode the corresponding shape information in a structured, fixed dimensional vector and 2) the vector elements are highly correlated with each other. The task of learning the distribution of elements of SMVs is theoretically similar to learning the pixel distribution of images, as in the latter case also, we only need to capture the joint probability distribution of pixels of each instance. In the case of image synthesis, however, GANs exploit the correlation of pixels using convolution kernels, which is not possible in the case of SMVs as correlation does not depend on proximity. Furthermore, different frequency portions of the SMVs show different characteristics. To handle these specific attributes, we propose a series of cascaded GANs, each consisting of only fully connected layers. Since each GAN only needs to generate a specific portion of the SMV, they can be designed as shallow models with fewer floating point operations (FLOPs).

7.3 Spectral GAN

We propose a 3D generative model that operates entirely in the spectral domain. Such a design offers unique advantages over spatial domain 3D generative models: (a) a compact representation of 3D shapes with an intuitive frequency-domain interpretation, (b) the flexibility to generate high-dimensional shapes with minimal changes to the model complexity, and (c) a permutation invariant representation which handles the irregularity of point-clouds. Below, we first introduce the spherical harmonics representations that serve as the basis for our proposed Spectral GAN model.

7.3.1 Spherical Harmonics for 3D Objects

Spherical harmonics are a set of complete and orthogonal basis functions, which can efficiently represent functions on the unit sphere $\mathbb{S}^2$ in $\mathbb{R}^3$. They are a higher

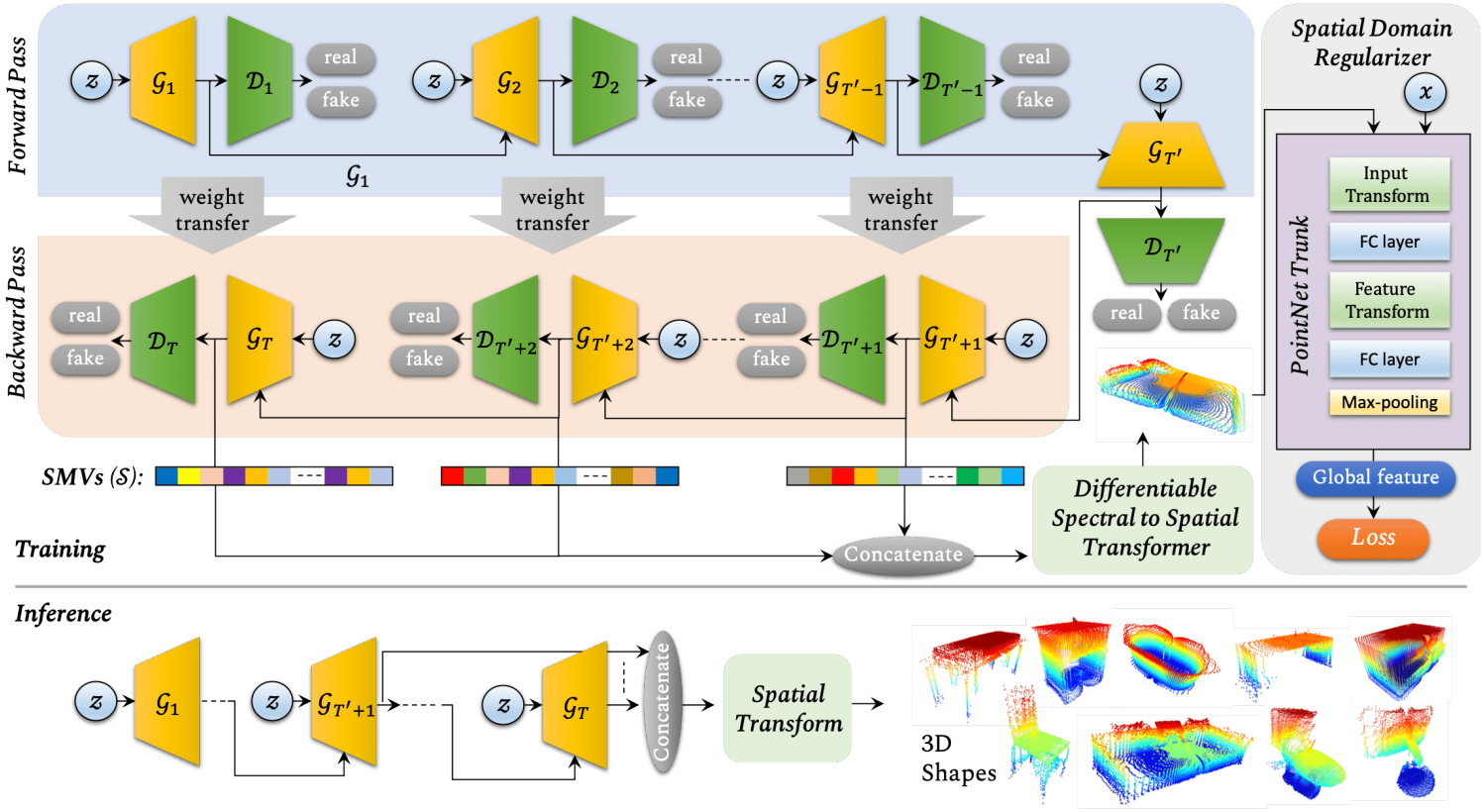


Figure 7.2: The overview of the Spectral Generative Adversarial Network. An unrolled version (with an explicit forward and backward pass) of the training scheme is shown for clarity.

dimensional analogy of the Fourier series, which forms a basis for functions on unit circle. The spherical harmonics are defined on $\mathbb{S}^2$ as,

$$Y_l^m(\theta,\phi) = N_l^m P_l^m(\cos\phi) e^{im\theta}, \tag{7.2}$$

where $\phi \in [0,\pi]$ is the polar angle, $\theta \in [0,2\pi]$ is the azimuth angle, $l \in \mathbb{Z}^+$ is a non-negative integer, $m \in \mathbb{Z}$ is an integer, $|m| < l$, $i = \sqrt{-1}$ is the imaginary unit, $N_l^m = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}$ is the normalization coefficient and $P_l^m(\cdot)$ is the associated Legendre function,

$$P_l^m(x) = (-1)^m \frac{(1-x^2)^{\frac{m}{2}}}{2^l l!}\frac{d^{l+m}}{dx^{l+m}}(x^2-1)^l. \tag{7.3}$$

Since spherical harmonics are orthogonal and complete over the continuous functions on $\mathbb{S}^2$ with finite energy, such a function $f : \mathbb{S}^2 \to \mathbb{R}$ can be expanded as,

$$f(\theta,\phi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} c_l^m Y_l^m(\theta,\phi), \tag{7.4}$$

where c_l^m are the spherical harmonic moments obtained by,

$$c_l^m = \int_0^{\pi}\int_0^{2\pi} f(\theta,\phi) Y_l^m(\theta,\phi)^{\dagger} \sin\phi \, d\phi d\theta. \tag{7.5}$$

The sufficient conditions for the expansion in Eq. 7.4 are given in [Hobson, 1931]. In practical cases, a bounded set of spherical harmonic basis functions $(M+1)^2$ is defined, where M is the maximum degree of harmonics series.

The process of 3D shape modeling via spherical harmonics can be decomposed into two major steps. First, sample points from the 3D shape surface and then computing spherical harmonic moments. Any polar 3D surface function can be represented as $r = f(\theta,\phi)$, where $f(\theta,\phi)$ is a single valued function on the unit sphere $\mathbb{S}^2$, r is the radial coordinate with respect to a predefined origin inside an object, and (θ,ϕ) is the direction vector. Thus, we can compute moments of the corresponding 3D point-cloud using Eq. 7.5.

7.3.2 Cascaded GAN Structure

SMVs provide a highly structured representation of 3D objects, as explained in Sec. 7.3.1. Due to this structured nature, the margin for error is significantly lower in our setup, compared to GANs that try to produce spatial domain representations. Also, different frequency bands of the SMV typically entail different characteristics, which makes it highly challenging for a single GAN to generalize over the complete SMV. Therefore, to overcome this obstacle, we use multiple cascaded GANs, where each GAN specializes in generating a pre-defined frequency band of the SMV.

Our approach uses a combination of T GAN models to generate the SMV of 3D shapes. Among them, the first model is a regular GAN while the remaining $T-1$

models are conditional GANs (cGAN). The objective of initial GAN model is given by a two-player min-max game,

$$\min_{\mathcal{G}_1} \max_{\mathcal{D}_1} L_{GAN}(\mathcal{G}_1, \mathcal{D}_1) = \mathbb{E}_{\bar{g}_1}[\log \mathcal{D}(\bar{g}_1)] + \mathbb{E}_{z_1}[\log(1 - \mathcal{D}(\mathcal{G}(z_1)))], \tag{7.6}$$

where $\bar{g}_i \sim p_g$ is the SMV band sampled from the spectral coefficient distribution and $z \sim p_z$ is the noise vector sampled from a Gaussian distribution. In a cGAN, synthetic data modes are controlled by forwarding conditioning variables (e.g., a class label) as additional information to the generator. In our case, we use a specific band of SMVs g_i predicted by the previous generator to condition the subsequent generator. Then, the cGAN objective becomes,

$$\min_{\mathcal{G}_i} \max_{\mathcal{D}_i} L_{cGAN}(\mathcal{G}_i, \mathcal{D}_i) = \mathbb{E}_{\bar{g}_i}[\log \mathcal{D}(\bar{g}_i)] + \mathbb{E}_{g_{i-1}, z_i}[\log(1 - \mathcal{D}(\mathcal{G}_i(g_{i-1}, z_i)))] : i > 1. \tag{7.7}$$

Each GAN generates a portion of the complete spherical moment vector for the next GAN to be conditioned upon. The setup includes two major steps: (i) forward pass and (ii) backward pass. Accordingly, the overall architecture can be decomposed into two sets of generators $\mathcal{G}_f$ and $\mathcal{G}_b$, that implement the forward and backward functions, respectively. In the forward pass, the model tries to generate a coarse shape representation, and the backward pass refines the coarse representation to generate a refined representation. The basis of our design is the *Markovian assumption*, i.e., given the outputs from the neighbouring generators, a current generator is independent from the outputs of the rest. We describe the two generation steps in Sec. 7.3.2.1 and 7.3.2.2.

7.3.2.1 Forward pass

In the forward pass, we have a set of T' generative models $\mathcal{G}_f = \{\mathcal{G}_1, \ldots, \mathcal{G}_{T'}\}$, which work in unison to generate a coarse representation of a 3D shape. Each $\mathcal{G}_i \in \{\mathcal{G}_2, \ldots \mathcal{G}_{T'}\}$ is conditioned upon the outputs of $\mathcal{G}_{i-1}$, and generates a predefined frequency band ($\mathcal{S}_i$) of the complete spherical harmonic representation ($\mathcal{S}$) of the corresponding 3D shape. It is worthwhile to note that the forward pass is sufficient to generate the complete SMV without the aid of a backward pass. However, a critical limitation of this setup is that each GAN is only conditioned upon lower frequency bands of the SMV. In practice, this results in noisy outputs. Therefore, we also perform a backward pass, which allows the GANs to refine the generation by observing the higher frequencies. This procedure is explained on Sec. 7.3.2.2.

7.3.2.2 Backward pass

As explained in Sec. 7.3.2.1, the aim of the backward pass is to generate a more refined SMV, which produces a more refined 3D shape. Similar to forward pass, the backward pass is implemented using another set of generators $\mathcal{G}_b = \{\mathcal{G}_{T'+1}, \ldots \mathcal{G}_T\}$, where

$T = 2T' - 1$. Each $\mathcal{G}_i \in \mathcal{G}_b$ is conditioned upon the outputs of $\mathcal{G}_{i-1}$ and generates a specific portion of the complete SMV. In the training phase, we first transfer the trained weights from $\{\mathcal{G}_f \backslash \mathcal{G}_{T'}\}$ to $\mathcal{G}_b$, before training $\{\mathcal{G}_b\}$. Therefore, this can be intuitively considered as fine-tuning $\{\mathcal{G}_1 \dots \mathcal{G}_{T'-1}\}$ based on higher frequencies. The training procedure is explained in Sec. 7.5.

7.4 Spatial domain regularizer

Since SMVs are highly structured, each element of a particular SMV is crucial for accurate reconstruction of its corresponding 3D point-cloud. In other words, even slight variations of a particular SMV cause significant variations in the spatial domain. Therefore, it is cumbersome for a GAN to synthesize SMVs, corresponding to visually pleasing point-clouds, by solely observing a distribution of ground truth SMVs.

To surmount this barrier, we use a spatial domain regularizer that can refine the weights of our cascaded GAN architecture, in order to synthesize more plausible SMVs. The spatial domain regularizer provides feedback from the spatial domain to the GANs, depending on the quality of the spatial reconstruction. Firstly, we employ a pre-trained PointNet [Qi et al., 2017a] model on the reconstructed synthetic point-cloud, and extract a global feature. Secondly, using the same procedure, we obtain another global feature from a ground truth point-cloud from the same class, and compute the L_2 distance between these two features. Finally, using back back-propagation, we update the weights of all the generators $\boldsymbol{G} = \{\mathcal{G}_f \cup \mathcal{G}_b\}$ to minimize the L_2 distance. The final architecture of the proposed model is shown in Fig. 7.2.

In order to back-propagate error signals from the spatial domain to the spectral domain, we require $\partial\mathcal{L}/\partial\boldsymbol{g}$, where $\boldsymbol{g}$ is the SMV and $\mathcal{L}$ is the loss. To this end, we derive the following formula: let $\boldsymbol{g} = (g_0^0, \dots, g_l^m, \dots g_K^K)^\top$ be the SMV of a particular instance and $\{r(\theta_0, \phi_0), \dots, r(\theta_n, \phi_n), \dots, r(\theta_N, \phi_N)\}$ be the corresponding reconstructed points on $\mathbb{S}^2$ for the same instance. Then, using the chain rule it can be shown that,

$$\frac{\partial\mathcal{L}}{\partial g_l^m} = \sum_\theta \sum_\phi \frac{\partial\mathcal{L}}{\partial r(\theta,\phi)} \frac{\partial r(\theta,\phi)}{\partial g_l^m}, \tag{7.8}$$

$$\text{where, } r(\theta,\phi) = \sum_{l=0}^{M} \sum_{m=-l}^{l} g_l^m Y_l^m(\theta,\phi). \tag{7.9}$$

Combining Eq. 7.8 and 7.9, we obtain,

$$\frac{\partial\mathcal{L}}{\partial g_l^m} = \sum_\theta \sum_\phi \frac{\partial\mathcal{L}}{\partial r(\theta,\phi)} Y_l^m(\theta,\phi). \tag{7.10}$$

The above expression can be written as a matrix-vector product to obtain derivatives $\partial\mathcal{L}/\partial\boldsymbol{g}$. This makes the transformer a fully differentiable and a network-agnostic module which can be used to communicate between spectral and spatial domains.

7.5 Network architecture and training

Our aim is to generate a compact spectral representation, i.e., , a vector, instead of a irregular point set. In the spatial domain, points are correlated across the spatial space, and convolutions can be adopted to capture these dependencies. In fact, convolution kernels extract local features, under the assumption that spatially closer data points form useful local features. In contrast, closer elements in spectral domain representations do not necessarily exhibit strong correlations. Therefore, convolutional layers fail to excel in this scenario and thus, we opt for fully connected (FC) layers in designing our GANs. Interestingly, however, our GANs learn to generate quality outputs with a low depth architecture.

Generator architecture: For our main experiments, we choose the maximum degree of SMVs and the number of GANs as M=100 and T=7, respectively, where $\mathcal{G}_f = \{\mathcal{G}_1, .., \mathcal{G}_4\}$ and $\mathcal{G}_b = \{\mathcal{G}_5, \mathcal{G}_6, \mathcal{G}_7\}$. Each generator in $\mathcal{G}_f$ respectively generates frequency bands $(0 \leq l \leq 50, -l \leq m \leq 0)$, $(0 \leq l \leq 50, 0 < m \leq l)$, $(50 < l \leq 100, -l \leq m \leq 0)$ and $(50 < l \leq 100, 0 < m \leq l)$. Since $\mathcal{G}_5, \mathcal{G}_6, \mathcal{G}_7$ are used to fine tune $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$, they generate the same frequency portions as the latter set. For all the generators, we use the same architecture, except for the last FC layer. Each generator consists of three FC layers, first two layers with 512 neurons each, and the number of neurons in the last layer depends on the output size. For the first two layers, we use ReLU activation and the final layer has a *tanh* activation.

Training: The input to each of our generators, except to $\mathcal{G}_1$, is a 300-d vector: a 200-d noise vector concatenated with a 100-d vector sampled in equal intervals from the previous generator output. For $\mathcal{G}_1$, we use a 200-d noise input. We use RMSprop as the optimization algorithm with ρ=0.9, momentum=0, ϵ=10^{-7}, where symbols refer to usual notation. For $\mathcal{G}_f$ and $\mathcal{G}_b$, we use learning rates 0.001 and 0.0001 respectively, and for discriminators, we use a learning rate 10^{-5}. While training, we use three discriminator updates per each generator update. Our sampling procedure is explained in supplementary materials and the training scheme is illustrated in Algorithm 2.

7.6 3D reconstruction from single image

As a different application, we propose a generative model which can reconstruct 3D objects by observing a single RGB image. The model follows the network architecture proposed in Sec. 7.5, with a few alterations. Instead of randomly choosing the latent vector z, we use a set of image encoders to obtain an object representative vector $\hat{z}$, by taking a 2D image as the input. We use the same image encoder proposed in [Wu et al., 2015], which consists of five spatial convolution layers with kernel size $\{11, 5, 5, 5, 8\}$ with strides $\{4, 2, 2, 2, 1\}$. We use batch normalization after each layer, and ReLu activation as the non-linearity.

We use T' such image encoders for each $\mathcal{G}_i \in \mathcal{G}_f$, and use the same $\hat{z}$ vectors generated for $\{\mathcal{G}_1, \ldots, \mathcal{G}_{T'-1}\}$ when training $\mathcal{G}_i \in \mathcal{G}_b$. Each image encoder is trained end-to-end with $\mathcal{G}_i \in \mathcal{G}_f$. The training procedure is similar to Algorithm 2, although

Algorithm 2: Training procedure for the Spectral-GAN.

$\boldsymbol{G} = \{\mathcal{G}_f \cup \mathcal{G}_b\}$;
R_o = A set of samples from ground truth point-clouds;
for *i iterations* **do**
  for *each* $\mathcal{G}_k \in \mathcal{G}_f$ **do**
    for *j iterations* **do**
      Train $\mathcal{G}_k$;
  $\mathcal{G}_b \xleftarrow{\text{Weights}} \{\mathcal{G}_1, \ldots \mathcal{G}_{T'-1}\}$;
  for *each* $\mathcal{G}_k \in \mathcal{G}_b$ **do**
    for *j iterations* **do**
      Train $\mathcal{G}_k$;
  for *p iterations* **do**
    $\boldsymbol{g} \xleftarrow{\text{Synthesize}} \{\mathcal{G}_{T'} \cup \mathcal{G}_b\}$;
    $\boldsymbol{r}_g \leftarrow \text{Reconstruct}(\boldsymbol{g})$;
    $f_g \leftarrow \text{PointNet}(r_g)$;
    $f_o \leftarrow \text{PointNet}(r_o \sim R_o)$;
    $L \leftarrow \left\| f_g - f_o \right\|_2$;
    $\boldsymbol{G} \leftarrow \text{Update}(\boldsymbol{G}, L)$;

we use different loss functions in this case. To optimize the GANs in spectral domain, we use two loss components: an adversarial loss $\mathcal{L}_{ad}$ and a spectral reconstruction loss $\mathcal{L}_{sr}$. The final spectral domain loss $L_{spectral}$ is,

$$L_{spectral} = L_{ad} + \alpha L_{sr}, \tag{7.11}$$

where L_{sr} is the L_2 distance between the ground-truth SMV and the generated SMV from $\mathcal{G}'_T \cup \mathcal{G}_b$ and $\mathcal{L}_{ad}$ is given as,

$$\mathcal{L}_{ad} = \log \mathcal{D}(x) + \log(1 - \mathcal{D}(\mathcal{G}(\mathcal{E}(y)))) \tag{7.12}$$

Here, $\mathcal{E}(\cdot)$ is the encoder function, $\mathcal{D}(\cdot)$, $\mathcal{G}(\cdot)$ and y are discriminator function, generator function and image input, respectively. α is a scalar weight. For the spatial domain optimization, we replace spatial regularization loss with the Chamfer distance as follows:

$$L_{spatial} = \sum_{u \in S_1} \min_{v \in S_2} \|u - v\|_2^2 + \sum_{v \in S_2} \min_{u \in S_1} \|u - v\|_2^2, \tag{7.13}$$

where S_1 and S_2 are ground-truth and generated point sets, respectively. First, we obtain S_2 by converting the SMV to a point-cloud using Eq. 7.4 and then compute the loss (Eq. 7.13).

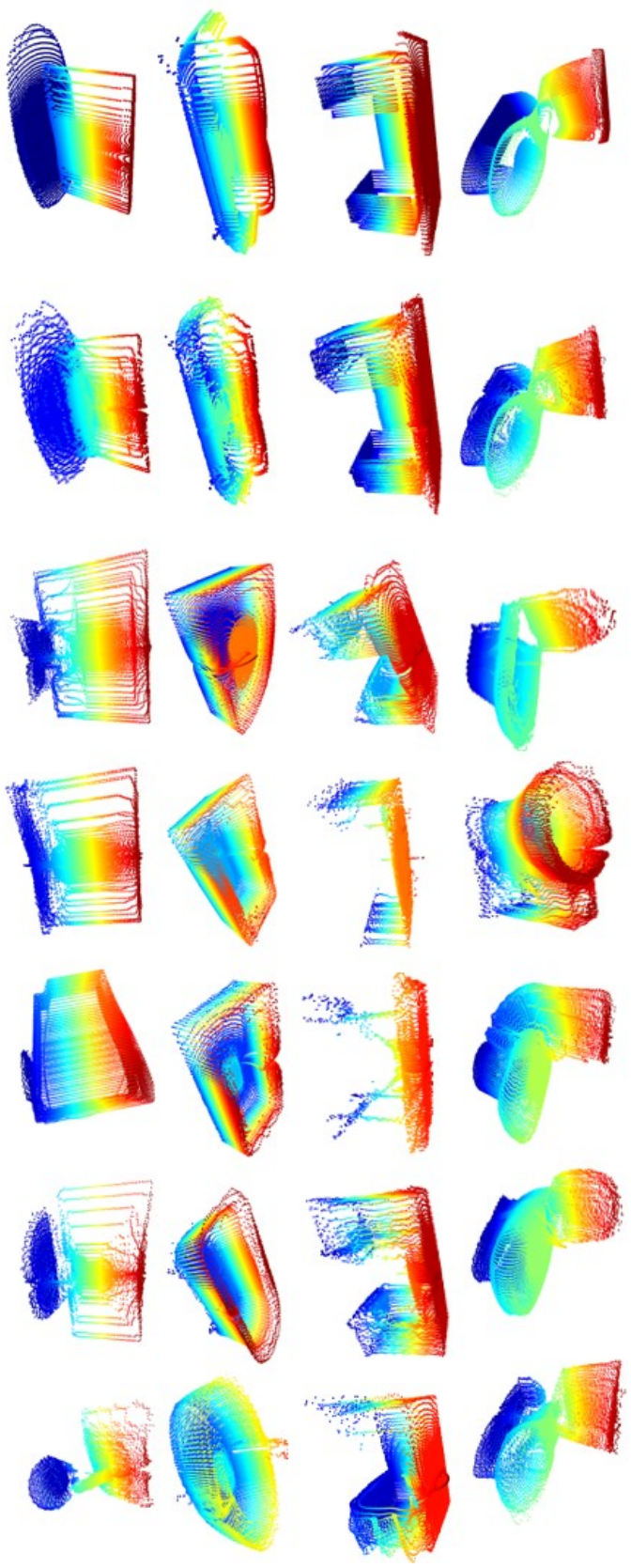


Figure 7.3: Qualitative analysis of the results. From the left, 1^{st} column: Ground truth, 2^{nd} column: ground truth point-clouds reconstructed by SMV, $3^{rd} - 7^{th}$ columns: generated samples using spectral GAN.

Table 7.1: 3D shape classification results on ModelNet10.

Method	Type	Accuracy
3D-ShapeNet (CVPR'15) [Wu et al., 2015]	Supervised	93.5%
EC-CNNs (CVPR'17) [Simonovsky and Komodakis, 2017]	Supervised	90.0%
Kd-Network (ICCV'17) [Klokov and Lempitsky, 2017]	Supervised	93.5%
LightNet (3DOR'17) [Zhi et al., 2017]	Supervised	93.4%
SO-Net (CVPR'18) [Li et al., 2018b]	Supervised	95.5%
Light Filed Descriptor [Chen et al., 2003]	Unsupervised	79.9%
Vconv-DAE (ECCV'16) [Sharma et al., 2016]	Unsupervised	80.5%
3D-GAN (NIPS'16) [Wu et al., 2016]	Unsupervised	91.0%
3D-DesNet (CVPR'18) [Xie et al., 2018a]	Unsupervised	92.4%
3D-WINN (AAAI'19) [Huang et al., 2019b]	Unsupervised	91.9%
PrimtiveGAN (CVPR'19) [Khan et al., 2019]	Unsupervised	92.2%
Spectral-GAN (ours)	Unsupervised	93.1%

7.7 Experiments

In this section, we evaluate our model both qualitatively and quantitatively, and develop useful insights.

7.7.1 3D shape generation

Qualitative results: We train our model for each category in ModelNet10 and show samples of generated 3D point-clouds in Fig. 7.3. As expected, the reconstruction from SMV adds some noise to the ground truth point-clouds. An interesting observation, however, is that the quality of generated point-clouds are not far from from the reconstructed point-clouds from the ground-truth. Since the network only consumes the reconstructed ground-truth, this observation highlights the ability of our network in accurate modeling of input data distributions.

Table 7.2: Inception scores (IS) for 3D shape generation. We only compare with voxel based methods since no point-cloud (p-cloud) based method reports IS.

Method	3D Data	Accuracy
3D-ShapeNet [Wu et al., 2015] (CVPR'15)	voxel	4.13 ± 0.19
3D-VAE [Kingma and Welling, 2013] (ICLR'15)	voxel	11.02 ± 0.42
3D-GAN [Wu et al., 2016] (NIPS'16)	voxel	8.66 ± 0.45
3D-DesNet [Xie et al., 2018a] (CVPR'18)	voxel	11.77 ± 0.42
3D-WINN [Huang et al., 2019b] (AAAI'19)	voxel	8.81 ± 0.18
PrimitiveGAN [Khan et al., 2019] (CVPR'19)	voxel	11.52 ± 0.33
Spectral-GAN (ours)	p-cloud	11.58 ± 0.08

Table 7.3: FID scores for 3D shape generation. (*lower is better*) All the methods except ours are voxel based.

Method	Dresser	Toilet	Stand	Chair	Table	Sofa	Monitor	Bed	Bathtub	Desk
3D-GAN [Wu et al., 2016] (NIPS'16)	-	-	-	469	-	517	-	-	-	651
3D-DesNet [Xie et al., 2018a] (CVPR'18)	414	662	517	490	538	494	511	574	-	-
3D-WINN [Huang et al., 2019b] (AAAI'19)	305	474	456	225	220	151	181	222	305	322
Spectral-GAN (ours)	462	195	452	472	522	180	192	230	208	354

Quantitative analysis: To assess the proposed approach quantitatively, we compare the Inception Score (IS) of our network with other voxel-based generative models in Tab. 7.2. In this experiment, we use Qi et al. [2016] as the reference network. IS evaluates a model in terms of both quality and diversity of the generated shapes. Overall, our model demonstrates the second highest performance with a score of 11.58. To the best of our knowledge, our work is the first 3D point-cloud GAN to report IS.

We further evaluate our model using Frechet Inception Distance (FID) proposed by Heusel et al. [2017], and compare with state-of-the-art. IS does not always coincide with human judgement regarding the quality of the generated shapes, as it does not directly capture the similarity between the synthetic and generated shapes. Therefore, FID is used as a complementary measure to evaluate GAN performance. Huang et al. [2019b] were the first to incorporate FID to 3D GANs, and following them, we also use Qi et al. [2016] as the reference network. As evident from Table 7.3, our results are on-par with state-of-the-art, getting highest scores in three categories: toilet, night stand and bath tub. Interestingly, our Spectral-GAN generally performs better with objects that have curved boundaries, which is a favorable characteristic, as curved boundaries are generally difficult to generate in Euclidean spaces. Note that we convert the point-clouds to meshes before evaluating with both IS and FID.

Comparison with point-cloud generation approaches: We use two metrics proposed in Achlioptas et al. [2017a] (i.e., MMD-CD, MMD-ED) to compare the performance of the proposed architecture with other point-cloud generation methods, and display the results in Table 7.4. In this experiment, we use 16 classes of ShapeNet [Yi et al., 2016]. As shown, our network gives best results. Intuitively, this suggests that shapes generated by our network have high fidelity compared to the test set.

Scalability to high resolutions: A favorable attribute of our network design is the ability to scale to higher resolutions with minimal changes to the architecture. To verify this, we vary the degree of SMV, and train our model separately for each case. Since the number of points n is tied to the maximum degree M of SMVs as $n{=}4M^2$, we obtain samples with different resolutions for each case (see Fig. 7.4). A key point here is that we only change the output layer size of the generator (according to the length of SMV) to generate point-clouds with different resolutions. Fig. 7.5 illustrates

Table 7.4: Comparison with point-cloud generative models.

Method	Class	MMD-CD	MMD-EMD
r-GAN (dense) [Achlioptas et al., 2017a]	Chair	0.0029	0.136
r-GAN (conv) [Achlioptas et al., 2017a]		0.0030	0.223
Valsesia et al. [2018] (no up.)		0.0033	0.104
Valsesia et al. [2018] (up.)		0.0029	0.097
TreeGAN [Shu et al., 2019]		0.0016	0.101
Spectral-GAN (ours)		0.0012	0.080
r-GAN (dense) [Achlioptas et al., 2017a]	Airplane	0.0009	0.094
r-GAN (conv) [Achlioptas et al., 2017a]		0.0008	0.101
Valsesia et al. [2018] (no up.)		0.0010	0.102
Valsesia et al. [2018] (up.)		0.0008	0.071
TreeGAN [Shu et al., 2019]		0.0004	0.068
Spectral-GAN (ours)		0.0002	0.057
r-GAN (dense) [Achlioptas et al., 2017a]	Sofa	0.0020	0.146
r-GAN (conv) [Achlioptas et al., 2017a]		0.0025	0.110
Valsesia et al. [2018] (no up.)		0.0024	0.094
Valsesia et al. [2018] (up.)		0.0020	0.083
Spectral-GAN (ours)		0.0020	0.080
r-GAN (dense) [Achlioptas et al., 2017a]	All classes	0.0021	0.155
TreeGAN [Shu et al., 2019]		0.0018	0.107
Spectral-GAN (w/o backward pass)		0.0020	0.127
Spectral-GAN (ours)		0.0015	0.097

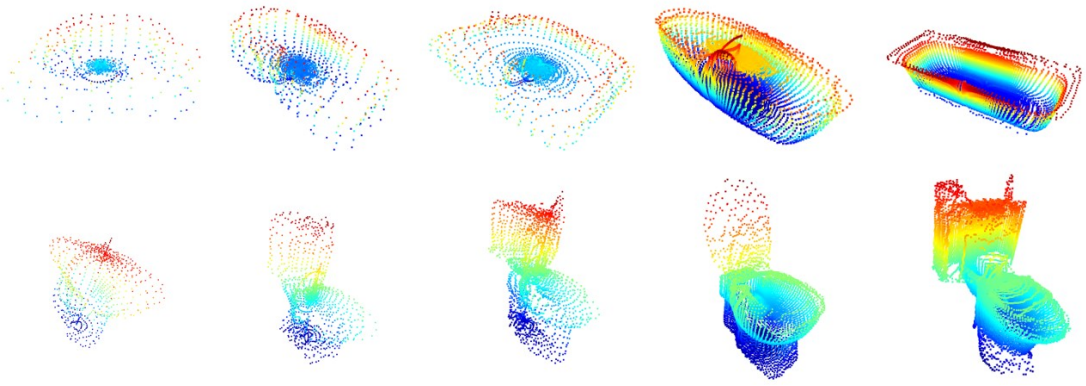


Figure 7.4: Scalability of the proposed network with resolution. We obtain increasingly dense resolution by only changing the output layer size in each training phase. Number of points from the left: $30^2, 60^2, 100^2, 150^2$ and 200^2

.

the variation of resolution with the number of FLOPs. Remarkably, we are able to generate high-resolution outputs up to $40,000$ points with only $0.3B$ FLOPs. Another intriguing observation is that our network is able to increase the output resolution by a factor of 40, while the number of FLOPs is only increased by a factor around 4.

Usefulness of backward pass: Fig. 7.6 illustrates the effect of performing a backward pass. As shown, the forward pass only generates a coarse representation of the shapes without fine details. This is anticipated, since cascaded GANs can only observe the lower frequency portions of SMV in the forward pass. In contrast, the backward pass observes the higher frequency portions, and fine tunes the coarse representation by adding complementary details.

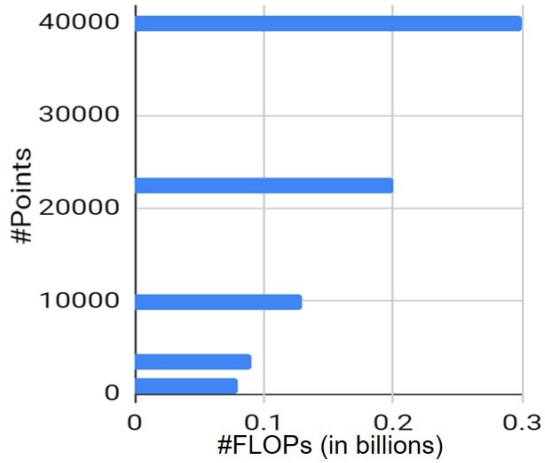


Figure 7.5: Spectral GAN can generate high-resolution outputs with minimal computational overhead. We increase resolution approximately $40\times$ while only an increase of $4\times$ FLOPs.

7.7.2 Unsupervised 3D Representation Learning

In this section, we evaluate the representation learning capacity of our discriminator. To this end, we pass relevant SMV frequency bands of 3D point-clouds through trained discriminators, extract the features from the third FC layer, and finally concatenate them to create a feature vector. This feature vector is then fed through a binary SVM classifier and the classification results are obtained as one-against-the-rest. The classification results on ModelNet10 are depicted in Table 7.1. As evident, we achieve the highest result with a value of 93.1%, which highlights the excellent representation learning capacity of our discriminators.

7.7.3 3D reconstruction results

In this section, we evaluate the performance of the 3D reconstruction network proposed in Sec. 7.6. First, we randomly apply a rotation $R = (R_x, R_y, R_z)$ to each

3D model from the IKEA dataset 15 times, and render the rotated model in front of background images obtained from Xiao et al. [2010]. Afterwards, we save the rendered images and the corresponding 3D models to create ground-truth image-3D model pairs. The ground truth 3D-models are manually aligned using the Iterative closest point (ICP) algorithm. While applying rotations, we set the constraints $-\frac{\pi}{6} < R_x, R_y < \frac{\pi}{6}$ and $-\pi < R_z < \pi$ and crop the rendered images for the object to be in the center of the images. For the test set, we use the original images provided in the IKEA dataset and test our network on four object classes: chair, sofa, table and bed. We train our model separately for each category and use mean average precision (mAP) to evaluate the performance. In evaluation, we voxelize the generated and ground truth point-clouds using a $20\times20\times20$ voxel grid, and obtain average precision for voxel prediction. The results and illustrative examples are shown in Table 7.5 and Fig. 7.7, respectively. As depicted, we surpass state-of-the-art results in sofa and bed categories, while achieving second best results in the table category.

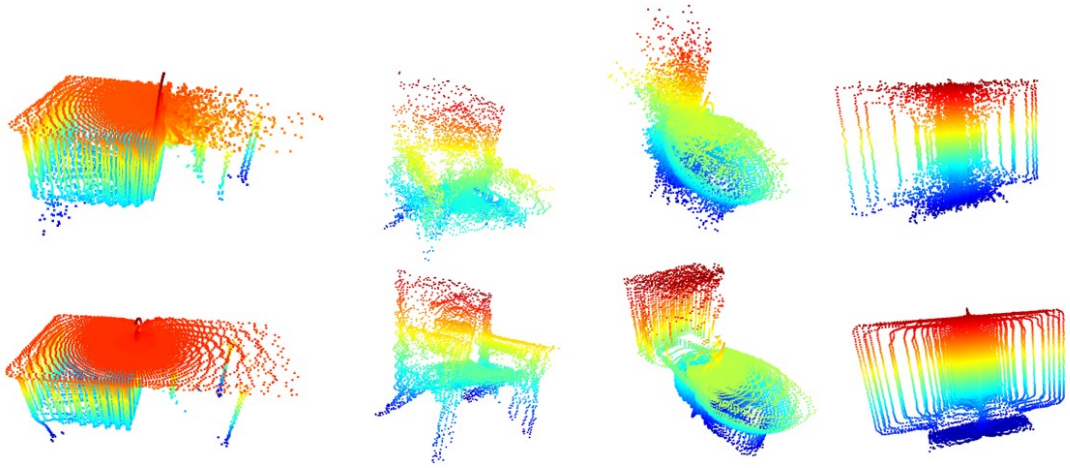


Figure 7.6: Effect of backward pass. Top row: samples generated using only forward pass. Bottom row: same samples after passing through both forward and backward pass. Backward pass refines the image by adding more fine details.

7.8 Sampling and reconstruction

A key attribute of any sampling theorem is the minimum number of sample points required to accurately represent a band-limited function in a particular space. Several such sampling theorems have been proposed to represent a signal with finite energy in S^2, whereas a most popular choice is the Driscoll and Healy's (DH) theorem proposed by Driscoll and Healy [1994], which we also use in our work.

According to DH theorem, to accurately represent a signal on S^2 using spherical harmonic moments band-limited at degree M, $4M^2$ equiangular sampled points are needed. For all the main experiments in this work, we choose $M = 100$ and obtain

Figure 7.7: Qualitative results for 3D point-cloud reconstruction from a single image.

Table 7.5: Average precision for 3D point-cloud reconstruction from single image. The point-clouds are voxelized before obtaining the score.

Method	Chair	Sofa	Bed	Table
AlexNet-fc8 [Girdhar et al., 2016]	20.4	38.8	29.5	16.0
AlexNet-conv4 [Girdhar et al., 2016]	31.4	69.3	38.2	19.1
T-L network [Girdhar et al., 2016]	32.9	71.7	56.3	23.3
3D-VAE-GAN [Wu et al., 2016]	47.2	78.8	63.2	42.3
VAE-IWGAN [Smith and Meger, 2017]	49.3	68.0	65.7	52.2
PrimtiveGAN [Khan et al., 2019]	47.5	77.1	68.4	60.0
Spectral-GAN (ours)	42.3	81.2	71.4	48.3

an equally sampled 200×200 grid in each θ and ϕ directions, where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. However, as mentioned in Sec. 7.4, spherical harmonics can represent only polar 3D shapes, which can result in less visually pleasing spatial representations of non-polar shapes. To overcome this obstacle, we follow the following sampling procedure.

First, we scale the 3D mesh to fit inside the unit ball $\mathbb{B}^3$, and cast rays from the centroid of the shape to outward direction, and take the first hit locations of the rays with a face as a sample point. In the first stage, we sample 200×100 such equiangular points in a 200×100 grid, sampled in θ and ϕ directions respectively, where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. In the second stage, we rotate the casted rays in ϕ direction, by an amount of $\frac{\pi}{99}$, and obtain the last hit locations of the each ray with a face of the 3d shape as a sample point. Union of these two sampling sets provide a more visually pleasing point-cloud for non-polar 3D shapes. This procedure is illustrated in Fig. 7.8.

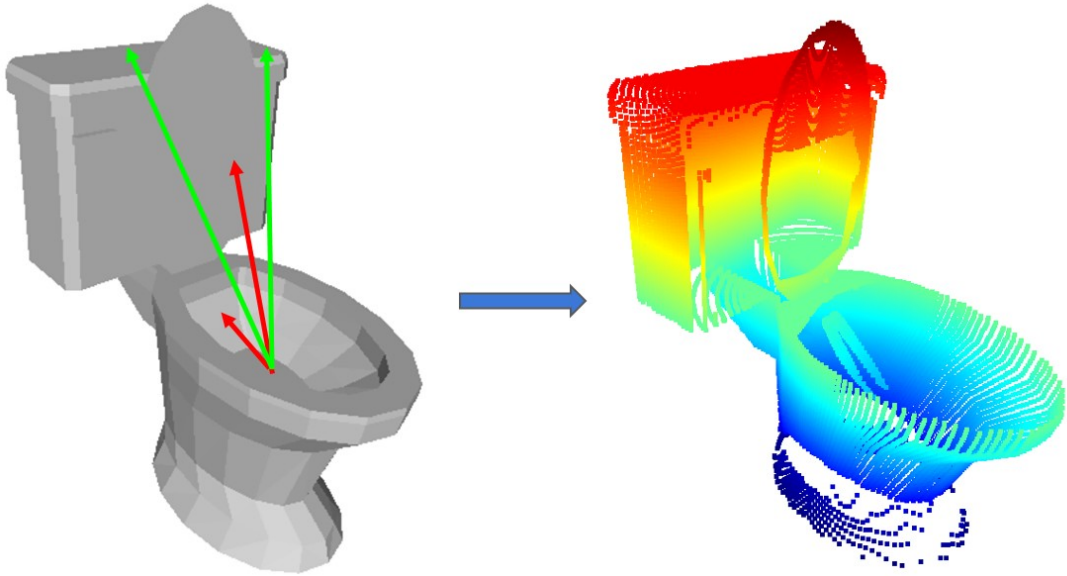


Figure 7.8: Illustration of the sampling procedure. Red arrows and green arrows demonstrate first stage and second stage sampling, respectively.

7.9 Literature on cascaded generative designs

Denton et al. [2015] proposed a cascaded GAN architecture for 2D image generation. Similar to our work, they also use a series of conditional GANs which are conditioned upon one another. These GANs generate image representations in a Laplacian pyramid framework to create increasingly refined images. Instead of generating images directly in the spatial domain, these generative models specialize in generating a specific residual image, according to the corresponding stage of the Laplacian pyramid, which are finally combined together to produce a high quality image. This is analogous to

our work, where our generators generate a specific frequency portion of SMVs, which are finally combined together to obtain the full representation. Other recent works also employ cascaded generative architectures to improve image quality e.g., Wang et al. [2018] use a combination of generators operating on low and high resolution domains, Wang and Gupta [2016] separately train generative models to learn style and structure components, Zhang et al. [2017b] progressively adds photorealistic details in low-resolution generated images. The conditional stacked GAN architecture of Huang et al. [2017b] is particularly close to ours, that feeds onto previous generators output and new latent vectors to create novel images. Finally, the seminal SinGAN [Shaham et al., 2019] approach designs a pyramid of coarse-to-fine generators that can be trained on a single image. However, as opposed to current work, all above efforts operate in the spatial domain and have no concrete definition of spectral bands.

7.10 Computational complexity analysis

A key feature of our network is its high computational efficiency despite being a cascaded design. Since the target is a 1-D structured vector, the generators are allowed to have a shallow architecture, which decreases the total number of FLOPs during operation. Table 7.6 compares the our model complexity against the state-of-the-art models. We achieve the best performance in terms of MMD-CD and MMD-EMD while having the lowest model complexity. Experiments are conducted for inference with 20 batch size.

7.11 Chapter summary

This chapter proposes a generative model for 3D point-clouds that operates in the spectral-domain. A key feature of our model is the use of structured representations, in the form of a set of cascaded GANs. Each GAN is an expert in generating the representations that correspond to a specific frequency band, and are allowed to communicate with other GANs to refine the outputs. We show that this modular approach is highly effective and can be scaled up easily. In contrast to previous methods that operate in the spatial-domain, our approach provides an inherent way to deal with the inherent redundancy and irregularity of point-clouds. We demonstrate that our model generates sound 3D outputs, can scale to high-dimensional outputs and learns discriminative features in an unsupervised manner. Further, it can be used for 3D reconstruction tasks with minimal changes to the architecture.

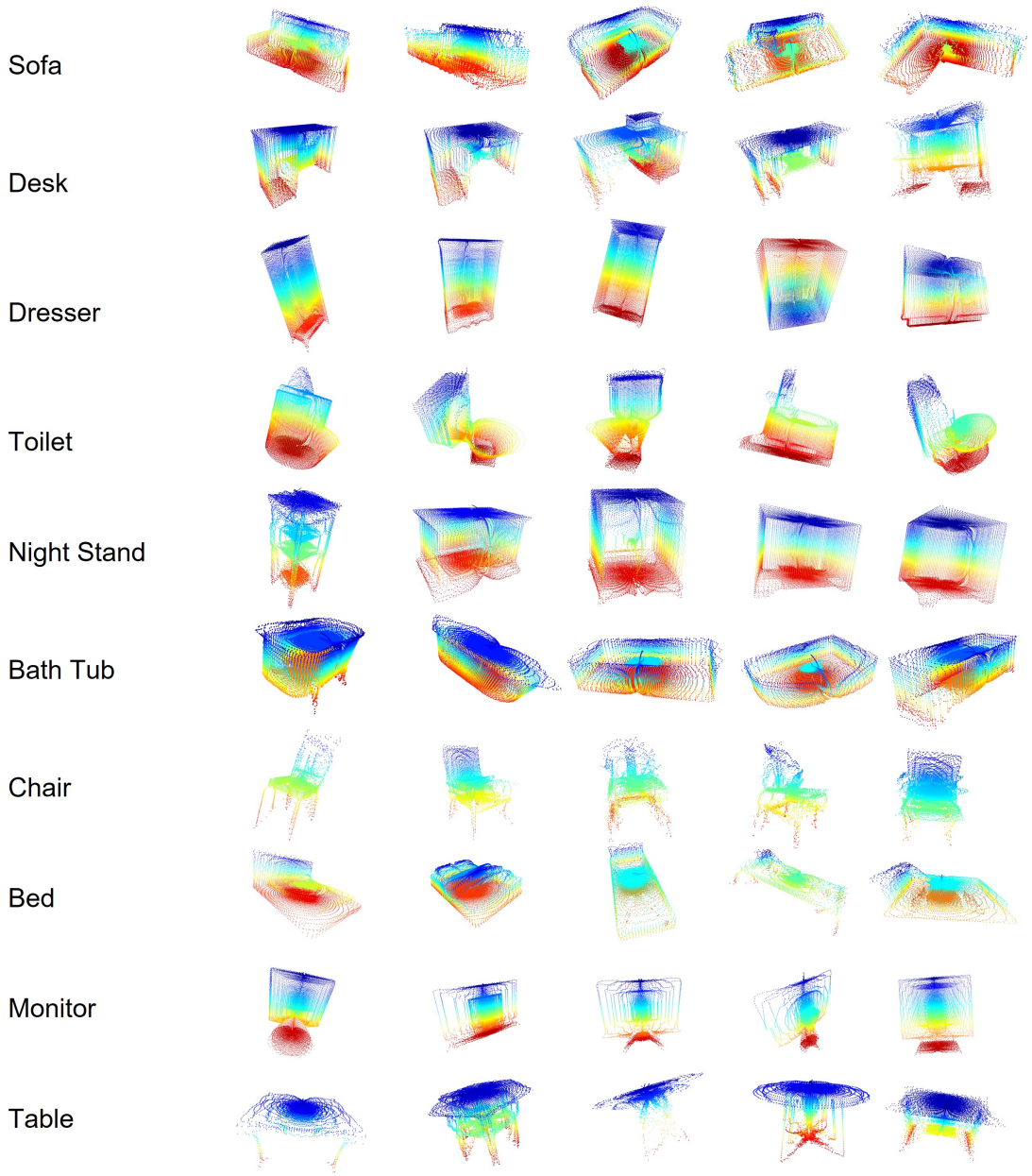


Figure 7.9: Qualitative results: generated point clouds for each class.

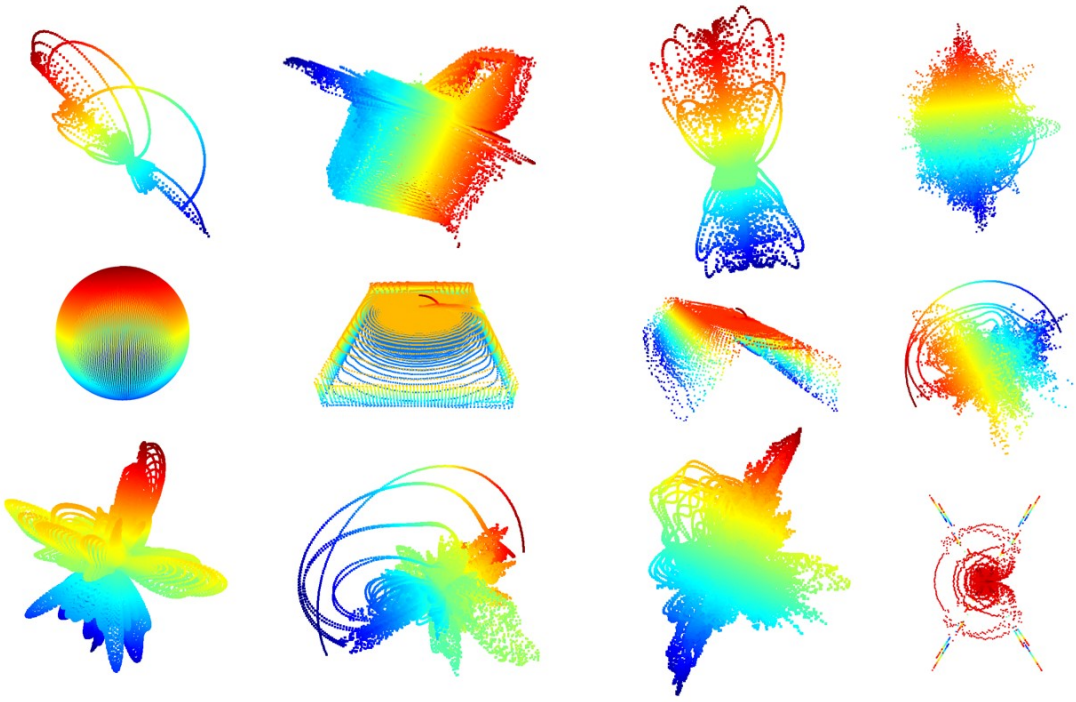


Figure 7.10: Our network tends to generate weird artifacts among plausible samples, when trained without the spatial domain regularizer, since small variations in spectral domain cause significant variations in spatial domain. A few such examples are illustrated here. These artifacts are effectively suppressed by our spatial domain regularizer.

Table 7.6: Model complexity comparison with point-cloud generative models (inference). We achieve the best performance while having the lowest complexity. (↓ denotes lower is better, ↑ denotes higher is better)

Method		MMD-CD (↓)	MMD-EMD (↓)	#FLOPs (↓)	#Points (↑)
r-GAN (dense) [Achlioptas et al., 2017a]	Chair	0.0029	0.136	0.1B	2048
Valsesia et al. [2018] (up.)		0.0029	0.097	304B	2048
Spectral-GAN (ours)		0.0012	0.080	0.09B	3600
r-GAN (dense) [Achlioptas et al., 2017a]	Airplane	0.0009	0.094	0.1B	2048
Valsesia et al. [2018] (up.)		0.0008	0.071	304B	2048
Spectral-GAN (ours)		0.0002	0.057	0.09B	3600
r-GAN (dense) [Achlioptas et al., 2017a]	Sofa	0.0020	0.146	0.1B	2048
Valsesia et al. [2018] (up.)		0.0020	0.083	304B	2048
Spectral-GAN (ours)		0.0020	0.080	0.09B	3600
r-GAN (dense) [Achlioptas et al., 2017a]	All classes	0.0021	0.155	0.1B	2048
Spectral-GAN (ours)		0.0015	0.097	0.09B	3600

Chapter 8

Conditional Generative Modeling via Learning the Latent Space

In the last chapter, we proposed a novel modular approach to synthesize 3D point cloud data, motivated by combinatorial generalization of the human brain. In this chapter, we show that this concept can be extended to the task of 2D image generation. Specifically, we decouple the image generation task into smaller modules, and propose a simple, novel conditional image generation mechanism that can outperform even sophisticated GAN models.

Conditional generative models provide a natural mechanism to jointly learn a data distribution and optimize predictions. In contrast, discriminative models improve predictions by modeling the label distribution. Learning to model the data distribution allows generating novel samples and is considered a preferred way to understand the real world. Existing conditional generative models have generally been explored in single-modal settings, where a one-to-one mapping between input and output domains exists [Nalisnick et al., 2019; Fetaya et al., 2020]. Here, we investigate continuous multimodal (CMM) spaces for generative modeling, where one-to-many mappings exist between input and output domains. This is critical since many real world situations are inherently multi-modal, e.g., humans can imagine several outcomes for a given occluded image.

In a discrete setting, this problem becomes relatively easy to tackle using techniques such as maximum-likelihood-estimation, since the output can be predicted as a vector [Zhang et al., 2016], which is not possible in continuous domains. One way to model CMM spaces is by using variational inference, e.g., variational autoencoders (VAE) [Kingma and Welling, 2013]. However, the approximated posterior distribution of VAEs are often restricted to the Gaussian family, which hinders its ability to model more complex distributions. As a solution, Maaløe et al. [2016] suggested using auxiliary variables to improve the variational distribution. To this end, the latent variables are hierarchically correlated through injected auxiliary variables, which can produce non-Gaussian distributions. A slightly similar work by Rezende and Mohamed [2015] proposed to perform variational inference using normalizing flows, that can hierarchically generate more complex probability distributions by applying a series of bijective mappings to an original simpler distribution. Recently, Chang et al.

[2019] proposed a model, where a separate variable can be used to vary the impact of different loss components at inference, which allows diverse outputs.

In addition to the aforesaid methods, in order to model CMM spaces, a prominent approach in the literature is to use a combination of reconstruction and adversarial losses [Isola et al., 2017; Zhang et al., 2016; Pathak et al., 2016a]. However, this entails key shortcomings. 1) The goals of adversarial and reconstruction losses are contradictory (Sec. 8.3), hence model engineering and numerous regularizers are required to support convergence [Lee et al., 2019b; Mao et al., 2019], thereby resulting in less-generic models tailored for specific applications [Zeng et al., 2019b; Vitoria et al., 2020]. 2) The adversarial loss based models are notorious for difficult convergence due to the challenge of finding Nash equilibrium of a non-convex min-max game in high-dimensions [Barnett, 2018; Chu et al., 2020a; Kodali et al., 2017]. 3) The convergence is heavily dependent on the architecture, hence such models show lack of scalability [Thanh-Tung et al., 2019; Arora and Zhang, 2017]. 4) The promise of assisting downstream tasks remains challenging, with a large gap in performance between the generative modelling approaches and their discriminative counterparts [Grathwohl et al., 2020; Jing and Tian, 2020].

In this work, we propose a general-purpose framework—Conditional Generation by Modeling the Latent Space (cGML)—for modeling CMM spaces using a set of domain-agnostic regression cost functions instead of the adversarial loss. This improves both the stability and eliminates the incompatibility between the adversarial and reconstruction losses, allowing more precise outputs while maintaining diversity. The underlying notion is to learn the '*behaviour of the latent variables*' in minimizing these cost functions while converging to an optimum mode during the training phase, and mimicking the same at inference. Despite being a novel direction, the proposed framework showcases promising attributes by: (a) achieving state-of-the-art results on a diverse set of tasks using a generic model, implying generalizability, (b) rapid convergence to optimal modes despite architectural changes, (c) learning useful features for downstream tasks, and (d) producing diverse outputs via traversal through multiple output modes at inference.

8.1 Proposed Methodology

We define a family of cost functions $\{E_{i,j} = d(y^g_{i,j}, \mathcal{G}(x_j, w))\}$, where $x_j \sim \chi$ is the input, $y^g_{i,j} \sim \mathrm{Y}$ is the i^{th} ground-truth mode for x_j, $\mathcal{G}$ is a generator function with weights w, and $d(\cdot,\cdot)$ is a distance function. Note that the number of cost functions $E_{(\cdot,j)}$ for a given x_j can vary over χ. Our aim here is to come up with a generator function $\mathcal{G}(x_j, w)$, that can minimize each $E_{i,j}, \forall i$ as $\mathcal{G}(x_j, w) \rightarrow y^g_{i,j}$. However, since $\mathcal{G}$ is a deterministic function (x and w are both fixed at inference), it can only produce a single output. Therefore, we introduce a latent vector z to the generator function, that can be used to converge $\bar{y}_{i,j} = \mathcal{G}(x_j, w, z_{i,j})$ towards a $y^g_{(i,j)}$ at inference, and possibly,

to multiple solutions. Formally, the family of cost functions now becomes:

$$\{\hat{E}_{i,j} = d(y^g_{i,j}, \mathcal{G}(x_j, w, z_{i,j}))\}, \forall z_{i,j} \sim \zeta. \tag{8.1}$$

Then, our training objective can be defined as finding a set of optimal $z^*_i \in \zeta$ and $w^* \in \omega$ by minimizing $\mathbb{E}_{i \sim I}[\hat{E}_i]$, where I is the number of possible solutions for x_j. Note that w^* is fixed for all i and a different z^*_i exists for each i. Considering all the training samples $x_j \sim \chi$, our training objective becomes,

$$\{\{z^*_{i,j}\}, w^*\} = \underset{z_{i,j} \in \zeta, w \in \omega}{\arg\min}\, \mathbb{E}_{i \in I, j \in J}[\hat{E}_{i,j}]. \tag{8.2}$$

It can be shown that Eq. 8.2 can be optimized via Algorithm 3 as follows: Let us consider a particular input x_j and an associated ground truth $y^g_{i,j}$. Then, for this particular case, we denote our cost function to be $\hat{E}_{i,j} = d(w, z)$. Further, a family of cost functions can be defined as,

$$f_w(z) = d(w, z), \tag{8.3}$$

for each $w \sim \omega$. Further, let us consider an arbitrary initial setting (z_{init}, w_{init}). Then, with enough iterations, gradient descent by $\nabla_z f_w(z)$ converges z_{init} to,

$$\bar{z} = \arg \inf_{z \in \zeta} f_w. \tag{8.4}$$

Next, with enough iterations, gradient descent by $\nabla_w f_w(\bar{z})$ converges w to,

$$\bar{w} = \arg \inf_{w \in \omega} f_w(\bar{z}). \tag{8.5}$$

Observe that $f_{\bar{w}}(\bar{z}) \leq f_{w_{init}}$, where the equality occurs when $\nabla_z f_w(z) = \nabla_w f_w(\bar{z}) = 0$. If $f_w(z)$ has a unique global minima, repeating Equation 8.4 and 8.5 converges to that global minima, giving $\{z^*_{i,j}, w^*_{i,j}\}$. It is straight forward to see that using a small number of iterations (usually one in our case) for each sample set for Equation 8.5, i.e., stochastic gradient descent, gives us,

$$\{z^*_{i,j}, w^*\} = \underset{z_{i,j} \in \zeta, w \in \omega}{\arg\min}\, \mathbb{E}_{i \in I, j \in J}[\hat{E}_{i,j}], \tag{8.6}$$

where w^* is fixed for all samples and modes [Robbins, 2007]. Note that the proof is valid only for the convex case, and we rely on stochastic gradient descent to converge to at least a good local minima, as commonly done in many deep learning settings.

Intuitively, the goal of Eq. 8.2 is to obtain a family of optimal latent codes $\{z^*_{i,j}\}$, each causing a global minima in the corresponding $\hat{E}_{i,j}$ as $y^g_{i,j} = \mathcal{G}(x_j, w, z^*_{i,j})$. Consequently, at inference, we can optimize $\bar{y}_{i,j}$ to converge to an optimal mode in the output space by varying z. Therefore, we predict an estimated $\bar{z}_{i,j}$ at inference,

$$\bar{z}_{i,j} \approx \min_z \hat{E}_{i,j}, \tag{8.7}$$

for each $y^g_{i,j}$, which in turn can be used to obtain the prediction $\mathcal{G}(x_j, \bar{z}_{i,j}, w) \approx y^g_{i,j}$. In other words, for a selected x_j, let $\bar{y}^t_{i,j}$ be the initial estimate for $\bar{y}_{i,j}$. At inference, z can traverse gradually towards an optimum point $y^g_{i,j}$ in the space, forcing $\bar{y}^{t+n}_{i,j} \rightarrow y^g_{i,j}$, in finite steps ($n$).

However, still a critical problem exists: Eq. 8.7 depends on $y^g_{i,j}$, which is not available at inference. As a remedy, we enforce Lipschitz constraints on $\mathcal{G}$ over $(x_j, z_{i,j})$, which bounds the gradient norm as,

$$\frac{\left\|\mathcal{G}(x_j, w^*, z^*_{i,j}) - \mathcal{G}(x_j, w^*, z_0)\right\|}{\left\|z^*_{i,j} - z_0\right\|} \leq \int \left\|\nabla_z \mathcal{G}(x_j, w^*, \gamma(t))\right\| dt \leq C, \tag{8.8}$$

where $z_0 \sim \zeta$ is an arbitrary random initialization, C is a constant, and $\gamma(\cdot)$ is a straight path from z_0 to $z^*_{i,j}$.

Proof.

$$||\mathcal{G}(x_j, w^*, z^*_{i,j}) - \mathcal{G}(x_j, w^*, z_0)|| = || \int_{z_0}^{z^*_{i,j}} \nabla_z \mathcal{G}(x_j, w^*, z) dz|| \tag{8.9}$$

Let $\gamma(t)$ be a straight path from z_0 to $z^*_{i,j}$, where $\gamma(0) = z_0$ and $\gamma(1) = z^*_{i,j}$. Then,

$$= || \int_0^1 \nabla_z \mathcal{G}(x_j, w^*, \gamma(t)) \frac{d\gamma}{dt} dt|| \tag{8.10}$$

$$= || \int_0^1 \nabla_z \mathcal{G}(x_j, w^*, \gamma(t))(z^*_{i,j} - z^*_0) dt|| \tag{8.11}$$

$$= ||(z^*_{i,j} - z^*_0) \int_0^1 \nabla_z \mathcal{G}(x_j, w^*, \gamma(t)) dt|| \tag{8.12}$$

$$\leq \left\|(z^*_{i,j} - z^*_0)\right\| \left\| \int_0^1 \nabla_z \mathcal{G}(x_j, w^*, \gamma(t)) dt \right\| \tag{8.13}$$

On the other hand the Lipschitz constraint ensures,

$$\left\|\nabla_z \mathcal{G}(x_j, w^*, \gamma(t))\right\| \leq \lim_{\epsilon \to 0} \frac{\left\|\mathcal{G}(x_j, w^*, \gamma(t)) - \mathcal{G}(x_j, w^*, \gamma(t+\epsilon))\right\|}{\|z_t - z_{t+\epsilon}\|} \leq C, \tag{8.14}$$

where C is a constant. Combining Eq. 8.13 and 8.14 we get,

$$\frac{\left\|\mathcal{G}(x_j, w^*, z^*_{i,j}) - \mathcal{G}(x_j, w^*, z_0)\right\|}{\left\|z^*_{i,j} - z_0\right\|} \leq \int_0^1 \left\|\nabla_z \mathcal{G}(x_j, w^*, \gamma(t))\right\| dt \leq C. \tag{8.15}$$

□

Intuitively, Eq. 8.8 implies that the gradients $\nabla_z \mathcal{G}(x_j, w^*, z_0)$ along the path $\gamma(\cdot)$ do not tend to vanish or explode, hence, finding the path to optimal $z^*_{i,j}$ in the space ζ becomes a fairly straight forward regression problem. Moreover, enforcing the Lipschitz constraint encourages meaningful structuring of the latent space: suppose $z^*_{1,j}$ and $z^*_{2,j}$ are two optimal codes corresponding to two ground truth modes for a particular input. Since $\|z^*_{2,j} - z^*_{1,j}\|$ is lower bounded by $\frac{\left\|\mathcal{G}(x_j, w^*, z^*_{2,j}) - \mathcal{G}(x_j, w^*, z^*_{1,j})\right\|}{L}$, where L is the Lipschitz constant, the minimum distance between the two latent codes is proportional to the difference between the corresponding ground truth modes. In practice, we observed that this encourages the optimum latent codes to be placed sparsely, which helps a network to learn distinctive paths towards different modes (see Sec. 8.3.1).

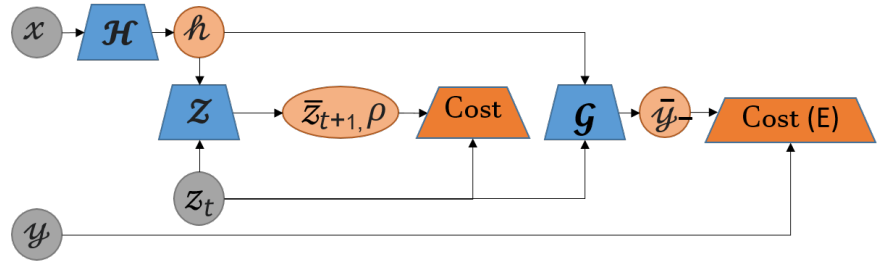
8.1.1 Convergence at inference

We formulate finding the convergence path of z at inference as a regression problem, i.e., $z_{t+1} = r(z_t, x_j)$. We implement $r(\cdot)$ as a recurrent neural network (RNN). The series of predicted values $\{z_{(t+k)} : k = 1, 2, .., N\}$ can be modeled as a first-order Markov chain requiring no memory for the RNN. We observe that enforcing Lipschitz continuity on $\mathcal{G}$ over z leads to smooth trajectories even in high dimensional settings, hence, memorizing more than one step in to the history is redundant. However, z_{t+1} is not a state variable, i.e., the existence of multiple modes for output prediction $\bar{y}$ leads to multiple possible solutions for z_{t+1}. On the contrary, $\mathbb{E}[z_{t+1}]$ is a state variable w.r.t. the state (z_t, x), which can be used as an approximation to reach the optimal z^* at inference. Therefore, instead of directly learning $r(\cdot)$, we learn a simplified version $r'(z_t, x) = \mathbb{E}[z_{t+1}]$.

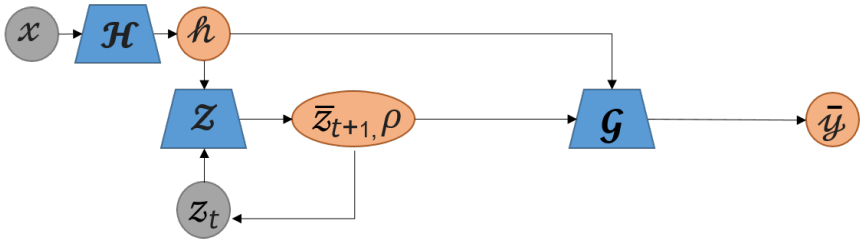
Intuitively, the whole process can be understood as observing the behavior of z on a smooth surface at the training stage, and predicting the movement at inference. A key aspect of $r'(z_t, x)$ is that the model is capable of converging to multiple possible optimum modes at inference based on the initial position of z.

8.1.2 Momentum as a supplementary aid

Based on Sec. 8.1.1, z can now traverse to an optimal position z^* during inference. However, there can exist rare symmetrical positions in the ζ where $\mathbb{E}[z_{t+1}] - z_t \approx 0$, although far away from $\{z^*\}$, forcing $z_{t+1} \approx z_t$. Simply, the above phenomenon can occur if some z_{t+1} has traveled in many non-orthogonal directions, so the vector addition of $z_{t+1} \approx 0$. This can *fool* the system to falsely identify convergence points, forming *phantom* optimum point distributions amongst the true distribution (see Fig. 8.3). To avoid such behavior, we consider $\vec{v}(z_t, x_j) = (z_{t+1} - z_t)_{x_j}$. Then, we learn the expected momentum $\mathbb{E}[\rho(z_t, x_j)] = \alpha \mathbb{E}[|\vec{v}(z_t, x_j)|]$ at each (z_t, x_j) during the training phase, where α is an empirically chosen scalar. In practice, $\mathbb{E}[\rho(z_t, x_j)] \to 0$



(a) Training



(b) Inference

Figure 8.1: Training and inference process. Refer to Algorithm 3 for the training process. At inference, z is iteratively updated using the predictions of $\mathcal{Z}$ and fed to $\mathcal{G}$ to obtain increasingly fine-tuned outputs (see Sec. 8.2).

as $z_{t+1}, z_t \rightarrow \{z^*\}$. Thus, to avoid *phantom* distributions, we improve the z update as,

$$z_{t+1} = z_t + \mathbb{E}[\rho(z_t, x_j)] \left[\frac{r'(z_t, x_j) - z_t}{\left\| r'(z_t, x_j) - z_t \right\|} \right]. \tag{8.16}$$

Since both $\mathbb{E}[\rho(z_t, x_j)]$ and $r'(z_t, x_j)$ are functions on (z_t, x_j), we jointly learn these two functions using a single network $\mathcal{Z}(z_t, x_j)$. Note that coefficient $\mathbb{E}[\rho(z_t, x_j)]$ serves two practical purposes: 1) slows down the movement of z near true distributions, 2) pushes z out of the phantom distributions.

8.2 Overall Design

The proposed model consists of three major blocks as shown in Fig. 8.1: an encoder $\mathcal{H}$, a generator $\mathcal{G}$, and $\mathcal{Z}$. The detailed architecture diagram for 128×128 is shown in Fig. 8.2. Note that for derivations in Sec. 8.1, we used x instead of $h = \mathcal{H}(x)$, as h is a high-level representation of x. The training process is illustrated in Algorithm 3. At each optimization $z_{t+1} = z_t - \beta \nabla_{z_t} [\hat{E}_{i,j}]$, $\mathcal{Z}$ is trained separately to approximate (z_{t+1}, ρ). At inference, x is fed to $\mathcal{H}$, and then the $\mathcal{Z}$ optimizes the output $\bar{y}$ by updating z for a pre-defined number of iterations of Eq. 8.16. For $\hat{E}(\cdot, \cdot)$, we use L_1 loss. Furthermore, it is important to limit the search space for z_{t+1}, to improve the

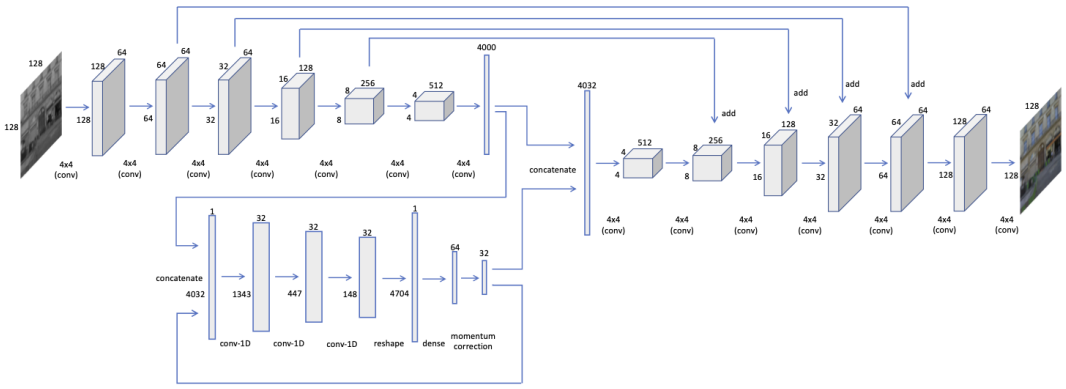


Figure 8.2: Overall architecture for 128×128 inputs.

Algorithm 3: Training algorithm

sample inputs $\{x_1, x_2, ..., x_J\} \in \chi$; sample outputs $\{y_1, y_2, ..., y_J\} \in \mathrm{Y}$;
for *k epochs* **do**
  for x *in* χ **do**
    for *l steps* **do**
      update $z = \{z_1, z_2, ..., z_J\}$: $\nabla_z \hat{E}$ ▷ Freeze $\mathcal{H}, \mathcal{G}, \mathcal{Z}$ and update z
      update $\mathcal{Z}$: $\nabla_w L_1[(z_{t+1}, \rho), \mathcal{Z}(z_t, \mathcal{H}(x))]$ ▷ Freeze $\mathcal{H}, \mathcal{G}, z$ and update $\mathcal{Z}$
    update $\mathcal{G}, \mathcal{H}$: $\nabla_w \hat{E}$ ▷ Freeze $\mathcal{Z}, z$ and update $\mathcal{H}, \mathcal{G}$

performance of $\mathcal{Z}$. To this end, we sample z from the surface of the n-dimensional sphere (S^n). Moreover, to ensure faster convergence of the model, we force the Lipschitz continuity on both $\mathcal{Z}$ and the $\mathcal{G}$.

8.3 Motivation

Here, we explain the drawbacks of conditional GAN methods and illustrate our idea via a toy example.

Incompatibility of adversarial and reconstruction losses: cGANs use a combination of adversarial and reconstruction losses. We note that this combination is suboptimal to model CMM spaces.

Remark 8.1. *Consider a generator $G(x, z)$ and a discriminator $D(x, z)$, where x and z are the input and the noise vector, respectively. Then, consider an arbitrary input x_j and the corresponding set of ground-truths $\{y^g_{i,j}\}, i = 1, 2, ..N$. Further, let us define the optimal generator $G^*(x_j, z) = \hat{y}, \hat{y} \in \{y^g_{i,j}\}$, $L_{GAN} = \mathbb{E}_i[\log D(y^g_{i,j})] + \mathbb{E}_z[\log(1 - D(G(x_j, z))]$*

and $L_\ell = \mathbb{E}_{i,z}[|y^g_{i,j} - G(x_j, z)|]$. *Then,* $G^* \neq \hat{G}^*$ *where* $\hat{G}^* = \arg\min_G \max_D L_{GAN} + \lambda L_\ell$, $\forall \lambda \neq 0$.

Proof. It is straightforward to derive the equilibrium point of $\arg\min_G \max_D L_{GAN}$ from the original GAN formulation. However, for clarity, we show some steps here.

Let,

$$V(G, D) = \arg\min_G \max_D \mathbb{E}_i[\log D(y^g_{i,j})] + \mathbb{E}_z[\log(1 - D(G(x_j, z))] \tag{8.17}$$

Let $p(\cdot)$ denote the probability distribution. Then,

$$V(G, D) = \arg\min_G \max_D \int_Y p(y^g_{\cdot,j}) \log D(y^g_{\cdot,j}) + p(\bar{y}_{\cdot,j})(\log(1 - D(G(x_j, z))dy \tag{8.18}$$

$$V(G, D) = \arg\min_G \max_D \mathbb{E}_{y \sim y^g_{\cdot,j}}[\log D(y)] + \mathbb{E}_{y \sim \bar{y}_{\cdot,j}}[\log(1 - D(y)] \tag{8.19}$$

Consider the inner loop. It is straightforward to see that $V(G, D)$ is maximized w.r.t. D when $D(y) = \frac{p(y^g_{\cdot,j})}{p(y^g_{\cdot,j}) + p(\bar{y}_{\cdot,j})}$. Then,

$$C(G) = V(G, D) = \arg\min_G \mathbb{E}_{y \sim y^g_{\cdot,j}}[\log \frac{p(y^g_{\cdot,j})}{p(y^g_{\cdot,j}) + p(\bar{y}_{\cdot,j})}] + \mathbb{E}_{y \sim \bar{y}_{\cdot,j}}[\log \frac{p(\bar{y}_{\cdot,j})}{p(y^g_{\cdot,j}) + p(\bar{y}_{\cdot,j})}] \tag{8.20}$$

Then, following the **Theorem 1** from Goodfellow et al. [2014a], it can be shown that the global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p(y^g_{\cdot,j}) = p(\bar{y}_{\cdot,j})$.

Next, consider the L_1 loss for x_j,

$$L_1 = \frac{1}{N} \sum_i \left| y^g_{i,j} - G(x_j, z, w) \right| \tag{8.21}$$

$$\nabla_w L_1 = -\frac{1}{N} \sum_i \text{sgn}(y^g_{i,j} - G(x_j, z, w)) \nabla_w (G(x_j, z, w)) \tag{8.22}$$

For L_1 to approach to a minima, $\nabla_w L_1 \to 0$. Since $\{y^g_{i,j}\}$ is not a singleton, when $L_1 \to 0$, $G(x_j, z, w) \neq \hat{y} \in \{y^g_{i,j}\}$.

Now, let us consider the L_2 loss,

$$L_2 = \frac{1}{N} \sum_i \left\| y^g_{i,j} - G(x_j, z, w) \right\|^2 \tag{8.23}$$

$$\nabla_w L_2 = -\frac{2}{N} \sum_i (y^g_{i,j} - G(x_j, z, w)) \nabla_w (G(x_j, z, w)) \tag{8.24}$$

For $\nabla_w L_2 \to 0$, $G(x_j, z, w) \to \frac{1}{N} \sum_i y^g_{i,j}$. However, omitting the very specific case

where $(\frac{1}{N}\sum_i y^g_{i,j}) \in \{y^g_{i,j}\}$, which is highly unlikely in a complex distribution, as $L_2 \to 0$, $G(x_j, z, w) \neq \hat{y} \in \{y^g_{i,j}\}$. Therefore, the goals of $\arg\min_G \max_D L_{GAN}$ and λL_ℓ are contradictory and $G^* \neq \hat{G}^*$. □

Note that we do not extend our proof to high order L losses as it is intuitive.

Generalizability: The incompatibility of above mentioned loss functions demands domain specific design choices from models that target high realism in CMM settings. This hinders the generalizability across different tasks [Vitoria et al., 2020; Zeng et al., 2019b]. We further argue that due to this discrepancy, cGANs learn sub-optimal features which are less useful for downstream tasks (Sec. 8.5.4).

Convergence and the sensitivity to the architecture: The difficulty of converging GANs to the Nash equilibrium of a non-convex min-max game in high-dimensional spaces is well explored [Barnett, 2018; Chu et al., 2020a; Kodali et al., 2017]. Goodfellow et al. [2014a] underlines *if the discriminator has enough capacity, and is optimal at every step of the GAN algorithm, then the generated distribution converges to the real distribution*; that cannot be guaranteed in a practical scenario. In fact, Arora et al. [2018] confirmed that the adversarial objective can easily approach to an equilibrium even if the generated distribution has very low support, and further, the number of training samples required to avoid mode collapse can be in order of $\exp(d)$ (d is the data dimension).

Multimodality: The ability to generate diverse outputs, i.e., convergence to multiple modes in the output space, is an important requirement. Despite the typical noise input, cGANs generally lack the ability to generate diverse outputs [Lee et al., 2019b]. Pathak et al. [2016a] and Iizuka et al. [2016] even state that better results are obtained when the noise is completely removed. Further, variants of cGAN that target diversity often face a trafe-off between the realism and diversity [He et al., 2018], as they have to compromise between the reconstruction and adversarial losses.

A toy example: Here, we experiment with the formulations in Sec. 8.1. Consider a 3D CMM space $y = \pm 4(x, x^2, x^3)$. Then, we construct three layer multi-layer perceptrons (MLP) to represent each of the functions, $\mathcal{H}$, $\mathcal{G}$, and $\mathcal{Z}$, and compare the proposed method against the L_1 loss. Figure 8.3 illustrates the results. As expected, L_1 loss generates the line $y = 0$, and is inadequate to model the multimodal space. As explained in Sec. 8.1.2, without momentum correction, the network is fooled by a phantom distribution where $\mathbb{E}[z_{t+1}] \approx 0$ at training time. However, the *push* of momentum removes the phantom distribution and refines the output to closely resemble the input distribution. As implied in Sec. 8.1.2, the momentum is maximized near the true distribution and minimized otherwise.

8.3.1 Lipschitz continuity and structuring of the latent space

Enforcing the Lipschitz constraint encourages meaningful structuring of the latent space: suppose $z^*_{1,j}$ and $z^*_{2,j}$ are two optimal codes corresponding to two ground truth

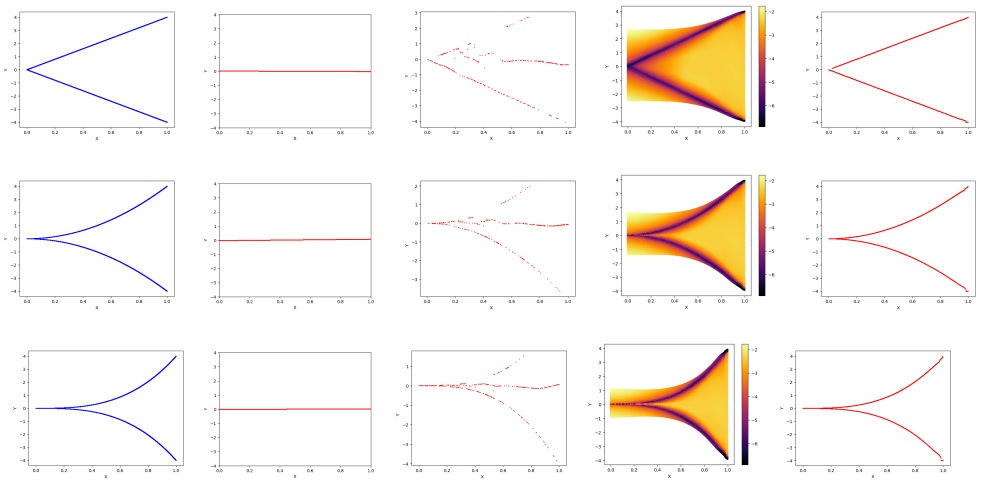


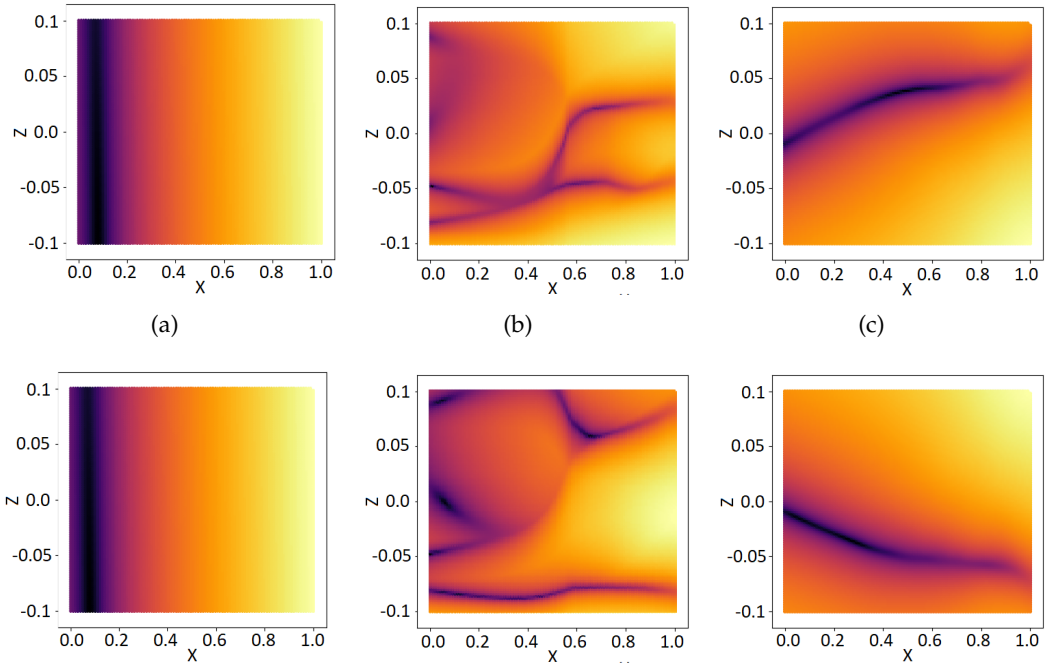
Figure 8.3: *Toy Example:* Plots generated for each dimension of the CMM space Y. (a) Ground-truth distributions. (b) Model outputs for L_1 loss. (c) Output when trained with the proposed objective (without ρ correction). Note the *phantom distribution* identified by the model. (d) $\mathbb{E}[\rho]$ as a heatmap on (x, y). $\mathbb{E}[\rho]$ is lower near the true distribution and higher otherwise. (e) Model outputs after ρ correction.

modes for a particular input. Since $\left\| z^*_{2,j} - z^*_{1,j} \right\|$ is lower bounded by $\frac{\left\| \mathcal{G}(x_j, w^*, z^*_{2,j}) - \mathcal{G}(x_j, w^*, z^*_{1,j}) \right\|}{L}$, where L is the Lipschitz constant, the minimum distance between the two latent codes is proportional to the difference between the corresponding ground truth modes. Also, in practice, we observed that this encourages the optimum latent codes to be placed sparsely. Fig. 8.4 illustrates a visualization from the toy example. As the training progresses, the optimal $\{z^*\}$ corresponding to minimas of $\hat{E}$ are identified and placed sparsely. Note that as expected, at the 10^{th} epoch the distance between the two optimum z^* increases as x goes from 0 to 1, in other words, as the $\left\| 4(x, x^2, x^3) - (-4(x, x^2, x^3)) \right\|$ increases.

Practical implementation is done as follows: during the training phase, a small noise e is injected to the inputs of $\mathcal{Z}$ and $\mathcal{G}$, and the networks are penalized for any difference in output. More formally, $L_{\mathcal{Z}}$ and $\hat{E}$ now become, $L_1[z_{t+1}, \mathcal{Z}(z_t, h)] + \alpha L_1[\mathcal{Z}(z_t + e, h + e), \mathcal{Z}(z_t, h)]$ and $L_1[y^g, \mathcal{G}(h, z)] + \alpha L_1[\mathcal{G}(h + e, z + e), \mathcal{G}(h, z)]$, respectively. Fig. 8.5 illustrates the procedure.

8.4 Experiments and discussions

The distribution of natural images lies on a high dimensional manifold, making the task of modelling it extremely challenging. Moreover, conditional image generation poses an additional challenge with their constrained multimodal output space (a single input may correspond to multiple outputs while not all of them are available for training). In this section, we experiment on several such tasks. For a fair comparison



(a) (b) (c)

(d) 0 epochs (e) 5 epochs (f) 10 epochs

Figure 8.4: The behaviour of cost heatmaps $\hat{E}$ against (x, z) as the training progresses (toy example). The latent space gets increasingly structured as $w \to w^*$. Also, in (c) the network intelligently puts the optimal latent codes further apart as the distance between the two ground truth modes ($m = 4$ and $m = -4$) keeps increasing.

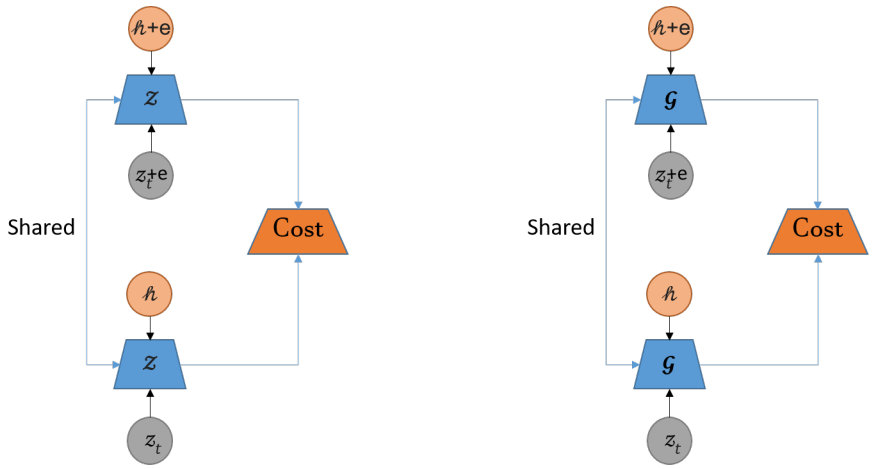


Figure 8.5: We enforce the Lipschitz continuity on both $\mathcal{G}$ and $\mathcal{Z}$.

with a similar capacity GAN, we use the encoder and decoder architectures used in Pathak et al. [2016a] for $\mathcal{H}$ and $\mathcal{G}$ respectively. We make two minor modifications: the channel-wise fully connected (FC) layers are removed and U-Net style skip connections are added.

We train the existing models for a maximum of 200 epochs where pretrained weights are not provided, and demonstrate the generalizability of our theoretical framework in diverse practical settings by using a generic network for all the experiments. Models used for comparisons are denoted as follows: PN [Zeng et al., 2019b], CA [Yu et al., 2018a], DSGAN [Yang et al., 2019a], CIC [Zhang et al., 2016], RFR [Li et al., 2020a], Chroma [Vitoria et al., 2020], P2P [Isola et al., 2017], Izuka [Iizuka et al., 2016], CE [Pathak et al., 2016a], CRN [Chen and Koltun, 2017a], and B-GAN [Zhu et al., 2017b].

8.5 Experiments

8.5.1 Experimental architectures

For the experiments on images, we mainly use 128×128 size inputs. However, to demonstrate the scalability, we use several different architectures and show that the proposed framework is capable of converging irrespective of the architecture. Fig. 8.6 shows the architectures for different input sizes.

For training, we use the Adam optimizer with hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$, and a learning rate $lr = 1 \times 10^{-5}$. We use batch normalization after each convolution layer, and leaky ReLu as the activation, except the last layer where we use *tanh*. All the weights are initialized using a random normal distribution with 0 mean and 0.5 standard deviation. Furthermore, we use a batch size of 20 for training, though we did not observe much change in performance for different batch sizes. We choose the dimensions of z to be $10, 16, 32, 64$ for 32×32, 64×64, 128×128, 256×256 input sizes, respectively. An important aspect to note here is that the dimension of z should not be increased too much, as it would increase the search space for z unnecessarily. While training, z is updated 20 times for a single $\mathcal{G}, \mathcal{H}$ update. Similarly, at inference, we use 20 update steps for z , in order to converge to the optimal solution. All the values are chosen empirically.

8.5.2 Corrupted Image Recovery

We design this task as image completion, i.e., given a masked image as input, our goal is to recover the masked area. Interestingly, we observed that the MNIST dataset, in its original form, does not have a multimodal behaviour, i.e., a fraction of the input image only maps to a single output. Therefore, we modify the training data as follows: first, we overlap the top half of an input image with the top half of another randomly sampled image. We carry out this corruption for 20% of the training data. Corrupted samples are not fixed across epochs. Then, we apply a random sized mask to the top half, and ask the network to predict the missing pixels. We choose two

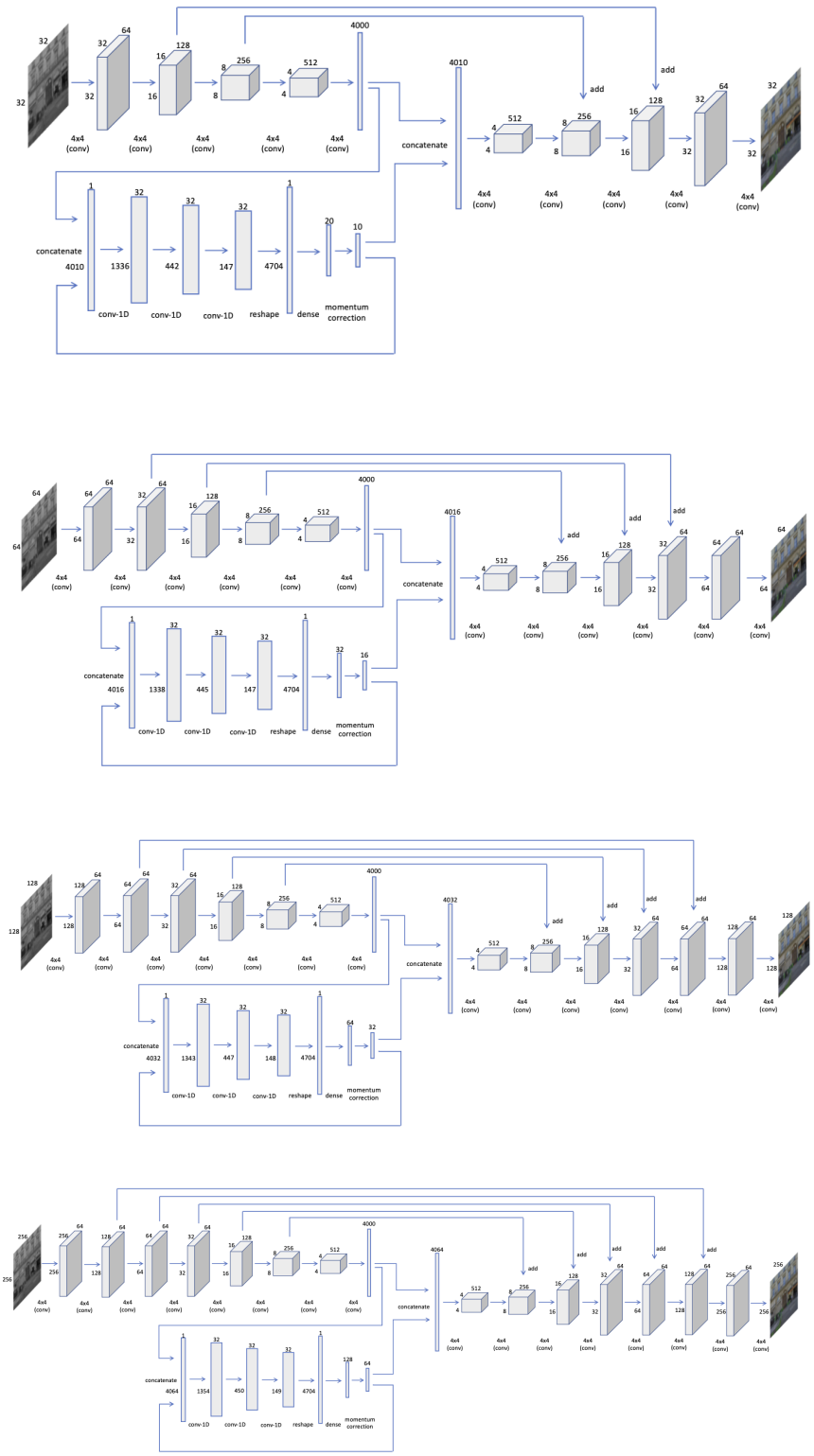


Figure 8.6: The model architecture for various input sizes. The same general structure in maintained with minimal changes to accomodate for the changing input size.

.

competitive baselines here: our network with the L_1 loss and CE. Fig. 8.7 illustrates the predictions. As shown, our model converges to the most probable non-corrupted mode without any ambiguity, while other baselines give sub-optimal results. In the next experiment, we add a small white box to the top part of the ground-truth images at different rates. At inference, our model was able to converge to both the modes (Fig. 8.8), depending on the initial position of z, as the probability of the alternate mode reaches 0.3. Moreover, Fig. 8.9 shows how the output gets more fine-tuned as z traverses to its optimal position.

Method	STL				ImageNet			
	LPIP ↓	PieAPP ↓	SSIM ↑	PSNR ↑	LPIP ↓	PieAPP ↓	SSIM ↑	PSNR ↑
Izuka	0.18	2.37	0.81	24.30	0.17	2.47	0.87	18.43
P2P	1.21	2.69	0.73	17.80	2.01	2.80	0.87	18.43
CIC	0.18	2.81	0.71	22.04	0.19	2.56	0.71	19.11
Chroma	0.16	2.06	0.91	25.57	**0.16**	2.13	0.90	23.33
Ours	**0.12**	**1.47**	**0.95**	**27.03**	**0.16**	**2.04**	**0.92**	**24.51**
Ours (w/o ρ)	0.16	1.90	0.89	25.02	0.20	2.11	0.88	23.21

Table 8.1: Colorization: Quantitative analysis of our method against the state-of-the-art. Ours perform better on a variety of metrics.

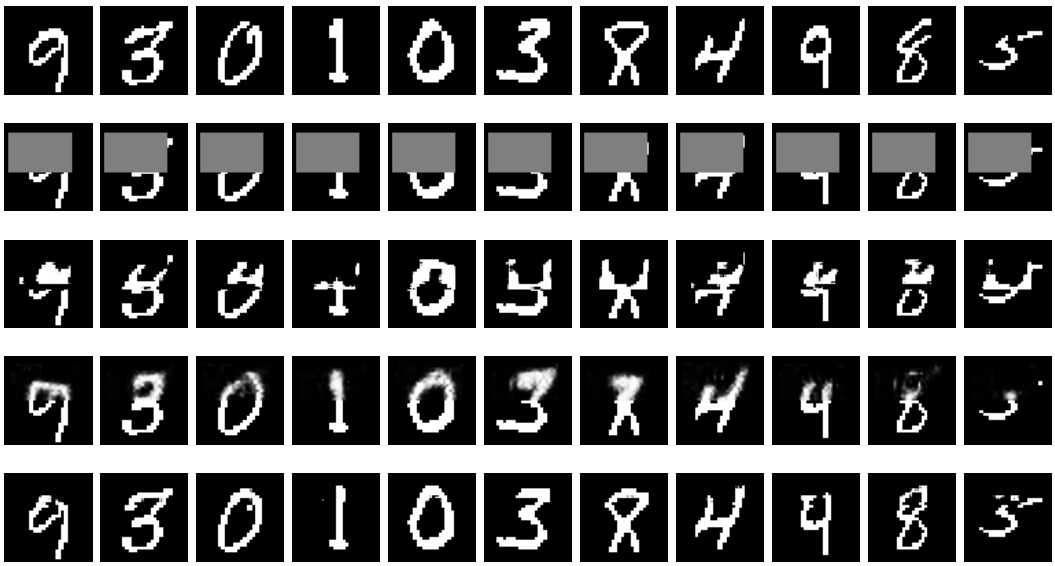


Figure 8.7: Performance with 20% corrupted data. From the top row: ground truth, inputs, outputs with L_1 loss, outputs by Pathak et al. [2016b], and outputs by our model. Our model demonstrates better convergence compared to L_1 loss and a similar capacity GAN [Pathak et al., 2016a].

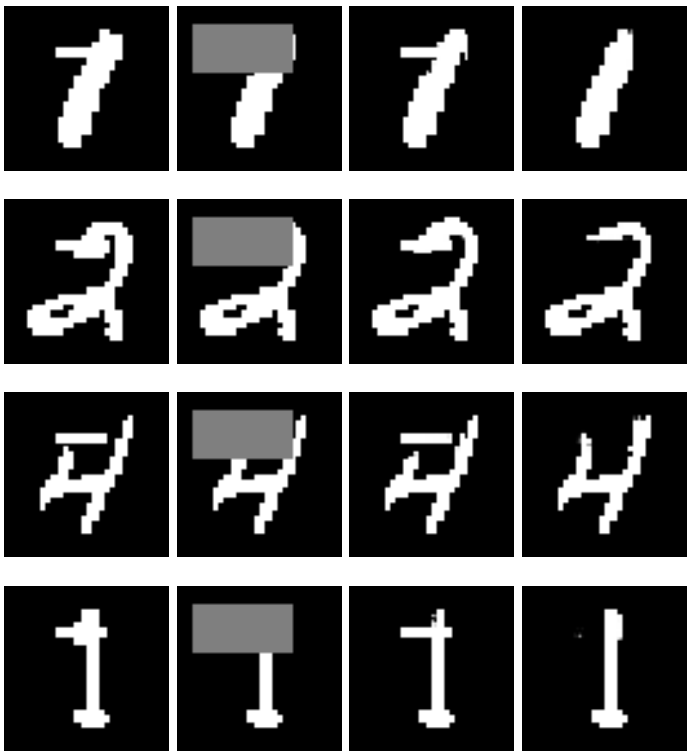


Figure 8.8: With >30% alternate mode data, our model can converge to both the input modes. From the left column: corrupted training samples, inputs, prediction mode 1, and prediction mode 2.

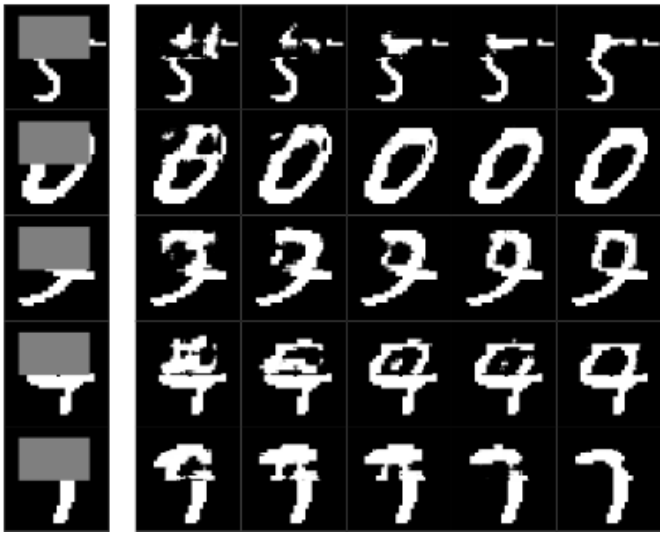


Figure 8.9: Output gets better as the z traverse to the optimum position at inference. Left column is the input. Five right columns show outputs at iterations 2, 4, 6, 8 and 10 (from left to right).

8.5.3 Automatic image colorization

Deep models have tackled this problem using semantic priors [Iizuka et al., 2016; Vitoria et al., 2020], adversarial and L_1 losses [Isola et al., 2017; Zhu et al., 2017c; Lee et al., 2019b], or by conversion to a discrete form through binning of color values [Zhang et al., 2016]. Although these methods provide compelling results, several inherent limitations exist: (a) use of semantic priors results in complex models, (b) adversarial loss suffers from drawbacks (see Sec. 8.3), and (c) discretization reduces the precision. In contrast, we achieve better results using a simpler model.

The input and the output of the network are l and (a, b) planes respectively (LAB color space). However, the color distribution of a natural dataset in a and b planes (LAB space) are strongly biased towards low values. If not taken into account, the loss function can be dominated by these desaturated values. Zhang et al. [2016] addressed this problem by rebalancing class weights according to the probability of color occurrence. However, this is only possible in a case where the output domain is discretized. To tackle this problem in the continuous domain, we push the output color distribution towards a uniform distribution as follows: we add another constraint to the cost function E to push the predicted a and b colors towards a uniform distribution: $E = \|a_{gt} - a\| + \|b_{gt} - b\| + \lambda(loss_{kl,a} + loss_{kl,b})$, where $loss_{kl,\cdot} = \mathrm{KL}(\cdot||u(0,1))$. Here, $\mathrm{KL}(\cdot||\cdot)$ is the KL divergence and $u(0,1)$ is a uniform distribution.

Fig. 8.10, Fig. 8.11, Fig. 8.12 and Table 8.1 depict our qualitative and quantitative results, respectively. We demonstrate the superior performance of our method against four metrics: LPIP, PieAPP, SSIM and PSNR. Although heavily used in the literature, per pixel metrics such as PSNR do not effectively capture the perceptual quality of an image. To overcome this shortcoming, more perceptually motivated metrics have been proposed such as SSIM Wang et al. [2004], MSSIM Wang et al. [2003], and FSIM Zhang et al. [2011]. However the similarity of two images is largely context dependant, and may not be captured by the aforementioned metrics. As a solution, recently, two deep feature based perceptual metrics–LPIP Zhang et al. [2018] and PieAPP Prashnani et al. [2018]–were proposed, which coincide well with the human judgement. To cover all these aspects, we evaluate our experiments against four metrics: LPIP, PieAPP, PSNR and SSIM. Fig. 8.13 depicts examples of multimodality captured by our model. Fig. 8.14 shows colorization behaviour as the z converges during inference.

We also compare the color distributions of the predicted a, b planes with state-of-the-art. The results are shown in Fig. 8.15 and Table 8.2. As evident, our method predicts the closest color distribution to the ground truth.

Method	a	b
Chroma	0.71	0.78
Izuka	0.68	0.63
Ours	0.82%	0.80%

Table 8.2: IOU of the predicted color distributions against the ground truth. Our method shows better results.

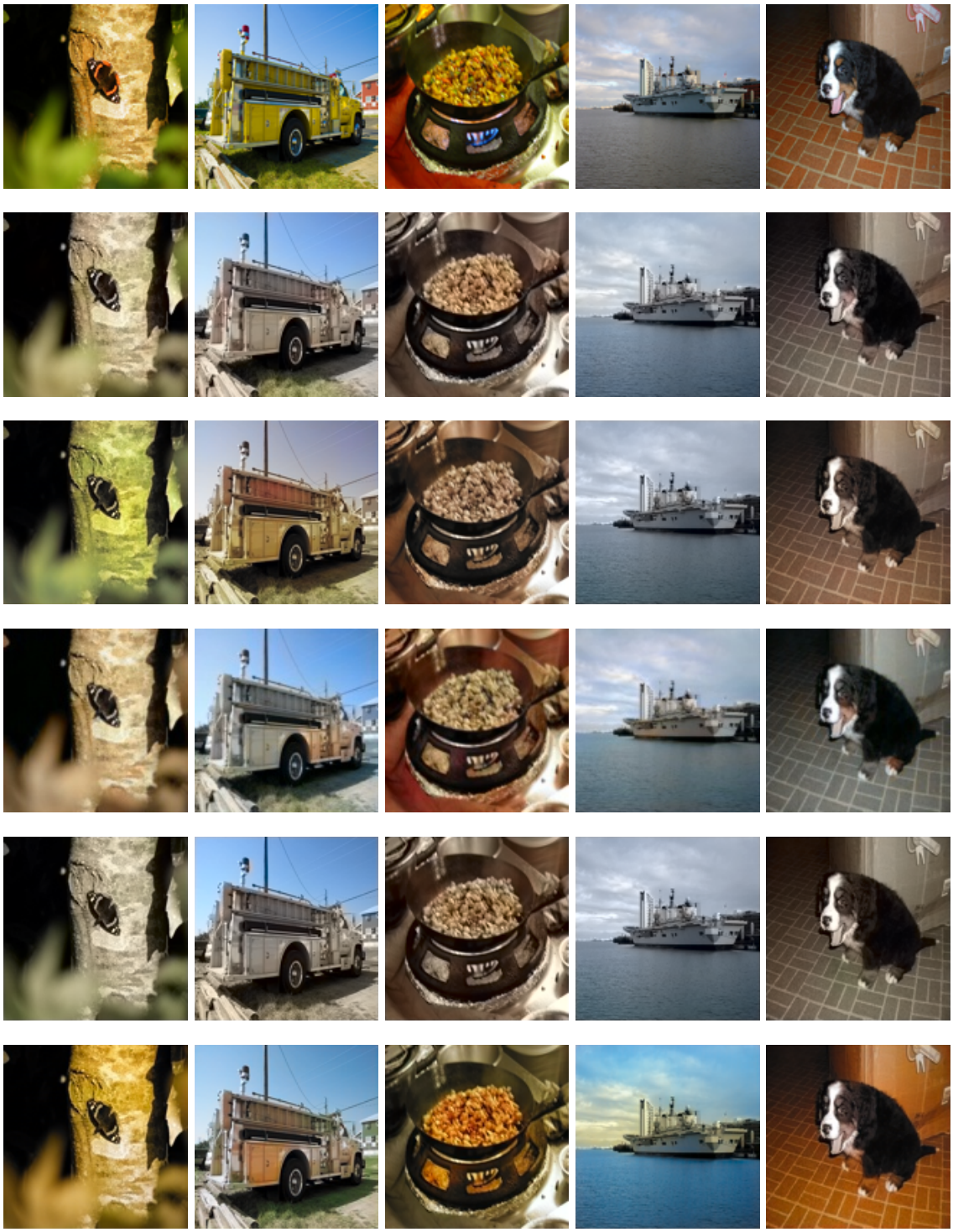


Figure 8.10: Qualitative comparison against the state-of-the-art on ImageNet dataset. From the top row: ground truth, Izuka, P2P, Chroma, CIC, and ours. Our model generally produces more vibrant and balanced color distributions.

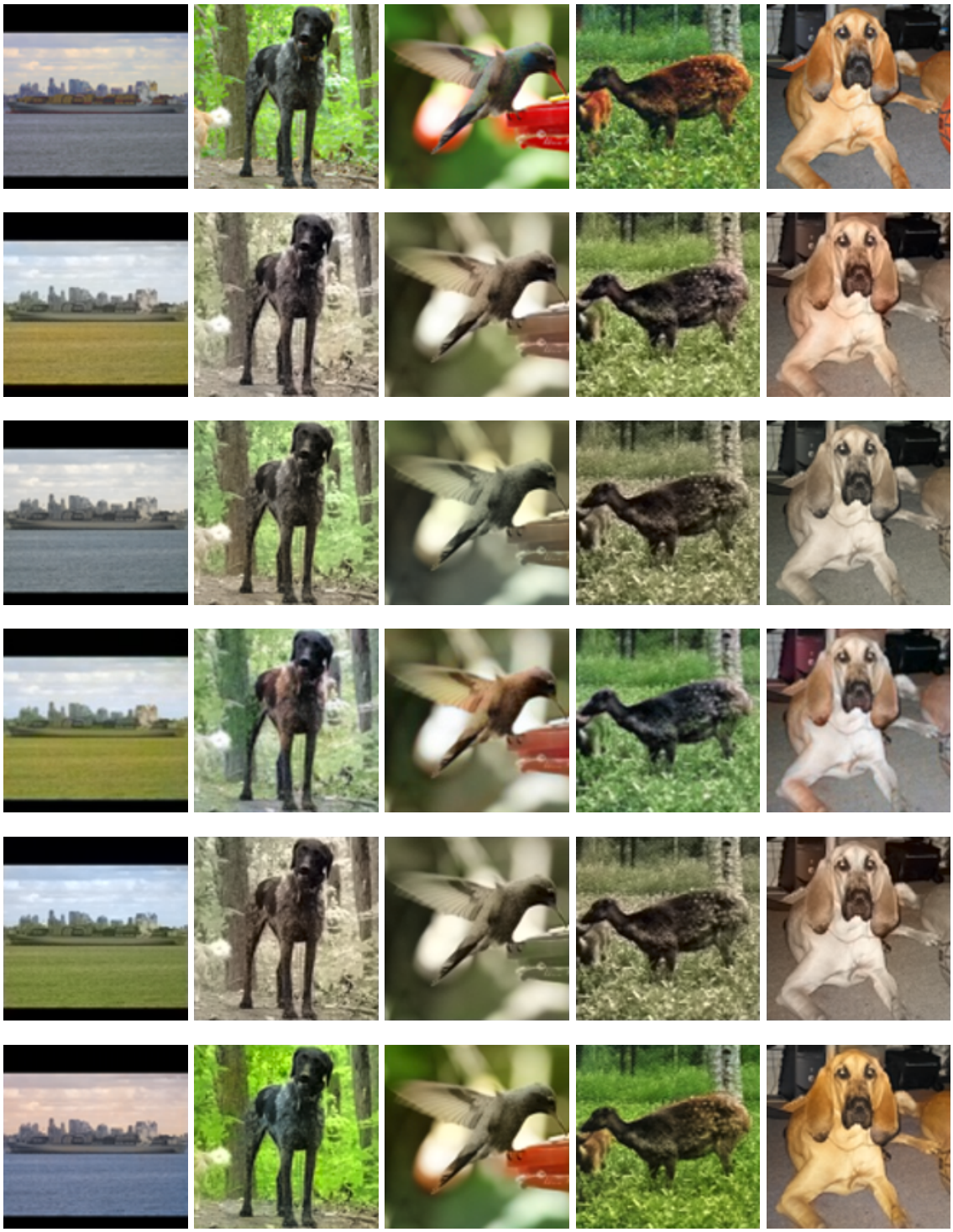


Figure 8.11: Qualitative comparison against the state-of-the-art on STL dataset. From the top row: ground truth, Izuka, P2P, Chroma, CIC, and ours. Our model generally produces more vibrant and balanced color distributions.

Figure 8.12: Qualitative results of our model in the colorization task on ImageNet dataset.

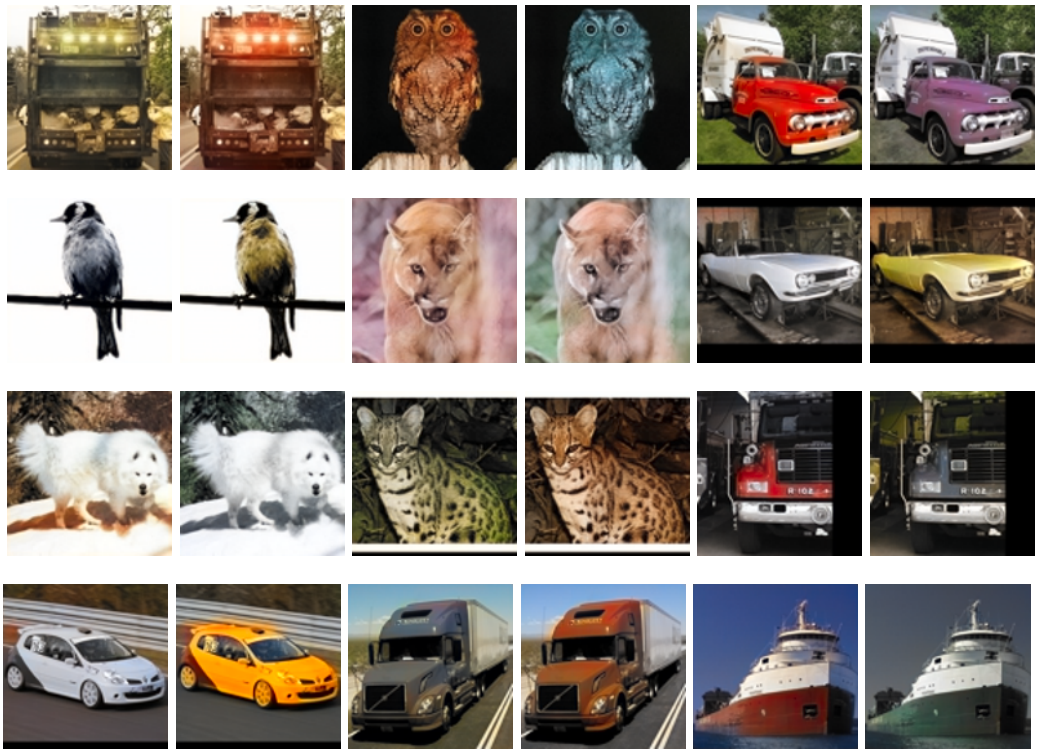


Figure 8.13: Multiple colorization modes predicted by our model for a single input. *(Best viewed in color).*

Method	User study		Turing test
	STL	ImageNet	ImageNet
Izuka	21.89	32.28	-
Chroma	32.40	31.67	-
Ours	**45.71**	**36.05**	31.66

Table 8.3: table
Colorization: Psychophysical study and Turing test results. All performances are in %.

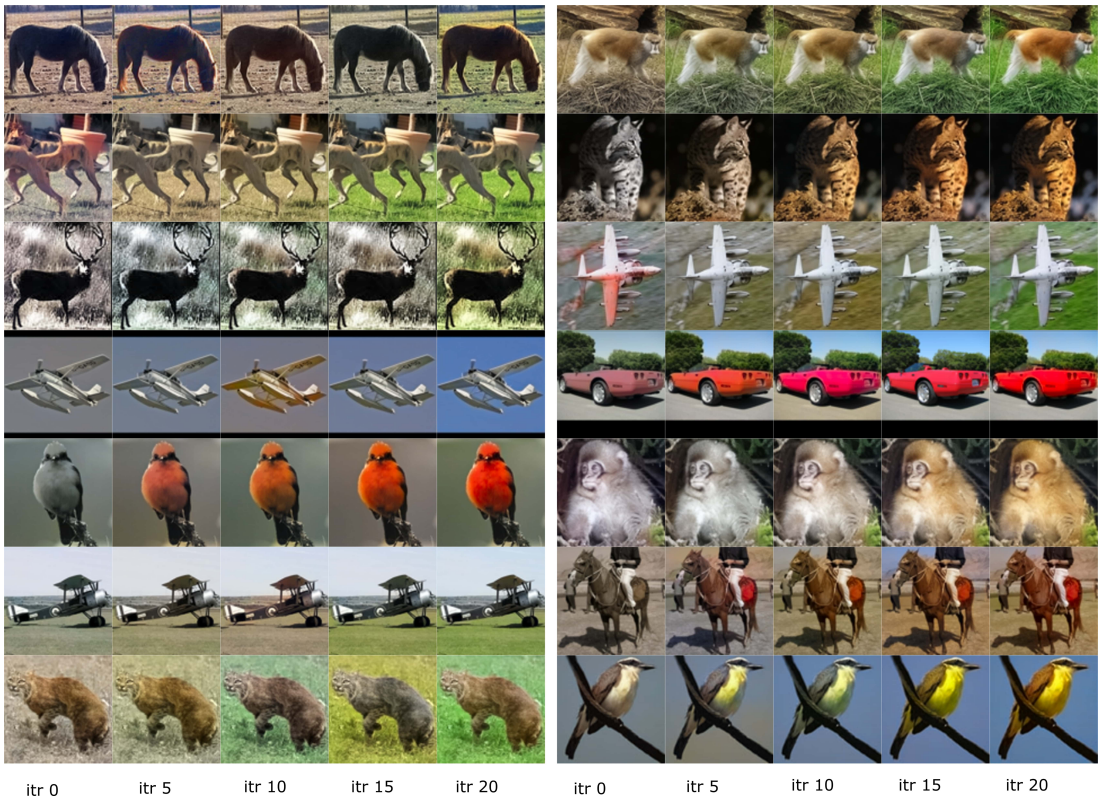


Figure 8.14: Output quality increases as $z \to z^*$ at inference.

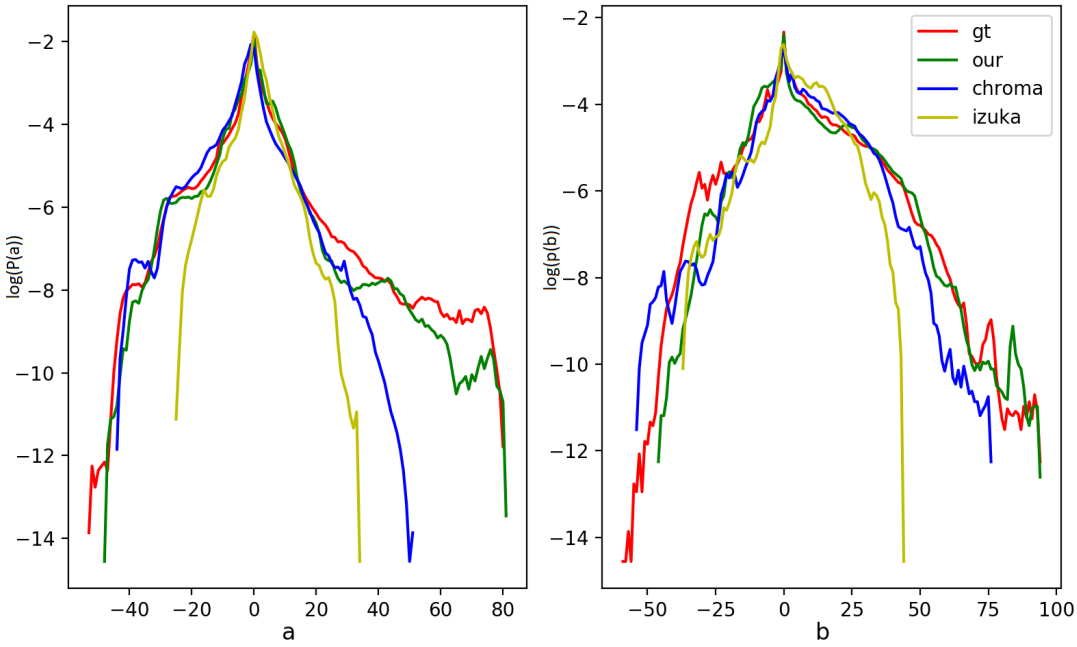


Figure 8.15: Color distribution comparison of a, b planes. Our method produces the closest distribution to the ground truth.

User study: Evaluation of synthesized images is an open problem [Salimans et al., 2016]. Although recent metrics such as LPIP [Zhang et al., 2018] and PieAPP [Prashnani et al., 2018] have been proposed, which coincide closely with human judgement, perceptual user studies remain the preferred method. Therefore, to evaluate the quality of our synthesized images in the colorization task, we conduct two types of user studies:

Therefore we conduct two user studies to further validate the quality of generated samples (Table 8.3). **a)** In the PSYCHOPHYSICAL STUDY, we present volunteers with batches of 3 images, each generated with a different method. A batch is displayed for 5 secs and the user has to pick the most realistic image. After 5 secs, the next image batch is displayed. **b)** We conduct a TURING TEST to validate our output quality against the ground-truth, following the setting proposed by Zhang et al. [2016]. The volunteers are presented with a series of paired images (ground-truth and our output). The images are visible for 1 sec, and then the user has an unlimited time to pick the realistic image.

8.5.4 Image completion

In this case, we show that our generic model outperforms a similar capacity GAN (CE) as well as task-specific GANs. In contrast to task-specific models, we do not use any domain-specific modifications to make our outputs perceptually pleasing.

We observe that with random irregular and fixed-sized masks, all the models

perform well, and we were not able to visually observe a considerable difference (see Fig. 8.16 and Fig. 8.17 for qualitative results). We also conduct Turing tests to evaluate the image completion tasks on Facades and Celeb-HQ datasets. The results are shown in Table 8.4.

Dataset	Celeb-HQ	Facades
GT	59.11%	55.75%
Ours	40.89%	44.25%

Table 8.4: Turing Test for GT vs ours on popular image datasets Celeb-HQ and Facades.

Therefore, we presented models with a more challenging task: train with random sized square-shaped masks and evaluate the performance against masks of varying sizes. Fig. 8.18 illustrates qualitative results of the models with 25% masked data. As evident, our model recovers details more accurately compared to the state-of-the-art. Notably, all models produce comparable results when trained with a fixed sized center mask, but find this setting more challenging. Table 8.5 includes a quantitative comparison. Observe that in the case of smaller sized masks, PN performs slightly better than ours, but worse otherwise. We also evaluate the learned features of the models against a downstream classification task (Table 8.6). First, we train all the models on Facades [Tyleček and Šára, 2013] against random masks, and then apply the trained models on CIFAR10 [Krizhevsky et al., 2009] to extract bottleneck features, and finally pass them through a FC layer for classification. We compare PN and ours against an oracle (AlexNet features pre-trained on ImageNet) and show our model performs closer to the oracle case.

8.5.4.1 Diversity predictions and generalizability.

We also experiment on a diverse set of image translation tasks to demonstrate our generalizability.

An appealing attribute of our network is its ability to converge to multiple optimal modes at inference. A few such examples are shown in Fig. 8.19, Fig. 8.20, Fig. 8.21, Fig. 8.22, Fig. 8.23, Fig. 8.24, and Fig. 8.25. For the *facial-land-marks-to-faces* experiment, we used the UTKFace dataset [Zhang and Qi, 2017]. For the *surface-normals-to-pets* experiment, we used the Oxford Pet dataset [Parkhi et al., 2012]. In order to get the surface normal images, we follow Bansal et al. [2017b]. First, we crop the bounding boxes of pet faces and then apply PixelNet [Bansal et al., 2017a] to extract surface normals. For *maps-to-ariel* and *edges-to-photos* experiments, we used the datasets provided by Isola et al. [2017].

For measuring the diversity, we adapt the following procedure: 1) we generate 20 random samples from the model. 2) calculate the mean pixel value μ_i of each sample. 3) pick the closest sample s_m to the average of all the mean pixels $\lambda = \frac{1}{20}\sum_{i=1}^{20}\mu_i$. 4) pick the 10 samples which have maximum mean pixel distance from s_m. 5) calculate the mean standard deviation of the 10 samples picked in step 4. 6) repeat the

Method	10% corruption				15% corruption				25% corruption			
	LPIP ↓	PieAPP ↓	PSNR ↑	SSIM ↑	LPIP ↓	PieAPP ↓	PSNR ↑	SSIM ↑	LPIP ↓	PieAPP ↓	PSNR ↑	SSIM ↑
DSGAN	0.101	1.577	20.13	0.67	0.189	2.970	18.45	0.55	0.213	3.54	16.44	0.49
PN	**0.045**	**0.639**	27.11	0.88	0.084	**0.680**	20.50	0.71	0.147	0.764	19.41	0.63
CE	0.092	1.134	22.34	0.71	0.134	2.134	19.11	0.63	0.189	2.717	17.44	0.51
P2P	0.074	0.942	22.33	0.79	0.101	1.971	19.34	0.70	0.185	2.378	17.81	0.57
CA	0.048	0.731	26.45	0.83	0.091	0.933	20.12	0.72	0.166	0.822	21.43	0.72
RFR	0.051	0.743	**29.31**	0.85	0.097	1.033	19.22	0.70	0.171	1.127	18.42	0.61
Ours (w/o ρ)	0.053	0.799	27.77	0.83	0.085	0.844	23.22	0.76	0.141	0.812	22.31	0.74
Ours	0.051	0.727	27.83	**0.89**	**0.080**	0.740	**26.43**	**0.80**	**0.129**	**0.760**	**24.16**	**0.77**

Table 8.5: *Image completion:* Quantitative analysis of our method against state-of-the-art on a variety of metrics.

Figure 8.16: Qualitative results of our model in the image completion task on Celeb-HQ dataset.

Figure 8.17: Qualitative results of our model in the image completion task on Facades dataset.

Method	Pretext	Acc. (%)
ResNet*	ImageNet Cls.	74.2
PN	Im. Completion	40.3
Ours	Im. Completion	**62.5**

Table 8.6: Comparison on downstream task (CIFAR10 cls). (*) denotes the oracle case.

Figure 8.18: Qualitative comparison for image completion with 25% missing data (models trained with random sized square masks).

experiment 5 times for each model and get the expected standard deviation.

8.5.5 Scalability

Another appealing feature of our model is its strong convergence properties irrespective of the architecture, hence, *scalability* to different input sizes. Fig. 8.27 shows examples from image completion and colorization for varying input sizes. We add layers to the architecture to be trained on increasingly high-resolution inputs, where our model was able to converge to optimal modes at each scale.

8.5.6 Convergence

The modular design of our model allows it to be a a generic model without any domain specific constraints. Consequently, our model exhibits better convergence properties, compared to state-of-the-art models. Fig. 8.29 depicts a quantitative comparison. We compare the convergence of our model on the image completion task (Paris view) against PN, CE, P2P, and CA. As evident, our model exhibits rapid and stable convergence compared to the other models.

8.5.7 Model complexity

Table 8.7 compares the number of FLOPS in our model against the other models. Despite showing impressive performance, our model consists of the second lowest

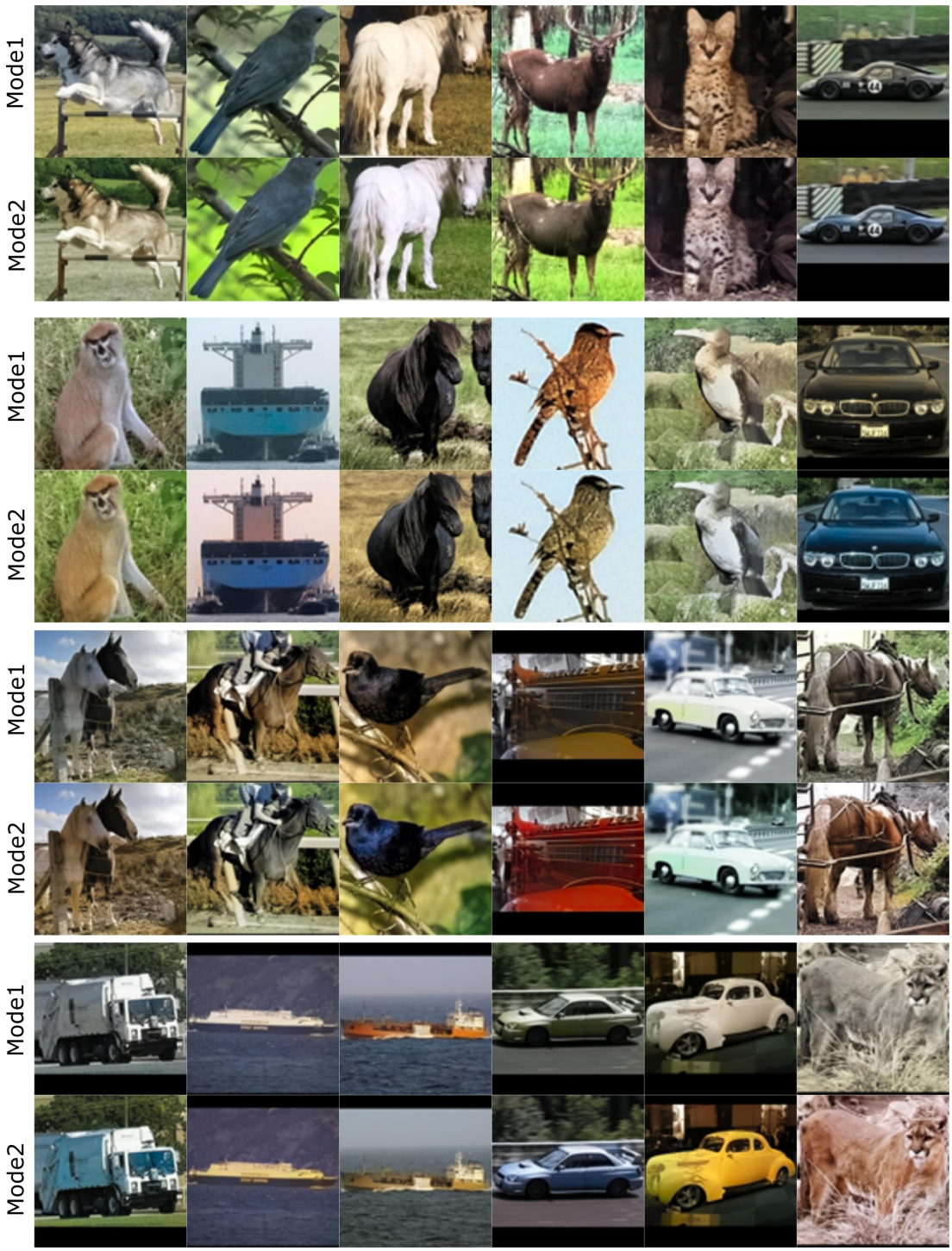


Figure 8.19: Multimodel predictions of our model in colorization

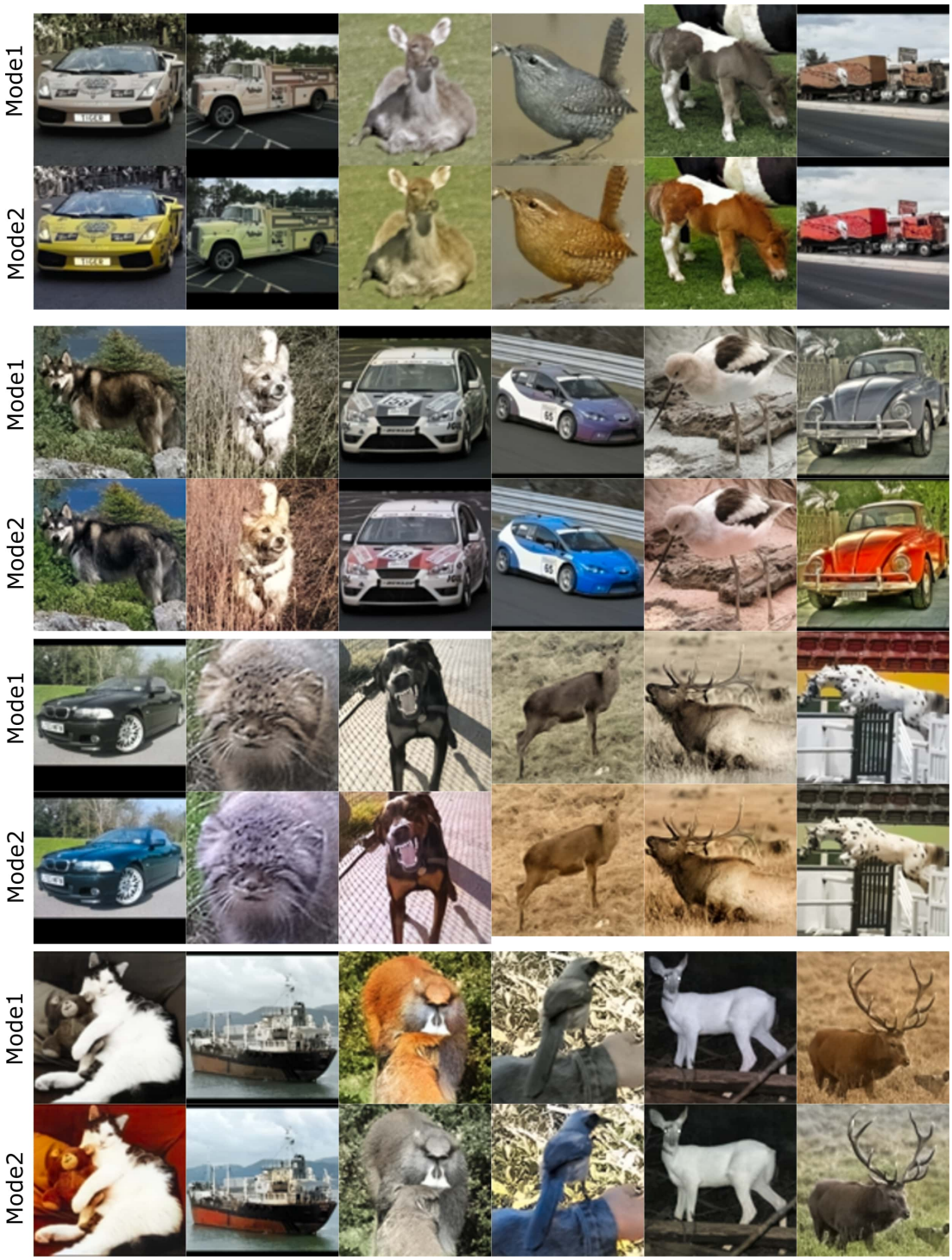


Figure 8.20: Multimodel predictions of our model in colorization

Figure 8.21: Multimodel predictions of our model in landmarks-to-faces.

Figure 8.22: Multimodel predictions of our model in face inpainting.

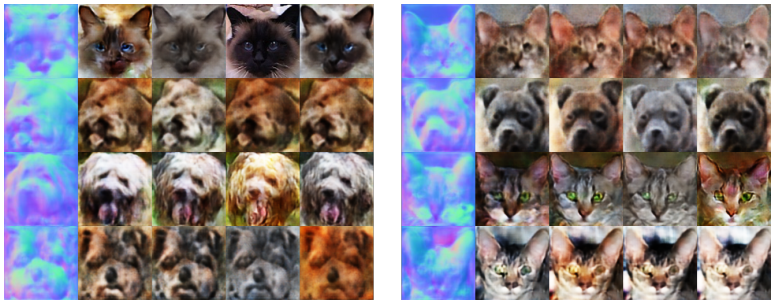


Figure 8.23: Multimodel predictions of our model in surface-normals-to-pet-faces. Note that this is generally a difficult task due to the diverse texture.

number of FLOPS, due to the simplicity of the network. Although CE contains a marginally lower number of FLOPS compared to ours, the performance of our model is far superior to CE. It is worthy to note that highly engineered models such as CIC, Izuka, and RFR comprise a significantly higher number of FLOPS than us, which demonstrates the efficacy of our approach.

Model	CE	PN	Chroma	CIC	P2P	Izuka	RFR	Ours
Flops (1×10^9)	0.634	0.946	1.275	52.839	0.732	14.082	25.64	0.638

Table 8.7: Model complexity comparison.

8.5.8 Denoising of 3D objects in spectral space

Spectral moments of 3D objects provide a compact representation, and help building light-weight networks [Ramasinghe et al., 2020a, 2019c; Cohen et al., 2018b; Esteves et al., 2018a]. However, spectral information of 3D objects has not been used before for self-supervised learning, a key reason being the difficulty of learning representations in the spectral domain due to the complex structure and unbounded spectral coefficients. Here, we present an efficient pretext task that is conducted in the spectral domain: denoising 3D spectral maps. To this end, we use two types of spectral spaces: spherical harmonics space and the Zernike space.

The minimum number of sample points required to accurately represent a finite energy function in a particular function space depends on the used sampling theorem. According to Driscoll and Healy's theorem Driscoll and Healy [1994], $4N^2$ equiangular sampled points are needed to represent a function on $\mathbb{S}^2$ using spherical moments at a maximum degree N. Therefore, we compute the first 16384 spherical moments of 3D objects where $l \leq 128$ by sampling 256×256 equiangular points in θ and ϕ directions, where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. Afterwards, we arrange the spherical moments as a 128×128 feature map, and convolve with a 2×2 kernel with stride size 2 to downsample the feature map to 64×64 size. The output is then fed to 64-size architecture. We add Gaussian noise and mask portions of the spectral map to corrupt it. Afterwards, the model is trained to de-noise the input.

Figure 8.24: Multimodel predictions of our model in sketch-to-shoes translation.

Figure 8.25: Multimodel predictions of our model in sketch-to-bag translation.

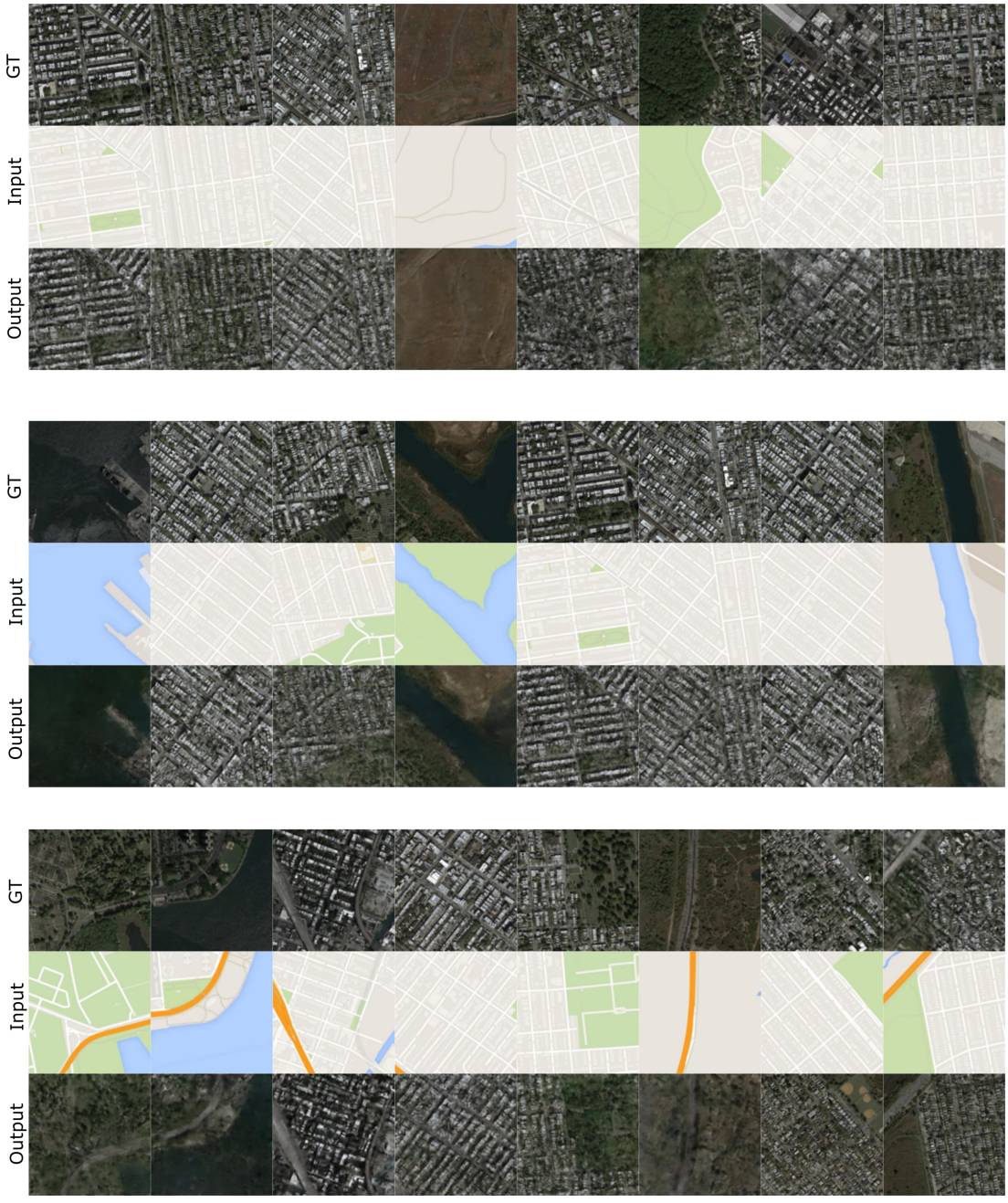


Figure 8.26: Qualitative results of our model in map-to-photo translation.

Figure 8.27: Scalability: we subsequently add layers to the architecture to be trained on increasingly high-resolution inputs. From the left: 32×32, 64×64. 128×128, 256×256.

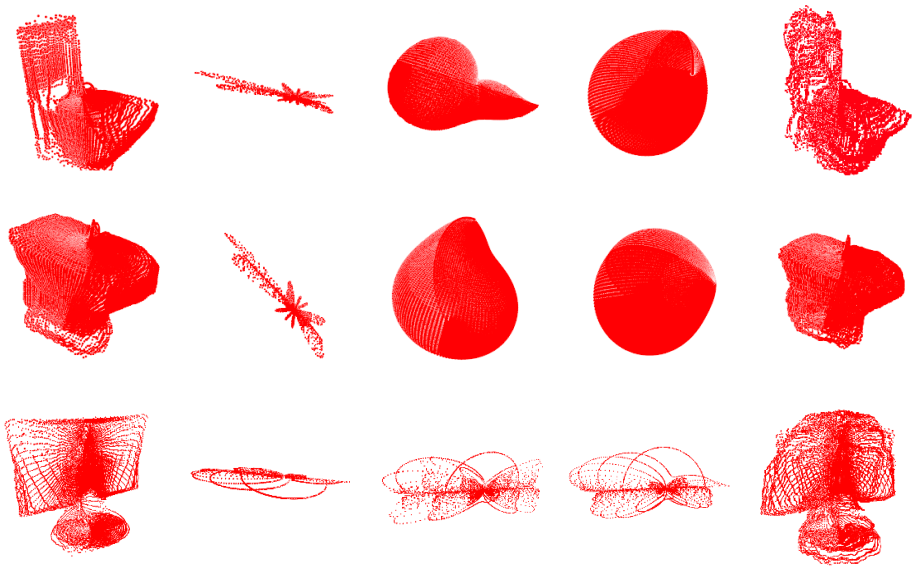


Figure 8.28: Qualitative comparison of 3D spectral denoising. The results are converted to the spatial domain for a clear visualization.

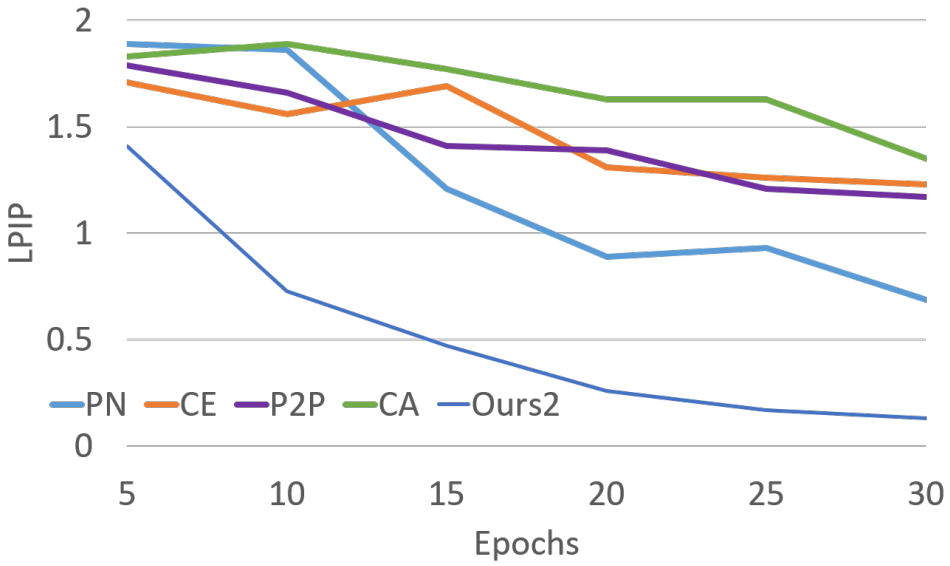


Figure 8.29: Convergence on image completion (Paris view). Our model exhibits rapid and stable convergence compared to state-of-the-art (PN, CE, P2P, CA).

For Zernike polynomials, we compute the first 100 moments for each 3D object where $n \leq 9$, and arrange the moments as a 10×10 feature map. Then, the feature map is upsampled using transposed convolution by using a 5×5 kernel and with a stride size 3. The upsamapled feature map is fed to a 32-size network and trained end-to-end to denoise. We first train the network on 55k objects in ShapeNet, and then apply the trained network on the Modelnet10 and Modelnet40 to extract the bottleneck features. These features are then fed to a single fully connected layer for classification.

Fig. 8.28 and Table 8.8 depicts our qualitative and quantitative results. We perform favorably well against other methods. To evaluate the learned features, we use Zernike polynomials, as they are more discriminative compared to spherical harmonics [Ramasinghe et al., 2019a]. We first train the network on the 55k ShapeNet objects by denoising spectral maps, and then apply the trained network on the ModelNet10 & 40. The features are then extracted from the bottleneck (similar to Sec. 8.5.4), and fed to a FC classifier (Table 8.9). We achieve the state-of-the-art results in ModelNet40 with a simple pretext task.

Method	M10	M40
CE	10.3	4.6
cVAE	8.7	4.2
Ours	**84.2**	**79.4**

Table 8.8: Reconstruction mAP of 3d spectral denoising.

Method	M10	M40
Sharma et al. [2016]	80.5%	75.5%
Han et al. [2019]	92.2%	90.2%
Achlioptas et al. [2017b]	**95.3%**	85.7%
Yang et al. [2018]	94.4%	88.4%
Sauder and Sievers [2019]	94.5%	90.6%
Ramasinghe et al. [2019d]	93.1%	-
Khan et al. [2019]	92.2%	-
Ours	92.4%	**90.9%**

Table 8.9: Downstream 3D object classification results on ModelNet10 and ModelNet40 using features learned in an unsupervised manner. All results in % accuracy.

8.5.9 Towards a measurement of uncertainty

In Bayesian approaches, the uncertainty is represented using the distribution of the network parameters ω. Since a network output is unique for fixed $\bar{w} \sim \omega$, sampling from the output is equivalent to sampling from ω. Often, ω is modeled as a parametric distribution or obtained through sampling, and at inference, the model uncertainty can be estimated as $\mathbb{VAR}_{p(y|x)}(y)$. One intuition behind this is that for more confident inputs, $p(y|x,w)$ will showcase less variance over the distribution of ω—hence lower $\mathbb{VAR}_{p(y|x)}(y)$—as the network parameters have learned redundant information [Loquercio et al., 2019].

As opposed to sampling from the distribution of network parameters, we model the optimal z^* for a particular input as a probability distribution $p(z^*)$, and measure $\mathbb{VAR}_{p(y|x)}(y)$ where $p(y|x) = \int p(y|x,z^*)p(z^*|x)dz$. Our intuition is that in the vicinity of well observed data $\mathbb{VAR}_{p(y|x)}(y)$ is lower, since for training data 1) we enforce the Lipschitz constraint on $\mathcal{G}(x,z)$ over (x,z) and 2) $\hat{E}(y^g, \mathcal{G}(x,z))$ resides in a relatively stable local minima against z^* for observed data, as in practice, $z^* = \mathbb{E}_{epochs}[z^*] + \epsilon$ for a given x, where ϵ is some random noise which is susceptible to change over each epoch. Further, Let (x,z^*) and y^g be the inputs to a network $\mathcal{G}$ and the corresponding ground truth label, respectively.

Formally, let $p(y^g|x,z^*) = \mathcal{N}(y^g; \mathcal{G}(x,z^*), \alpha\mathbb{I})$ and $z^* \sim \mathcal{U}(|z^* - \mathbb{E}(z^*)| < \delta)$, where α is some variable describing the noise in the input x and δ is a small positive scalar. Then,

$$\mathbb{COV}_{p(y^g|x)}(y^g) \approx= \frac{1}{K}\sum_{k=1}^{K}[\alpha_k\mathbb{I}] + \overline{\mathbb{COV}}(\mathcal{G}(x,z^*)). \tag{8.25}$$

where $\overline{\mathbb{COV}}$ is the sample covariance.

Proof.

$$\begin{aligned}
\mathbb{E}_{p(y^g|x)}(y^g) &= \int y^g p(y^g|x) dy^g \\
&= \int y^g [\int \mathcal{N}(y^g; \mathcal{G}(x, z^*), \alpha \mathbb{I}) p(z^*|x) dz^*] dy^g \\
&= \int [\int y^g \mathcal{N}(y^g; \mathcal{G}(x, z^*), \alpha \mathbb{I}) p(z^*|x) dy^g] dz^* \\
&= \int [\int y^g \mathcal{N}(y^g; \mathcal{G}(x, z^*), \alpha \mathbb{I}) dy^g] p(z^*|x) dz^* \\
&= \int \mathcal{G}(x, z^*) p(z^*|x) dz^*
\end{aligned} \tag{8.26}$$

Let $\pi\delta^2 = A$, and $p(z^*|x) \approx \frac{1}{A}$. Then, by Monte-Carlo approximation,

$$\approx \frac{1}{K} \sum_{k=1}^{K} \mathcal{G}(x, z_k^*) \tag{8.27}$$

Next, consider,

$$\begin{aligned}
\mathbb{COV}_{p(y^g|x)}(y^g) &= \mathbb{E}_{p(y^g|x)}((y^g)(y^g)^T) - \mathbb{E}_{p(y^g|x)}(y^g)\mathbb{E}_{p(y^g|x)}(y^g)^T \\
&= \int \int (y^g)(y^g)^T p(y^g|x, z^*) p(z^*|x) dz^* dy^g - \mathbb{E}_{p(y^g|x)}(y^g)\mathbb{E}_{p(y^g|x)}(y^g)^T \\
&= \int [\mathbb{COV}_{p(y^g|x,z^*)} + \mathbb{E}_{p(y^g|x,z^*)}\mathbb{E}^T_{p(y^g|x,z^*)} p(z^*|x) dz - \mathbb{E}_{p(y^g|x)}(y^g)\mathbb{E}_{p(y^g|x)}(y^g)^T \\
&\approx \frac{1}{K} \sum_{k=1}^{K} [\alpha_k \mathbb{I} + G(x, z_k^*) G(x, z_k^*)^T] - \frac{1}{K^2} [(\sum_{k=1}^{K} G(x, z_k^*))(\sum_{k=1}^{K} G(x, z_k^*))^T] \\
&= \frac{1}{K} \sum_{k=1}^{K} [\alpha_k \mathbb{I}] + \overline{\mathbb{COV}}(\mathcal{G}(x, z^*))
\end{aligned} \tag{8.28}$$

□

Note that in similar to Bayesian uncertainty estimations, where an approximate distribution $q(w)$ is used to estimate $p(w|D)$, where D is data, our model sample from the an empirical distribution $p(z^*|x)$. In practice, we treat α_k as a constant over all the samples–hence omit from the calculation—and use stochastic forward passes to obtain Eq. 8.25. Then, the diagonal entries are used to calculate the uncertainty in the each dimension of the output. We test this hypothesis on the toy example and the colorization task, as shown in Fig. 8.30 and Fig. 8.31, respectively.

8.6 Related work

Conditional Generative Modeling. Conditional generation involves modeling the data distribution given a set of conditioning variables that control of modes of the

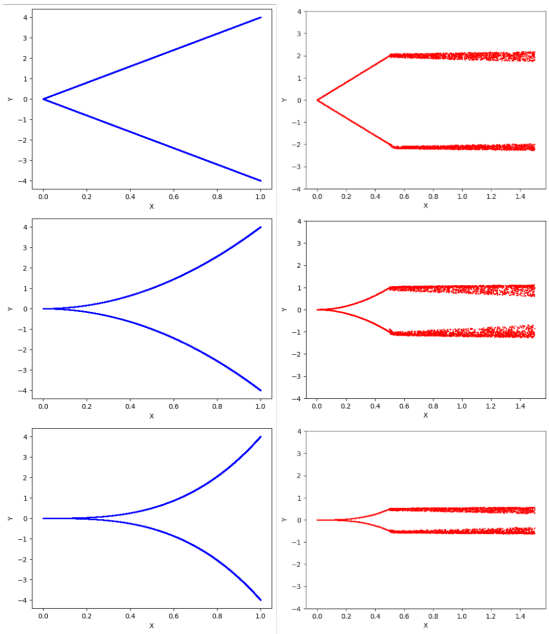


Figure 8.30: The uncertainty measurement illustration with the toy example. *(left-column: ground truth, right-column: prediction)*. We train the model with $x \in [0, 0.5]$ and test with $x \in [0, 1.5]$. During the testing, we add a small Gaussian noise to z^* at each x and get stochastic outputs. As illustrated, the sample variance (the uncertainty measurement) increases as x deviates from the observed data portion.

generated samples. With the success of *VAEs* [Kingma and Welling, 2014] and *GANs* [Goodfellow et al., 2014b] in standard generative modeling tasks, their conditioned counterparts [Sohn et al., 2015; Mirza and Osindero, 2014] have dominated conditional generative tasks recently [Vitoria et al., 2020; Zhang et al., 2016; Isola et al., 2017; Pathak et al., 2016a; Lee et al., 2019b; Zhu et al., 2017c; Bao et al., 2017; Lee et al., 2018; Zeng et al., 2019b]. While probabilistic latent variable models such as VAEs generate relatively low quality samples and poor likelihood estimates at inference [Maaløe et al., 2019], GAN based models perform significantly better at high dimensional distributions like natural images but demonstrate unstable training behaviour. A distinct feature of GANs is its mapping of points from a random noise distribution to the various modes of the output distribution. However, in the conditional case where an additional loss is incorporated to enforce the conditioning on the input, the significantly better performance of GANs is achieved at the expense of multimodality; the conditioning loss pushes the GAN to learn to mostly ignore its noise distribution. In fact, some works intentionally ignore the noise input in order to achieve more stable training [Isola et al., 2017; Pathak et al., 2016a; Mathieu et al., 2015b; Xie et al., 2018b].

Multimodality. *Conditional VAE-GANs* are one popular approach for generating multimodal outputs [Bao et al., 2017; Zhu et al., 2017c] using the VAE's ability to enforce diversity through its latent variable representation and the GAN's ability to

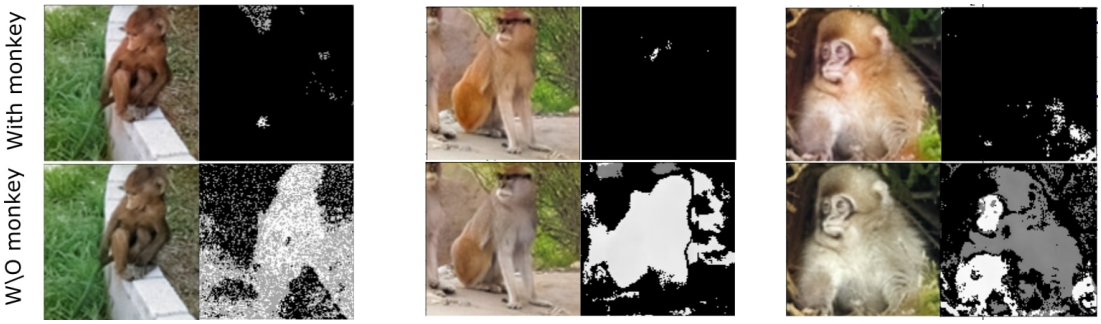


Figure 8.31: Colorization predictions for models trained with and without monkey class. Output images are shown side by side with corresponding uncertainty maps. For models trained without monkey data, high uncertainty is predicted for pixels belonging to the monkey portion (intensity is higher for high uncertainty).

enforce output fidelity through its learnt discrimanator model. *Mixture models* [Chen and Koltun, 2017b; Ghosh et al., 2018; Deshpande et al., 2017] that discretize the output space are another approach. Domain specific *disentangled representations* [Lee et al., 2018; Huang et al., 2018c] and explicit encoding of multiple modes as inputs Zhu et al. [2016]; Isola et al. [2017] have also been successful in generating diverse outputs. *Sampling-based loss* functions enforcing similarity at a distribution level [Lee et al., 2019b] have also been successful in multimoal generative tasks. Further, the use of additional *specialized reconstruction losses* (often using higher-level features extracted from the data distribution) and attention mechanisms also achieves multimodality through intricate model architectures in domain specific cases [Zeng et al., 2019b; Chen and Koltun, 2017b; Vitoria et al., 2020; Zhang et al., 2016; Iizuka et al., 2016; Zhang et al., 2020; Yu et al., 2018b; Sagong et al., 2019; Wang et al., 2018b; Iizuka et al., 2016].

We propose a simpler direction through our domain-independent energy function based approach that is also capable of learning generic representations that better support downstream tasks. Notably, our work contrasts from energy based models previously investigated for likelihood modeling due to their simplicity, however, such models are notoriously difficult to train especially on high-dimensional spaces [Du and Mordatch, 2019].

VAEs and other related work. Variational auto encoders (VAE) are a special class of autoencoders that are trained in a manner that ensures the latent space has good properties allowing the generation of new data. In VAEs, for an input x, the latent space is modeled as a probability distribution $q(z|x)$, which is sampled from a known family of distributions (typically Gaussian), as the true posterior distribution $p(z|x)$ is intractable. However, assuming the posterior distribution of the latent space as a Gaussian distribution constrains the quality of the generated data distribution, as the true distribution may be far from a Gaussian. Therefore, it is beneficial to model $q(z|x)$ as a more complex distribution, in order to generate high dimensional data distributions.

As a solution, Maaløe et al. [2016] suggested using a set of auxiliary variables a to improve the flexibility of $q(z|x)$. The key idea is to obtain a complex marginal $q(z|x) = \int q(z|a,x)p(a,x)da$, which can be non-Gaussian. On the other hand, Normalizing flows (NF) [Rezende and Mohamed, 2015], among other benefits, provides an ideal mechanism for the above task. NFs apply a series of bijective mappings $NF : P(z_i) \rightarrow P(z_{i+1})$, where $P(z_{i+1})$ is typically more complex compared to $P(z_i)$. In contrast, we do not explicitly model our latent space as a probability distribution. However, we draw some interesting analogies from a probabilistic perspective as follows: our latent space ζ can be interpreted as a set of energy surfaces $E_{x_j} : \zeta \rightarrow \mathbb{R}$, as $E_{x_j} = ||y_j^g - G(x, z_j)||$ for each ground truth mode y_j^g. From this perspective, Fig. 8.5 illustrates the energy heatmaps for the toy example. As shown, high energies are indicated by a brighter color. Since our system has a finite energy, the combined energy $E_x = \sum_j E_{x_j}$ can be transformed to a probability distribution via the Gibbs measure as $p'(z) = \frac{1}{T(\beta)} \exp(-\beta E_x(z))$, where $T(\cdot)$ is the partition function. Note that this probability is not restricted to a simple distribution.

A critical difference between the VAEs and our model is that we do not sample directly from $p'(z)$, since to obtain $p'(z)$, we need to integrate E_x over the latent space. However, our predictor network $\mathcal{Z}$ learns the high probability coordinates $\{z^*\}$ of $p'(z)$, and is able to converge to such locations at inference. This probabilistic perspective of our latent space (or the corresponding energy surface) is intuitively justified by the convergence samples shown in Fig. 52. The intermediate samples we obtain as we go from z to z^* also produce plausible results, however, the visual quality at the $z*$ is maximized, indicating high $p'(z = z^*|x)$. In contrast to NFs, our model does not explicitly learn the probability distributions, rather, the predictor network learns to converge to the high probability areas in complex distributions. Also, the complexity of the $p'(z)$ increases with the complexity and the multimodality of the corresponding higher dimensional target data distribution. Therefore, the required dimensionality of the z tends to increase in such cases. A property of NFs is that each transformation affects only a small volume in the original space, hence, we need a higher number of layers to work with high dimensional spaces (the volume grows exponentially with the dimension of the space). In contrast, we did not observe such an increase in the required capacity of the predictor with the dimension of z.

The model proposed by Chang et al. [2019] also uses a separate input S, that can vary the output of the generator, in cases where multiple loss components are used. The variable S is used both as an input to the generator and also to control the weights of the loss components while training. Interestingly, at the inference, the model is able to approximate the change in loss based on the input S, and generate diverse outputs. However, this method is more useful in scenarios where the required diversity is in the form of different styles, which can be induced by different loss functions.

8.7 Chapter summary

Conditional generation in multimodal domains is a challenging task due to its ill-posed nature. Similar to chapter, this chapter follows a modular approach and propose a novel generative framework that minimize a family of cost functions during training. In contrast to most of the existing methods, we remove the adversarial loss function from the final objective function, which gives us advantages such as training stability and faster convergence. Our architecture consists of a separate path-finding module that observes the convergence patterns of latent variables and applies this knowledge during inference to traverse to multiple output modes during inference. This modularity removes the generator from the burden of capturing the multiple modes alone. The generator communicates with the path-finding module at inference to produce multiple outputs for a single input. Despite using a simple and generic architecture, we show impressive results on a diverse set of tasks. The proposed approach demonstrates faster convergence, scalability, generalizability, diversity and superior representation learning capability for downstream tasks.

Chapter 9

Conclusions

Inducing inductive bias and structured representations into ML models is essential for ML models to reach human intelligence. Following this motivation, the thesis proposes several methods that aim to systematically include novel inductive biases and structured representations into deep models, and shows that appealing results can be achieved. The proposed methods cover both generative and discriminative tasks, in both 2D and 3D domains. This final chapter summarizes our contributions and discusses several promising future research branches in this direction.

9.1 Summary

Although deep networks have matched or exceeded human performance in various tasks, they are far from human intelligence in generalizing knowledge to new situations and learning from limited data. Inspired by insights from cognitive science and psychology, we study several methods to inject inductive bias into deep models to mimic two crucial attributes of human intelligence: combinatorial generalization and intuitive physics. Chapter 3, 4, 5, and chapter 6 focus on blending intuitive physics with deep models. Similarly, chapter 7 and chapter 8 follow the principles of combinatorial generalization.

In Chapter 3, we study the problem of equivariant representation learning. The world around us comprises many symmetries that humans can intuitively understand. ML models need an extensive amount of data to capture these symmetries without properly introduced inductive bias, i.e., equivariance. In contrast, a natural way to exploit these symmetries is to design filters that are symmetric against the relevant transformation groups. Chapter 3 proposes a novel volumetric convolution operation that is equivariant to the $\mathrm{SO}(3)$ group. To this end, we first project the data onto the Zernike basis and propose the necessary theoretical formulae to perform the convolution in the spectral space, making the convolution highly efficient and differentiable. We apply the proposed framework to the 3D object recognition and retrieval task and show that our formulations can be used to construct significantly cheaper models.

Chapter 4 further extends this work to achieve both rotational and translational equivariance. Since the Zernike basis functions do not comprise the necessary properties to obtain translational equivariance, we from scratch derive a novel set of

orthogonal and complete functions in $\mathbb{B}^3$ which comprise such properties. Further, to handle the irregularity and the redundancy of the 3D point clouds, we project the point-clouds onto a low-dimensional latent space and then perform the convolution on the latent space. We propose a theoretical framework that can integrate the aforesaid convolution and projection into a single operation, increasing the efficiency.

Chapter 5 focuses on several key problems associated with the traditional training approaches for cGANs: mode collapse, lack of structure in the latent space, and incompatibility of the loss components. We approach this problem from a geometrical perspective and posit that the aforementioned drawbacks can be eluded by preserving a homeomorphism between the latent and generated manifolds. First, we theoretically demonstrate the validity of our claims and show that the theoretical claims hold in practice by conducting empirical evaluations.

In Chapter 6, we use Bernstein-type polynomials to develop a framework for constructing flow-models. Leveraging the known properties of Bernstein-type polynomials, one can manipulate the behavior of the proposed model to a greater extent. Compared to the state-of-the-art, our framework consists of appealing properties such as higher interpretability, known error bounds, robustness, the ability to control target distribution bounds, and a constructive universality proof.

Inspired by the human brain's combinatorial generalization, Chapter 7 introduces a cascaded GAN architecture that can work collectively to generate high-resolution point-clouds. The GANs work entirely in the spectral space, where each GAN focuses on generating a pre-defined specific frequency band. At the training and inference, the GANs pass information between each other to refine outputs. While employing a single GAN to model the entire frequency band is sub-optimal—due to the different properties of each frequency band—we show that collectively, the GANs can produce high-quality outputs. Moreover, we design a generic module that can translate information between the spectral and spatial domains. This module allows the GANs to receive feedback from the spatial domain, allowing further refinement of the outputs.

Chapter 8 addresses the problem of conditional generation in multimodal output spaces. Similar to Chapter 7, we again follow a modular approach, aiming to embed a meaningful structured representation into the model. Our model is a latent variable model, i.e., the generated high-dimensional outputs are encoded in a low-dimensional latent space. We employ a separate path-finding module for guiding the generator to converge to multiple outputs at inference. The path finding module learns to traverse to optimal latent codes at the training phase and communicates with the generator at inference. The purpose of such a module is to release the generator from the burden of memorizing many output modes, allowing a simple and generalizable network across multiple tasks. We demonstrate that compared to state-of-the-art, our model has better scalability, convergence properties, and learns better features for downstream tasks.

9.2 Emerging directions

Injecting inductive biases into deep networks is increasingly gaining momentum within the machine learning community, as it allows building networks that are interpretable, efficient, robust, and more effective. Below, we briefly discuss two such emerging research directions.

9.2.1 Deep implicit layers

There are emerging applications in ML concerning systems that can be parameterized using continuous temporal inputs, solving constrained optimization problems, and smooth density estimation. Implicit layers are a framework that can be used to solve these problems. The origin of implicit layers goes back several decades [Pineda, 1987; Rico-Martinez and Kevrekidis, 1993; Farber et al., 1993]. However, the interest in incorporating implicit layers in deep networks have resurfaced through several recent pioneering works: deep declarative networks [Gould et al., 2019, 2021], structured variational autoencoders [Johnson et al., 2016b], OptNet [Amos and Kolter, 2017], and cvxpy layers [Agrawal et al., 2019].

A layer in a deep network can be defined as a *differentiable parametric function*, which can be broadly categorized into two classes: a) explicit layers and b) implicit layers. Almost all commonly used layers in deep networks (e.g., convolution, activation, dropout, batch-normalization) are explicit layers and take the form $y = f(x)$, where x,y and f are inputs, outputs, and some deterministic function, respectively. On the contrary, implicit layers demand inputs and outputs to satisfy a joint condition. For instance, the objective of an implicit layer can be to find y, such that $g(x, y) = 0$, where $g(\cdot)$ is a non-linear function. This allows for a much broader class of models and increased expressive power compared to explicit layers. Implicit layers have multiple advantages compared to explicit layers: a) implicit layers can represent complex representations such as solutions for differential equations. b) Memory efficiency. c) decoupling of the task and the solution procedure.

Implicit layers are an ideal way to incorporate physical rules that govern the natural world into deep models since many physical systems (e.g., optimizers, control systems, physics engines, and game solvers) can be represented as a solution to a non-linear equation $g(x, y) = 0$. One way to use an implicit layer in an end-to-end trainable model is to "roll-out" the solution procedure, which can typically consist of an iterative method, and back-propagate through all the steps. However, this results in long computational graphs causing unstable gradients and high memory consumption. Alternatively, we can first find an exact solution for the layer and analytically backpropagate through it at the solution points using the implicit function theorem.

Implicit layers are useful in situations where complex mathematical constraints must be integrated into a deep model. For instance, Wang et al. [2019] introduced a differentiable (smoothed) maximum satisfiability (MAXSAT) solver that can be blended into deep learning systems. The proposed solver relies on a fast coordinate

descent approach to solving the semidefinite program (SDP) correlated with the MAXSAT problem and can learn the logical structure of complex problems in a minimally supervised fashion. Yang et al. [2019c] constructed a model that can parametrize homeomorphisms and synthesize 3D point clouds. Wang et al. [2019] proposed a differentiable submodule maximization approach that can be used in data summarization, feature selection, and active learning. Song et al. [2020] used constrained stochastic differential equations for generative modeling by transforming between complex and prior distributions. Deep equilibrium models [Bai et al., 2019] are another interesting approach that aims to represent state-of-the-art deep networks using a single implicit layer in NLP and computer vision domains. Furthermore, deep implicit layers can be used for applications that include modeling continuous-time physical models [Sanchez-Gonzalez et al., 2019; Cranmer et al., 2020a; Wang et al., 2020b; Zhong et al., 2019].

Although implicit layers consist of many appealing attributes, there is still much room for improvement that demands extensive future research. Despite the low memory requirements of the implicit layers, how to optimally design architectures to utilize this advantage needs further investigation. Currently, most of the proposed models that use implicit layers follow similar architectures to typical deep learning models, which might not be ideal. Secondly, further study is needed for embedding latent stochastic differentiable equations in implicit layers for large-scale applications. Thirdly, using partial differential equations solvers as layers is another exciting research direction. Although some interesting work has been done in this regard, this aspect needs further attention. Finally, it is beneficial to investigate methods to regularize the optimization in order to reduce the required number of function evaluations for solving an implicit layer.

9.2.2 Geometric deep learning

Geometric deep learning is an umbrella term used for deep networks that use geometric inductive biases. This typically involves working with data that has a natural non-Euclidean geometry. Interestingly, many real-world systems and data consist of an underlying non-Euclidean geometry, including social networks data, sensor networks, medical imagery, genetic data, molecular structures, physics systems, and manifolds. Therefore, geometric deep learning is capturing increasing attention from the ML community.

The two typical types of constructs studied in this field are *graphs* and *manifolds*. Manifolds can be roughly considered as surfaces that are locally Euclidean, and graphs are abstract models of systems with many entities and interactions. Below, we discuss some recent advances in these two areas and consider possible future research directions.

9.2.2.1 Manifolds

Geometric deep learning on manifolds can be broadly classified as *analysis* and *synthesis*. In analysis, the salient concern is to obtain features that are symmetric under deformations. A naive way of extracting features from manifolds is to embed the manifold in the Euclidean space and apply traditional convolutions [Ji et al., 2012] . In this approach, however, the learned function is not symmetric under deformations. Therefore, it is vital to consider *geometric (intrinsic) convolutions* on the manifold, where the convolution occurs in the intrinsic coordinates of the manifold. Such an operation, by construction, is symmetric under (certain) deformations.

Geometric convolution holds unique challenges compared to its Euclidean counterpart, such as ambiguous numbers of neighbors and permutations. These hurdles can be addressed to a certain degree by performing the convolution in a suitable spectral space. To this end, the manifold is first projected onto a set of basis functions using the generalized Fourier transform—which reduces the convolution to a form of matrix multiplication—and then the convolution is performed [Shuman, 2020; Cohen et al., 2018b; Esteves et al., 2018b; Ramasinghe et al., 2019c]. However, this approach suffers from several shortcomings, including computational complexity (at least $O(n^2)$), number of parameters ($O(n)$), no guarantee of spatial localization, isotropic filters (hence no notion of direction), and filters depend on the basis (hence lack of generalization). To tackle these problems, several methods have been proposed, such as anisotropic filters [Gao et al., 2020], generalizable filters [Levie et al., 2019], polynomial filters [Defferrard et al., 2016], and rational filters [Levie et al., 2018]. Despite these efforts, we believe that a significant amount of further research is required to further enhance the geometric convolutions that are symmetric to more general groups of deformations and transformations.

On the other hand, the synthesis of manifolds has also been reinforced by many recent advancements. Graph convolutional autoencoders [Litany et al., 2018], metric preservation priors [Cosmo et al., 2020], mesh convolutional operators [Gong et al., 2019; Kulon et al., 2019; Vidaurre et al., 2020], and GANs [Goodfellow et al., 2014a] are few such notable works. The challenge of generating manifolds in higher dimensions, compared to images, is that the manifolds do not possess the canonical correspondence between predictions, inputs, and outputs. Thus, a simple reconstruction loss between the outputs and the ground truth is not optimal. Therefore, certain assumptions like fixed topology and canonical co-existence between the inputs and outputs are utilized in existing models. Hence, the synthesis of manifolds between arbitrary topology demands a more comprehensive study. Generalizability is another aspect that needs to be improved in current geometric deep models, both in analysis and synthesis. Generalizability is an important requirement in many tasks, where the model should be able to learn unified representations from multimodal inputs and cross-representations (images, point-clouds, meshes). Application of spatial transformers to non-Euclidean data [Yi et al., 2017] is perhaps an encouraging approach for this. Similarly, the expressive power, robustness, and performance guarantees of current architectures need more attention. Moreover, deep learning with meshes typically

consists of a strong theoretical background. However, strong assumptions are often put in place in practical implementations since real-world data do not always satisfy the mathematical properties of manifolds. Also, models assume prior knowledge (e.g., isometric shape deformations), and oftentimes do not apprehend the full richness of the data. Thus, there is exists a gap between the theory and implementations, which must be closely studied.

9.2.2.2 Graphs

Graphs can be used to model complex systems and interactions and are starting to be employed in important industrial applications. Several examples include fake news detection in Twitter [Meyers et al., 2020], drug discovery and design [Gaudelet et al., 2020; Stokes et al., 2020], combinatorial drug therapy [Zitnik et al., 2018], protein design [Ingraham et al., 2021; Gainza et al., 2020; Veselkov et al., 2019], recommender systems [Monti et al., 2017], particle physics [Battaglia et al., 2016; Chang et al., 2016], and medical image analysis [Ktena et al., 2017]. However, there is still a gap (compared to image and video domains) between research and large-scale industrial applications where graph-based deep learning is concerned. Therefore, it is vital to fill this gap for the progress of the research area.

Until recently, graph learning did not have a large-scale dataset that compares to Imagenet for images. Hu et al. [2020] recently released the *open graph benchmark*, which addresses this necessity, but, the field is in dire need of more large-scale datasets. Similarly, the scientific community needs to deliver more specialized software libraries and packages for geometric deep learning.

Moreover, most of the real-world graphs, e.g., social networks, are not static and exhibit continuous-time dynamic graphs. These graphs have to be modeled using streams of asynchronous node events and edge events, which are still under-explored. Temporal graph networks are a step in this direction [Rossi et al., 2020]. They learn a memory that compresses all the past interactions of a node and predicts future edges, which can be utilized in recommender systems. Besides, in some medical applications that use graph-based deep learning, e.g., decease classification, the graph must be hand-crafted explicitly [Parisot et al., 2018; Kazi et al., 2019; Li et al., 2020b]. Typically, a patient point cloud is created first using phenotypic and imaging features, and the graph is built based on it afterward. However, this is not always ideal, as for some diseases, the similarities of patients can be completely different. Therefore, it is sometimes useful to learn this graph end-to-end. To this end, Kazi et al. [2020] proposed a differentiable graph module that allows the construction of a graph from data that can be used for feature learning. This *learning the graph* concept comprises tremendous possibilities and needs more follow-up work in the future.

Modeling complex dynamical systems with graphs is another promising research direction. Cranmer et al. [2020b] proposed a framework that can model the motion of complex entities in a dynamical system using graphs. Their framework can learn the symbolic equations of the messages sent along the edges. Learning these equations provides not only better generalizability but also higher interpretability. This area has

a vast amount of practical applications, from astronomy to molecular dynamics, as it allows modeling the interactions of millions of entities embedded in a constrained system extremely fast. However, this area needs further attention to achieve better scalability. Finally, analyzing directed graphs is also a challenging topic, as such graphs comprise nonsymmetric Laplacian matrices that do not have orthogonal eigen-decompositions.

Bibliography

Abdelhamed, A.; Brubaker, M. A.; and Brown, M. S., 2019. Noise flow: Noise modeling with conditional normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3165–3173. (cited on page 117)

Achlioptas, D., 2001. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 274–281. (cited on page 99)

Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L., 2017a. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, (2017). (cited on pages 135, 138, 148, 149, and 157)

Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L., 2017b. Representation learning and adversarial generation of 3d point clouds. *arXiv preprint arXiv:1707.02392*, (2017). (cited on pages 16 and 196)

Agathos, A.; Pratikakis, I.; Papadakis, P.; Perantonis, S. J.; Azariadis, P. N.; and Sapidis, N. S., 2009. Retrieval of 3d articulated objects using a graph-based representation. *3DOR*, 2009 (2009), 29–36. (cited on page 52)

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, Z., 2019. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, (2019). (cited on pages 4 and 205)

Ahamad, A., 2018. Generating text through adversarial training using skip-thought vectors. *arXiv preprint arXiv:1808.08703*, (2018). (cited on page 14)

Ahlberg, J. H.; Nilson, E. N.; and Walsh, J. L., 1967. *The Theory of Splines and their Applications*. Academic Press. (cited on page 125)

Alipanahi, B.; Delong, A.; Weirauch, M. T.; and Frey, B. J., 2015. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33, 8 (2015), 831–838. (cited on page 4)

Amit, D. J.; Gutfreund, H.; and Sompolinsky, H., 1985. Spin-glass models of neural networks. *Physical Review A*, 32, 2 (1985), 1007. (cited on page 3)

Amos, B. and Kolter, J. Z., 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 136–145. PMLR. (cited on pages 4 and 205)

Ankerst, M.; Kastenmüller, G.; Kriegel, H.-P.; and Seidl, T., 1999. 3d shape histograms for similarity search and classification in spatial databases. In *International Symposium on Spatial Databases*, 207–226. Springer. (cited on page 25)

Arbter, K.; Snyder, W. E.; Burkhardt, H.; and Hirzinger, G., 1990. Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE Transactions on pattern analysis and machine intelligence*, 12, 7 (1990), 640–647. (cited on pages 24 and 64)

Arjovsky, M. and Bottou, L., 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, (2017). (cited on page 16)

Arjovsky, M.; Chintala, S.; and Bottou, L., 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. PMLR. (cited on page 16)

Arora, S.; Ge, R.; Liang, Y.; Ma, T.; and Zhang, Y., 2017. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, (2017). (cited on page 18)

Arora, S.; Risteski, A.; and Zhang, Y., 2018. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*. (cited on page 167)

Arora, S. and Zhang, Y., 2017. Do gans actually learn the distribution? an empirical study. *arXiv preprint arXiv:1706.08224*, (2017). (cited on page 160)

Arvanitidis, G.; Hansen, L. K.; and Hauberg, S., 2017. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, (2017). (cited on pages 84, 90, and 92)

Arvanitidis, G.; Hauberg, S.; and Schölkopf, B., 2020. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, (2020). (cited on pages 88 and 90)

Atzmon, M.; Maron, H.; and Lipman, Y., 2018. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, (2018). (cited on pages 76 and 77)

Baars, B., 1988. A cognitive theory of consciousness. cambridge universitypress.[anc, bjb, rnc](1993) how does a serial, integrated and very limited stream of consciousness emerge from a nervous system that is mostly unconscious, distributed, parallel and of enormous capacity? theoretical and experimental studies of consciousness. In *Ciba Foundation Symposium*, vol. 174, 282303. (cited on page 7)

Bahdanau, D.; Cho, K.; and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, (2014). (cited on page 1)

Bai, S.; Bai, X.; Zhou, Z.; Zhang, Z.; and Latecki, L. J., 2016. Gift: A real-time and scalable 3d shape search engine. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 5023–5032. IEEE. (cited on pages 52, 75, and 80)

Bai, S.; Kolter, J. Z.; and Koltun, V., 2019. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, (2019). (cited on page 206)

Baillargeon, R., 2004. Infants' physical world. *Current directions in psychological science*, 13, 3 (2004), 89–94. (cited on page 3)

Baillargeon, R.; Li, J.; Ng, W.; and Yuan, S., 2009. An account of infants' physical reasoning. *Learning and the infant mind*, (2009), 66–116. (cited on page 3)

Baldi, P.; Bauer, K.; Eng, C.; Sadowski, P.; and Whiteson, D., 2016. Jet substructure classification in high-energy physics with deep neural networks. *Physical Review D*, 93, 9 (2016), 094034. (cited on page 4)

Bansal, A.; Chen, X.; Russell, B.; Gupta, A.; and Ramanan, D., 2017a. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv preprint arXiv:1702.06506*, (2017). (cited on page 181)

Bansal, A.; Sheikh, Y.; and Ramanan, D., 2017b. Pixelnn: Example-based image synthesis. *arXiv preprint arXiv:1708.05349*, (2017). (cited on page 181)

Bao, J.; Chen, D.; Wen, F.; Li, H.; and Hua, G., 2017. Cvae-gan: Fine-grained image generation through asymmetric training. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on pages 90 and 198)

Barnett, S. A., 2018. Convergence problems with generative adversarial networks (gans). *arXiv preprint arXiv:1806.11382*, (2018). (cited on pages 160 and 167)

Bashiri, F. S.; Rostami, R.; Peissig, P.; D'Souza, R. M.; and Yu, Z., 2019. An application of manifold learning in global shape descriptors. *arXiv preprint arXiv:1901.02508*, (2019). (cited on page 78)

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al., 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, (2018). (cited on pages 2 and 3)

Battaglia, P. W.; Hamrick, J. B.; and Tenenbaum, J. B., 2013. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110, 45 (2013), 18327–18332. (cited on page 4)

Battaglia, P. W.; Pascanu, R.; Lai, M.; Rezende, D.; and Kavukcuoglu, K., 2016. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, (2016). (cited on page 208)

Ben-Shabat, Y.; Lindenbaum, M.; and Fischer, A., 2017. 3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks. *arXiv preprint arXiv:1711.08241*, (2017). (cited on page 76)

Bentley, J. L., 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18, 9 (1975), 509–517. (cited on page 61)

Berg, R. v. d.; Hasenclever, L.; Tomczak, J. M.; and Welling, M., 2018. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, (2018). (cited on page 118)

Bernstein, S., 1912. Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. *Communications of the Kharkov Mathematical Society*, (1912), 1–2. (cited on pages 119 and 121)

Bertasius, G.; Wang, H.; and Torresani, L., 2021. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, (2021). (cited on page 1)

Bobrow, D. G. and Winograd, T., 1977. An overview of krl, a knowledge representation language. *Cognitive science*, 1, 1 (1977), 3–46. (cited on page 3)

Bogachev, V. I.; Kolesnikov, A. V.; and Medvedev, K. V., 2005. Triangular transformations of measures. *Sbornik: Mathematics*, 196, 3 (2005), 309. (cited on pages 20 and 118)

Bolthausen, E., 2014. An iterative construction of solutions of the tap equations for the sherrington–kirkpatrick model. *Communications in Mathematical Physics*, 325, 1 (2014), 333–366. (cited on page 4)

Boomsma, W. and Frellsen, J., 2017. Spherical convolutions and their application in molecular modelling. In *Advances in Neural Information Processing Systems*, 3436–3446. (cited on page 21)

Boscaini, D.; Masci, J.; Melzi, S.; Bronstein, M. M.; Castellani, U.; and Vandergheynst, P., 2015. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, vol. 34, 13–23. Wiley Online Library. (cited on page 57)

Boscaini, D.; Masci, J.; Rodolà, E.; and Bronstein, M., 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, 3189–3197. (cited on page 57)

Botvinick, M. M.; Braver, T. S.; Barch, D. M.; Carter, C. S.; and Cohen, J. D., 2001. Conflict monitoring and cognitive control. *Psychological review*, 108, 3 (2001), 624. (cited on page 6)

Boutsidis, C.; Drineas, P.; Kambadur, P.; Kontopoulou, E.-M.; and Zouzias, A., 2017. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533 (2017), 95–117. (cited on page 99)

Brock, A.; De, S.; Smith, S. L.; and Simonyan, K., 2021. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, (2021). (cited on page 1)

Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N., 2016. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, (2016). (cited on pages 49, 50, 61, 75, and 76)

Bronstein, A.; Bronstein, M.; Ovsjanikov, M.; and Guibas, L., 2009. Shape google: a computer vision approach to invariant shape retrieval. *Proc. NORDIA*, 1, 4 (2009), 6. (cited on page 64)

Bronstein, A. M.; Bronstein, M. M.; Kimmel, R.; Mahmoudi, M.; and Sapiro, G., 2010. A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. *International Journal of Computer Vision*, 89, 2-3 (2010), 266–286. (cited on page 64)

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P., 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34, 4 (2017), 18–42. (cited on page 57)

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y., 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, (2013). (cited on page 57)

Bubeck, S. and Sellke, M., 2021. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34 (2021). (cited on page 21)

Bustamante, J., 2017. *Bernstein Operators and Their Properties*. Birkhauser. (cited on pages 121 and 130)

Canterakis, N., 1996. Complete moment invariants and pose determination for orthogonal transformations of 3d objects. In *Mustererkennung 1996*, 339–350. Springer. (cited on pages 24 and 25)

Canterakis, N., 1999. 3d zernike moments and zernike affine invariants for 3d image analysis and recognition. In *In 11th Scandinavian Conf. on Image Analysis*. Citeseer. (cited on pages 22, 24, 64, and 67)

Carrière, M.; Oudot, S. Y.; and Ovsjanikov, M., 2015. Stable topological signatures for points on 3d shapes. In *Computer Graphics Forum*, vol. 34, 1–12. Wiley Online Library. (cited on page 59)

Chang, M. B.; Ullman, T.; Torralba, A.; and Tenenbaum, J. B., 2016. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, (2016). (cited on page 208)

Chang, S.; Park, S.; Yang, J.; and Kwak, N., 2019. Sym-parameterized dynamic inference for mixed-domain image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, 4803–4811. (cited on pages 90, 159, and 200)

Chen, D.-Y.; Tian, X.-P.; Shen, Y.-T.; and Ouhyoung, M., 2003. On visual similarity based 3d model retrieval. In *Computer graphics forum*, vol. 22, 223–232. Wiley Online Library. (cited on pages 64 and 147)

Chen, Q. and Koltun, V., 2017a. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, 1511–1520. (cited on page 170)

Chen, Q. and Koltun, V., 2017b. Photographic image synthesis with cascaded refinement networks. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 199)

Cheraghian, A. and Petersson, L., 2019. 3dcapsule: Extending the capsule architecture to classify 3d point clouds. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1194–1202. doi:10.1109/WACV.2019.00132. (cited on page 61)

Cheraghian, A.; Rahman, S.; Campbell, D.; and Petersson, L., 2019a. Mitigating the hubness problem for zero-shot learning of 3d objects. In *British Machine Vision Conference (BMVC'19)*. (cited on page 61)

Cheraghian, A.; Rahman, S.; Campbell, D.; and Petersson, L., 2020. Transductive zero-shot learning for 3d point cloud classification. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. doi:10.1109/WACV.2019.00132. (cited on page 61)

Cheraghian, A.; Rahman, S.; and Petersson, L., 2019b. Zero-shot learning of 3d point cloud objects. In *International Conference on Machine Vision Applications (MVA)*. (cited on page 61)

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J., 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8789–8797. (cited on page 14)

Chomsky, N., 2014. *Aspects of the Theory of Syntax*, vol. 11. MIT press. (cited on page 6)

Chu, C.; Minami, K.; and Fukumizu, K., 2020a. Smoothness and stability in gans. *arXiv preprint arXiv:2002.04185*, (2020). (cited on pages 160 and 167)

Chu, M.; Xie, Y.; Mayer, J.; Leal-Taixé, L.; and Thuerey, N., 2020b. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39, 4 (2020), 75–1. (cited on page 14)

Chung, J. S.; Nagrani, A.; Coto, E.; Xie, W.; McLaren, M.; Reynolds, D. A.; and Zisserman, A., 2019. Voxsrc 2019: The first voxceleb speaker recognition challenge. *arXiv preprint arXiv:1912.02522*, (2019). (cited on page 1)

Coates, A.; Ng, A.; and Lee, H., 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223. (cited on page 101)

Cohen, T.; Geiger, M.; and Weiler, M., 2018a. A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017*, (2018). (cited on pages 11 and 59)

Cohen, T. S.; Geiger, M.; Köhler, J.; and Welling, M., 2018b. Spherical cnns. *arXiv preprint arXiv:1801.10130*, (2018). (cited on pages 2, 11, 21, 22, 24, 30, 32, 63, 64, 77, 190, and 207)

Cosmo, L.; Norelli, A.; Halimi, O.; Kimmel, R.; and Rodola, E., 2020. Limp: Learning latent shape representations with metric preservation priors. *arXiv preprint arXiv:2003.12283*, 2 (2020). (cited on page 207)

Cranmer, M.; Greydanus, S.; Hoyer, S.; Battaglia, P.; Spergel, D.; and Ho, S., 2020a. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, (2020). (cited on page 206)

Cranmer, M.; Sanchez-Gonzalez, A.; Battaglia, P.; Xu, R.; Cranmer, K.; Spergel, D.; and Ho, S., 2020b. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*, (2020). (cited on page 208)

Dash, A.; Gamboa, J. C. B.; Ahmed, S.; Liwicki, M.; and Afzal, M. Z., 2017. Tac-gan-text conditioned auxiliary classifier generative adversarial network. *arXiv preprint arXiv:1703.06412*, (2017). (cited on page 14)

Decelle, A.; Krzakala, F.; Moore, C.; and Zdeborová, L., 2011. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84, 6 (2011), 066106. (cited on page 3)

Deco, G. and Brauer, W., 1995. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8, 4 (1995), 525–535. (cited on page 20)

Defferrard, M.; Bresson, X.; and Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, (2016). (cited on pages 57 and 207)

Defferrard, M.; Milani, M.; Gusset, F.; and Perraudin, N., 2020. Deepsphere: a graph-based spherical cnn. *arXiv preprint arXiv:2012.15000*, (2020). (cited on page 2)

Dehaene, S.; Lau, H.; and Kouider, S., 2017. What is consciousness, and could machines have it? *Science*, 358, 6362 (2017), 486–492. (cited on page 7)

Denton, E. L.; Chintala, S.; Fergus, R.; et al., 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 1486–1494. (cited on pages 16 and 153)

Deshpande, A.; Lu, J.; Yeh, M.-C.; Jin Chong, M.; and Forsyth, D., 2017. Learning diverse image colorization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 199)

Dinh, L.; Krueger, D.; and Bengio, Y., 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, (2014). (cited on pages 20 and 118)

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S., 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, (2016). (cited on pages 20 and 118)

Donahue, C.; McAuley, J.; and Puckette, M., 2018. Synthesizing audio with generative adversarial networks. *arXiv preprint arXiv:1802.04208*, 1 (2018). (cited on page 14)

Dong, H.-W.; Hsiao, W.-Y.; Yang, L.-C.; and Yang, Y.-H., 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. (cited on page 14)

Driscoll, J. R. and Healy, D. M., 1994. Computing fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 15, 2 (1994), 202–250. (cited on pages 151 and 190)

Du, Y. and Mordatch, I., 2019. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems 32* (Eds. H. Wallach; H. Larochelle; A. Beygelzimer; F. d Alché-Buc; E. Fox; and R. Garnett), 3608–3618. Curran Associates, Inc. (cited on page 199)

Dumoulin, V. and Visin, F., 2016. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, (2016). (cited on page 2)

Dupoux, E., 2018. Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner. *Cognition*, 173 (2018), 43–59. (cited on pages xvii and 4)

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G., 2019a. Cubic-spline flows. *arXiv preprint arXiv:1906.02145*, (2019). (cited on pages 118, 125, and 127)

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G., 2019b. Neural spline flows. *arXiv preprint arXiv:1906.04032*, (2019). (cited on pages 20, 118, 125, and 127)

Džeroski, S.; De Raedt, L.; and Driessens, K., 2001. Relational reinforcement learning. *Machine learning*, 43, 1 (2001), 7–52. (cited on page 7)

El Mallahi, M.; Zouhri, A.; El Affar, A.; Tahiri, A.; and Qjidaa, H., 2017. Radial hahn moment invariants for 2d and 3d image recognition. *International Journal of Automation and Computing*, (2017), 1–13. (cited on page 24)

El Moselhy, T. A. and Marzouk, Y. M., 2012. Bayesian inference with optimal maps. *Journal of Computational Physics*, 231, 23 (2012), 7815–7850. (cited on page 20)

Elad, M.; Tal, A.; and Ar, S., 2002. Content based retrieval of vrml objects—an iterative and interactive approach. In *Multimedia 2001*, 107–118. Springer. (cited on page 64)

Engel, J.; Agrawal, K. K.; Chen, S.; Gulrajani, I.; Donahue, C.; and Roberts, A., 2019. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, (2019). (cited on page 14)

Esling, P.; Masuda, N.; Bardet, A.; Despres, R.; et al., 2019. Universal audio synthesizer control with normalizing flows. *arXiv preprint arXiv:1907.00971*, (2019). (cited on page 117)

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, vol. 96, 226–231. (cited on page 56)

Esteves, C.; Allen-Blanchette, C.; Makadia, A.; and Daniilidis, K., 2018a. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 52–68. (cited on pages 2, 11, and 190)

Esteves, C.; Allen-Blanchette, C.; Makadia, A.; and Daniilidis, K., 2018b. Learning so(3) equivariant representations with spherical cnns. In *The European Conference on Computer Vision (ECCV)*. (cited on pages 52, 53, 63, 64, 80, and 207)

Farber, R. M.; Lapedes, A. S.; Rico-Martínez, R.; and Kevrekidis, I. G., 1993. Identification of continuous-time dynamical systems: Neural network based algorithms and parallel implementation. *arXiv preprint comp-gas/9305001*, (1993). (cited on page 205)

Farouki, R. T. and Goodman, T. N. T., 1996. On the optimal stability of the Bernstein basis. *Mathematics of Computation*, 65, 216 (1996), 1553—-1566. (cited on pages 122, 123, and 126)

Farouki, R. T. and Rajan, V. T., 1987. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design 4*, 4, 3 (1987), 191—-216. (cited on pages 122, 123, and 126)

Fetaya, E.; Jacobsen, J.-H.; Grathwohl, W.; and Zemel, R., 2020. Understanding the limitations of conditional generative models. In *International Conference on Learning Representations*. (cited on page 159)

Fikes, R. E. and Nilsson, N. J., 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2, 3-4 (1971), 189–208. (cited on page 7)

Flusser, J.; Boldys, J.; and Zitová, B., 2003. Moment forms invariant to rotation and blur in arbitrary number of dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, 2 (2003), 234–246. (cited on pages 24 and 64)

Fortunato, S., 2010. Community detection in graphs. *Physics reports*, 486, 3-5 (2010), 75–174. (cited on page 3)

Fotenos, A. F.; Mintun, M. A.; Snyder, A. Z.; Morris, J. C.; and Buckner, R. L., 2008. Brain volume decline in aging: evidence for a relation between socioeconomic status, preclinical alzheimer disease, and reserve. *Archives of neurology*, 65, 1 (2008), 113–120. (cited on page 80)

Frome, A.; Huber, D.; Kolluri, R.; Bülow, T.; and Malik, J., 2004. Recognizing objects in range data using regional point descriptors. In *European conference on computer vision*, 224–237. Springer. (cited on page 25)

Fukushima, K. and Miyake, S., 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, 267–285. Springer. (cited on page 3)

Furuya, T. and Ohbuchi, R., 2016. Deep aggregation of local 3d geometric features for 3d model retrieval. In *BMVC*. (cited on pages 52, 53, 77, and 80)

Gabrié, M.; Tramel, E. W.; and Krzakala, F., 2015. Training restricted boltzmann machines via the thouless-anderson-palmer free energy. *arXiv preprint arXiv:1506.02914*, (2015). (cited on page 4)

Gainza, P.; Sverrisson, F.; Monti, F.; Rodola, E.; Boscaini, D.; Bronstein, M.; and Correia, B., 2020. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17, 2 (2020), 184–192. (cited on page 208)

Gallot, S.; Hulin, D.; and Lafontaine, J., 1990. *Riemannian geometry*, vol. 2. Springer. (cited on page 89)

Gao, Z.; Zhai, G.; Zhang, J.; Yang, Y.; and Yang, X., 2020. Pai-gcn: Permutable anisotropic graph convolutional networks for 3d shape representation learning. *arXiv preprint arXiv:2004.09995*, (2020). (cited on page 207)

Garcia-Garcia, A.; Gomez-Donoso, F.; Garcia-Rodriguez, J.; Orts-Escolano, S.; Cazorla, M.; and Azorin-Lopez, J., 2016. Pointnet: A 3d convolutional neural network for real-time object class recognition. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, 1578–1584. IEEE. (cited on page 49)

Gardner, E., 1988. The space of interactions in neural network models. *Journal of physics A: Mathematical and general*, 21, 1 (1988), 257. (cited on page 3)

Gaudelet, T.; Day, B.; Jamasb, A. R.; Soman, J.; Regep, C.; Liu, G.; Hayter, J. B.; Vickers, R.; Roberts, C.; Tang, J.; et al., 2020. Utilising graph machine learning within drug discovery and development. *arXiv preprint arXiv:2012.05716*, (2020). (cited on page 208)

Gauss, C. F., 1828. *Disquisitiones generales circa superficies curvas*, vol. 1. Typis Dieterichianis. (cited on page 88)

Germain, M.; Gregor, K.; Murray, I.; and Larochelle, H., 2015. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, 881–889. PMLR. (cited on page 20)

Gerstenberg, T.; Goodman, N. D.; Lagnado, D. A.; and Tenenbaum, J. B., 2015. How, whether, why: Causal judgments as counterfactual contrasts. In *CogSci*. (cited on page 4)

Ghosh, A.; Kulharia, V.; Namboodiri, V. P.; Torr, P. H.; and Dokania, P. K., 2018. Multi-agent diverse generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 199)

Girdhar, R.; Fouhey, D. F.; Rodriguez, M.; and Gupta, A., 2016. Learning a predictable and generative vector representation for objects. In *ECCV*, 484–499. Springer. (cited on page 152)

Gong, S.; Chen, L.; Bronstein, M.; and Zafeiriou, S., 2019. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 0–0. (cited on page 207)

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y., 2014a. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680. (cited on pages 16, 83, 90, 166, 167, and 207)

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y., 2014b. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, 2672–2680. (cited on page 198)

Goodwin, G. P. and Johnson-Laird, P., 2005. Reasoning about relations. *Psychological review*, 112, 2 (2005), 468. (cited on page 6)

Gould, S.; Hartley, R.; and Campbell, D., 2019. Deep declarative networks: A new hope. *arXiv preprint arXiv:1909.04866*, (2019). (cited on pages 4 and 205)

Gould, S.; Hartley, R.; and Campbell, D. J., 2021. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2021). (cited on pages 4 and 205)

Goyal, A. and Bengio, Y., 2020. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, (2020). (cited on pages 2 and 3)

Grathwohl, W.; Chen, R. T.; Bettencourt, J.; Sutskever, I.; and Duvenaud, D., 2018. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, (2018). (cited on page 118)

Grathwohl, W.; Wang, K.-C.; Jacobsen, J.-H.; Duvenaud, D.; Norouzi, M.; and Swersky, K., 2020. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*. (cited on page 160)

Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; and Wang, J., 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. (cited on page 14)

Guo, X., 1993. Three dimensional moment invariants under rigid transformation. In *International Conference on Computer Analysis of Images and Patterns*, 518–522. Springer. (cited on pages 24 and 64)

Guo, Y.; Bennamoun, M.; Sohel, F.; Lu, M.; Wan, J.; and Kwok, N. M., 2016. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116, 1 (2016), 66–89. (cited on pages 25 and 65)

Györgyi, G. and Tishby, N., 1990. Neural networks and spin glasses ed wk theumann and r köberle. (cited on page 3)

Han, Z.; Liu, Z.; Vong, C.-M.; Liu, Y.-S.; Bu, S.; Han, J.; and Chen, C. P., 2018. Deep spatiality: Unsupervised learning of spatially-enhanced global and local 3d features by deep neural network with coupled softmax. *IEEE Transactions on Image Processing*, 27, 6 (2018), 3049–3063. (cited on page 78)

Han, Z.; Shang, M.; Liu, Y.-S.; and Zwicker, M., 2019. View inter-prediction gan: Unsupervised representation learning for 3d shapes by learning global shape memories to support local view predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 8376–8384. (cited on page 196)

Hayes-Roth, B. and Hayes-Roth, F., 1979. A cognitive model of planning. *Cognitive science*, 3, 4 (1979), 275–310. (cited on page 3)

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. (cited on page 21)

He, Y.; Schiele, B.; and Fritz, M., 2018. Diverse conditional image generation by stochastic regression with latent drop-out codes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 406–421. (cited on page 167)

Henaff, M.; Bruna, J.; and LeCun, Y., 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, (2015). (cited on page 57)

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. In *NeurIPS*, 6626–6637. (cited on page 148)

Hewitt, E. and Savage, L. J., 1955. Symmetric measures on cartesian products. *Transactions of the American Mathematical Society*, 80, 2 (1955), 470–501. (cited on page 139)

Hinton, G. E., 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14, 8 (2002), 1771–1800. (cited on page 4)

Ho, J.; Chen, X.; Srinivas, A.; Duan, Y.; and Abbeel, P., 2019. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, 2722–2730. (cited on page 117)

Hobson, E. W., 1931. *The theory of spherical and ellipsoidal harmonics*. CUP Archive. (cited on page 141)

Holyoak, K. J., 1987. Parallel distributed processing: explorations in the microstructure of cognition. *Science*, 236 (1987), 992–997. (cited on page 3)

Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79, 8 (1982), 2554–2558. (cited on page 3)

Horie, M.; Morita, N.; Ihara, Y.; and Mitsume, N., 2020. Isometric transformation invariant and equivariant graph convolutional networks. *arXiv preprint arXiv:2005.06316*, (2020). (cited on page 11)

Hu, M.-K., 1962. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8, 2 (1962), 179–187. (cited on pages 24 and 64)

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J., 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, (2020). (cited on page 208)

Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A., 2018a. Neural autoregressive flows. In *International Conference on Machine Learning*, 2078–2087. PMLR. (cited on page 20)

Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A., 2018b. Neural autoregressive flows. *ICML*, (2018). (cited on page 125)

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q., 2017a. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708. (cited on page 2)

Huang, J.; Zhang, H.; Yi, L.; Funkhouser, T.; Niessner, M.; and Guibas, L. J., 2019a. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4440–4449. (cited on page 63)

Huang, W.; Lai, B.; Xu, W.; and Tu, Z., 2019b. 3d volumetric modeling with introspective neural networks. In *AAAI*. (cited on pages 147 and 148)

Huang, X.; Li, Y.; Poursaeed, O.; Hopcroft, J.; and Belongie, S., 2017b. Stacked generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 135 and 154)

Huang, X.; Liu, M.-Y.; Belongie, S.; and Kautz, J., 2018c. Multimodal unsupervised image-to-image translation. In *The European Conference on Computer Vision (ECCV)*. (cited on page 199)

Ibarra-Berastegi, G.; Saénz, J.; Esnaola, G.; Ezcurra, A.; and Ulazia, A., 2015. Short-term forecasting of the wave energy flux: Analogues, random forests, and physics-based models. *Ocean Engineering*, 104 (2015), 530–539. (cited on page 4)

Iizuka, S.; Simo-Serra, E.; and Ishikawa, H., 2016. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)*, 35, 4 (2016), 1–11. (cited on pages 167, 170, 174, and 199)

Ilse, M.; Tomczak, J. M.; and Welling, M., 2018. Attention-based deep multiple instance learning. *arXiv preprint arXiv:1802.04712*, (2018). (cited on pages 53 and 55)

Ingraham, J.; Garg, V. K.; Barzilay, R.; and Jaakkola, T. S., 2021. Generative models for graph-based protein design. (2021). (cited on page 208)

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A., 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134. (cited on pages xix, 6, 8, 83, 84, 85, 87, 90, 96, 100, 101, 160, 170, 174, 181, 198, and 199)

Jain, V. and Zhang, H., 2007. A spectral approach to shape-based retrieval of articulated 3d models. *Computer-Aided Design*, 39, 5 (2007), 398–407. (cited on page 64)

Jaini, P.; Selby, K. A.; and Yu, Y., 2019. Sum-of-squares polynomial flow. *arXiv preprint arXiv:1905.02325*, (2019). (cited on pages 18, 20, 118, 125, 126, and 127)

Janssen, M. H.; Janssen, A. J.; Bekkers, E. J.; Bescós, J. O.; and Duits, R., 2018. Design and processing of invertible orientation scores of 3d images. *Journal of mathematical imaging and vision*, 60, 9 (2018), 1427–1458. (cited on page 24)

Javanmard, A. and Montanari, A., 2013. State evolution for general approximate message passing algorithms, with applications to spatial coupling. *Information and Inference: A Journal of the IMA*, 2, 2 (2013), 115–144. (cited on page 4)

Ji, S.; Xu, W.; Yang, M.; and Yu, K., 2012. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35, 1 (2012), 221–231. (cited on page 207)

Jia, X.; Willard, J.; Karpatne, A.; Read, J. S.; Zwart, J. A.; Steinbach, M.; and Kumar, V., 2020. Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *arXiv preprint arXiv:2001.11086*, (2020). (cited on page 4)

Jiang, C.; Wang, D.; Huang, J.; Marcus, P.; Niessner, M.; et al., 2019. Convolutional neural networks on non-uniform geometrical signals using euclidean spectral transformation. *arXiv preprint arXiv:1901.02070*, (2019). (cited on page 63)

Jing, L. and Tian, Y., 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, (2020). (cited on page 160)

Johns, E.; Leutenegger, S.; and Davison, A. J., 2016. Pairwise decomposition of image sequences for active multi-view recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 3813–3822. IEEE. (cited on pages 49, 50, and 75)

Johnson, J.; Alahi, A.; and Fei-Fei, L., 2016a. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 694–711. Springer. (cited on page 104)

Johnson, M. J.; Duvenaud, D.; Wiltschko, A. B.; Datta, S. R.; and Adams, R. P., 2016b. Structured vaes: Composing probabilistic graphical models and variational autoencoders. *arXiv preprint arXiv:1603.06277*, 2 (2016), 2016. (cited on pages 4 and 205)

Junginger, A.; Hanselmann, M.; Strauss, T.; Boblest, S.; Buchner, J.; and Ulmer, H., 2018. Unpaired high-resolution and scalable style transfer using generative adversarial networks. *arXiv preprint arXiv:1810.05724*, (2018). (cited on page 83)

Kac, M., 1938. Une remarque sur les polynomes de M. S. Bernstein. *Studia Math.*, 7 (1938), 49–51. (cited on page 121)

Kahneman, D., 2011. *Thinking, fast and slow*. Macmillan. (cited on page 6)

Kalfaoglu, M. E.; Kalkan, S.; and Alatan, A. A., 2020. Late temporal modeling in 3d cnn architectures with bert for action recognition. In *European Conference on Computer Vision*, 731–747. Springer. (cited on page 1)

Kanezaki, A.; Matsushita, Y.; and Nishida, Y., 2016. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. *arXiv preprint arXiv:1603.06208*, (2016). (cited on page 52)

Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J., 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, (2017). (cited on page 16)

Kauwe, S. K.; Graser, J.; Vazquez, A.; and Sparks, T. D., 2018. Machine learning prediction of heat capacity for solid inorganics. *Integrating Materials and Manufacturing Innovation*, 7, 2 (2018), 43–51. (cited on page 4)

Kazi, A.; Cosmo, L.; Navab, N.; and Bronstein, M., 2020. Differentiable graph module (dgm) graph convolutional networks. *arXiv preprint arXiv:2002.04999*, (2020). (cited on page 208)

Kazi, A.; Shekarforoush, S.; Krishna, S. A.; Burwinkel, H.; Vivar, G.; Kortüm, K.; Ahmadi, S.-A.; Albarqouni, S.; and Navab, N., 2019. Inceptiongcn: receptive field aware graph convolutional network for disease prediction. In *International Conference on Information Processing in Medical Imaging*, 73–85. Springer. (cited on page 208)

Kemp, C. and Tenenbaum, J. B., 2008. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105, 31 (2008), 10687–10692. (cited on page 6)

Khalil, M. I. and Bayoumi, M. M., 2001. A dyadic wavelet affine invariant function for 2d shape recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 10 (2001), 1152–1164. (cited on pages 24 and 64)

Khan, S. H.; Guo, Y.; Hayat, M.; and Barnes, N., 2019. Unsupervised primitive discovery for improved 3d generative modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9739–9748. (cited on pages 135, 138, 147, 152, and 196)

Khan, S. H.; Hayat, M.; and Barnes, N., 2018. Adversarial training of variational auto-encoders for high fidelity image generation. In *Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on*, 1312–1320. IEEE. (cited on pages 25 and 65)

Khrulkov, V. and Oseledets, I., 2018. Geometry score: A method for comparing generative adversarial networks. *arXiv preprint arXiv:1802.02664*, (2018). (cited on pages 88 and 90)

Kim, T. *Anime Sketch Colorization Pair*. https://www.kaggle.com/ktaebum/anime-sketch-colorization-pair. (cited on page 101)

Kingma, D. P. and Dhariwal, P., 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 10215–10224. (cited on page 117)

Kingma, D. P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; and Welling, M., 2016. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, (2016). (cited on page 20)

Kingma, D. P. and Welling, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013). (cited on pages 2, 89, 135, 147, and 159)

Kingma, D. P. and Welling, M., 2014. Auto-encoding variational bayes. *CoRR*, abs/1312.6114 (2014). (cited on page 198)

Klein, F., 1893. A comparative review of recent researches in geometry. *Bulletin of the American Mathematical Society*, 2, 10 (1893), 215–249. (cited on page 11)

Klokov, R. and Lempitsky, V., 2017. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 863–872. IEEE. (cited on pages xix, 49, 50, 61, 62, 75, and 147)

Knothe, H., 1957. Contributions to the theory of convex bodies. *Michigan Mathematical Journal*, 4, 1 (1957), 39–52. (cited on page 20)

Kobyzev, I.; Prince, S.; and Brubaker, M., 2020. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2020), 1—1. (cited on pages 18 and 117)

Kodali, N.; Abernethy, J.; Hays, J.; and Kira, Z., 2017. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, (2017). (cited on pages 160 and 167)

Koller, D.; Friedman, N.; Džeroski, S.; Sutton, C.; McCallum, A.; Pfeffer, A.; Abbeel, P.; Wong, M.-F.; Meek, C.; Neville, J.; et al., 2007. *Introduction to statistical relational learning*. MIT press. (cited on page 7)

Kondor, R., 2018. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, (2018). (cited on page 24)

Kondor, R.; Lin, Z.; and Trivedi, S., 2018. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network. *arXiv preprint arXiv:1806.09231*, (2018). (cited on page 24)

Kosslyn, S. M. and Osherson, D. N., 1995. *An invitation to cognitive science: Visual cognition*, vol. 2. MIT Press. (cited on page 3)

Krizhevsky, A.; Nair, V.; and Hinton, G., 2009. Cifar-10 (canadian institute for advanced research). (2009). http://www.cs.toronto.edu/~kriz/cifar.html. (cited on page 181)

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25 (2012), 1097–1105. (cited on pages 1 and 21)

Krzakala, F.; Moore, C.; Mossel, E.; Neeman, J.; Sly, A.; Zdeborová, L.; and Zhang, P., 2013. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110, 52 (2013), 20935–20940. (cited on page 3)

Ktena, S. I.; Parisot, S.; Ferrante, E.; Rajchl, M.; Lee, M.; Glocker, B.; and Rueckert, D., 2017. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 469–477. Springer. (cited on page 208)

Kukačka, J.; Golkov, V.; and Cremers, D., 2017. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, (2017). (cited on page 2)

Kulon, D.; Wang, H.; Güler, R. A.; Bronstein, M.; and Zafeiriou, S., 2019. Single image 3d hand reconstruction with mesh convolutions. *arXiv preprint arXiv:1905.01326*, (2019). (cited on page 207)

Kumaraswamy, P., 1980. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46, 1–2 (1980), 79–88. (cited on page 125)

Kumawat, S. and Raman, S., 2019. Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4903–4912. (cited on page 75)

Kurtek, S.; Klassen, E.; Ding, Z.; and Srivastava, A., 2010. A novel riemannian framework for shape analysis of 3d objects. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1625–1632. IEEE. (cited on page 57)

Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J., 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40 (2017). (cited on page 3)

Lavoué, G., 2012. Combination of bag-of-words descriptors for robust partial shape retrieval. *The Visual Computer*, 28, 9 (2012), 931–942. (cited on pages 52 and 78)

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al., 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690. (cited on page 14)

Lee, C.-H.; Liu, Z.; Wu, L.; and Luo, P., 2020. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 101)

Lee, H.-Y.; Tseng, H.-Y.; Huang, J.-B.; Singh, M.; and Yang, M.-H., 2018. Diverse image-to-image translation via disentangled representations. In *The European Conference on Computer Vision (ECCV)*. (cited on pages 90, 198, and 199)

Lee, S.; Ha, J.; and Kim, G., 2019a. Harmonizing maximum likelihood with GANs for multimodal conditional generation. In *International Conference on Learning Representations*. (cited on pages 8, 83, 84, 85, 87, 88, and 90)

Lee, S.; Ha, J.; and Kim, G., 2019b. Harmonizing maximum likelihood with GANs for multimodal conditional generation. In *International Conference on Learning Representations*. (cited on pages 160, 167, 174, 198, and 199)

Lees, R. B., 1957. Syntactic structures. (cited on page 7)

Lerer, A.; Gross, S.; and Fergus, R., 2016. Learning physical intuition of block towers by example. In *International conference on machine learning*, 430–438. PMLR. (cited on page 4)

Levie, R.; Huang, W.; Bucci, L.; Bronstein, M. M.; and Kutyniok, G., 2019. Transferability of spectral graph convolutional neural networks. *arXiv preprint arXiv:1907.12972*, (2019). (cited on page 207)

Levie, R.; Monti, F.; Bresson, X.; and Bronstein, M. M., 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67, 1 (2018), 97–109. (cited on page 207)

Li, C.-L.; Zaheer, M.; Zhang, Y.; Poczos, B.; and Salakhutdinov, R., 2018a. Point cloud gan. *arXiv preprint arXiv:1810.05795*, (2018). (cited on pages 138 and 139)

Li, H.-B.; Huang, T.-Z.; Zhang, Y.; Liu, X.-P.; and Gu, T.-X., 2011. Chebyshev-type methods and preconditioning techniques. *Applied Mathematics and Computation*, 218, 2 (2011), 260–270. (cited on pages 34 and 99)

Li, J.; Chen, B. M.; and Lee, G. H., 2018b. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9397–9406. (cited on pages xix, 49, 50, 61, 62, 75, 76, and 147)

Li, J.; Wang, N.; Zhang, L.; Du, B.; and Tao, D., 2020a. Recurrent feature reasoning for image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7760–7768. (cited on page 170)

Li, J.; Zhang, S.; Liu, T.; Ning, C.; Zhang, Z.; and Zhou, W., 2020b. Neural inductive matrix completion with graph convolutional networks for mirna-disease association prediction. *Bioinformatics*, 36, 8 (2020), 2538–2546. (cited on page 208)

Li, Y.; Bu, R.; Sun, M.; and Chen, B., 2018c. Pointcnn. *arXiv preprint arXiv:1801.07791*, (2018). (cited on pages xxv and 76)

Li, Y.; Pirk, S.; Su, H.; Qi, C. R.; and Guibas, L. J., 2016. Fpnn: Field probing neural networks for 3d data. In *Advances in Neural Information Processing Systems*, 307–315. (cited on pages 25 and 65)

Lin, C. and Chellappa, R., 1987. Classification of partial 2-d shapes using fourier descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , 5 (1987), 686–690. (cited on pages 24 and 64)

Litany, O.; Bronstein, A.; Bronstein, M.; and Makadia, A., 2018. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1886–1895. (cited on page 207)

Litman, R.; Bronstein, A.; Bronstein, M.; and Castellani, U., 2014. Supervised learning of bag-of-features shape descriptors using sparse coding. In *Computer Graphics Forum*, vol. 33, 127–136. Wiley Online Library. (cited on page 64)

Liu, W.; Zhang, Y.-M.; Li, X.; Yu, Z.; Dai, B.; Zhao, T.; and Song, L., 2017. Deep hyperspherical learning. In *Advances in Neural Information Processing Systems*, 3950–3960. (cited on pages 54, 55, and 78)

Long, J.; Shelhamer, E.; and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440. (cited on page 2)

Loquercio, A.; Segù, M.; and Scaramuzza, D., 2019. A general framework for uncertainty estimation in deep learning. *arXiv preprint arXiv:1907.06890*, (2019). (cited on page 196)

Lu, Y. and Huang, B., 2020. Woodbury transformations for deep generative flows. *Advances in Neural Information Processing Systems*, 33 (2020). (cited on pages 117 and 118)

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*, 6548–6558. (cited on page 198)

Maaløe, L.; Sønderby, C. K.; Sønderby, S. K.; and Winther, O., 2016. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, (2016). (cited on pages 90, 159, and 200)

Mao, Q.; Lee, H.-Y.; Tseng, H.-Y.; Ma, S.; and Yang, M.-H., 2019. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1429–1437. (cited on pages 84, 88, 90, and 160)

Mao, X.; Li, Q.; Xie, H.; Lau, R. Y.; Wang, Z.; and Paul Smolley, S., 2017. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2794–2802. (cited on page 16)

Maron, H.; Ben-Hamu, H.; Shamir, N.; and Lipman, Y., 2018. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, (2018). (cited on pages 57 and 59)

Marzouk, Y.; Moselhy, T.; Parno, M.; and Spantini, A., 2016. An introduction to sampling via measure transport. *arXiv preprint arXiv:1602.05023*, (2016). (cited on page 20)

Masci, J.; Boscaini, D.; Bronstein, M.; and Vandergheynst, P., 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, 37–45. (cited on page 57)

Mathieu, M.; Couprie, C.; and LeCun, Y., 2015a. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, (2015). (cited on page 85)

Mathieu, M.; Couprie, C.; and LeCun, Y., 2015b. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440 (2015). (cited on page 198)

Maturana, D. and Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 922–928. IEEE. (cited on pages 52, 56, 75, and 76)

Mazoure, B.; Doan, T.; Durand, A.; Pineau, J.; and Hjelm, R. D., 2020. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, 430–444. PMLR. (cited on page 117)

Meagher, D., 1982. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19, 2 (1982), 129–147. (cited on page 61)

Meyers, M.; Weiss, G.; and Spanakis, G., 2020. Fake news detection on twitter using propagation structures. In *Multidisciplinary International Symposium on Disinformation in Open Online Media*, 138–158. Springer. (cited on page 208)

Mirza, M. and Osindero, S., 2014. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784 (2014). (cited on page 198)

Monti, F.; Bronstein, M. M.; and Bresson, X., 2017. Geometric matrix completion with recurrent multi-graph neural networks. *arXiv preprint arXiv:1704.06803*, (2017). (cited on pages 57 and 208)

Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M., 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356, 6337 (2017), 508–513. (cited on page 1)

Nalisnick, E.; Matsukawa, A.; Teh, Y. W.; Gorur, D.; and Lakshminarayanan, B., 2019. Do deep generative models know what they don't know? *International Conference on Learning Representations*, (2019). (cited on page 159)

Navon, D., 1977. Forest before trees: The precedence of global features in visual perception. *Cognitive psychology*, 9, 3 (1977), 353–383. (cited on page 6)

Newell, A. and Simon, H. A., 1961. Gps, a program that simulates human thought. Technical report, RAND CORP SANTA MONICA CALIF. (cited on page 3)

Newell, A.; Simon, H. A.; et al., 1972. *Human problem solving*, vol. 104. Prentice-hall Englewood Cliffs, NJ. (cited on page 3)

Noether, E., 1918. Invariant variation problems. *Transport theory and statistical physics*, 1, 3 (1918), 186–207. (cited on page 11)

Nowack, P.; Braesicke, P.; Haigh, J.; Abraham, N. L.; Pyle, J.; and Voulgarakis, A., 2018. Using machine learning to build temperature-based ozone parameterizations for climate sensitivity simulations. *Environmental Research Letters*, 13, 10 (2018), 104016. (cited on page 4)

Osada, R.; Funkhouser, T.; Chazelle, B.; and Dobkin, D., 2002. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21, 4 (2002), 807–832. (cited on page 25)

Papadakis, P.; Pratikakis, I.; Theoharis, T.; Passalis, G.; and Perantonis, S., 2008. 3d object retrieval using an efficient and compact hybrid shape descriptor. In *Eurographics Workshop on 3D object retrieval*. (cited on pages 52 and 78)

Papamakarios, G.; Pavlakou, T.; and Murray, I., 2017. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, (2017). (cited on page 20)

Parisot, S.; Ktena, S. I.; Ferrante, E.; Lee, M.; Guerrero, R.; Glocker, B.; and Rueckert, D., 2018. Disease prediction using graph convolutional networks: application to autism spectrum disorder and alzheimer's disease. *Medical image analysis*, 48 (2018), 117–130. (cited on page 208)

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. V., 2012. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on page 181)

Pathak, D.; Krähenbühl, P.; Donahue, J.; Darrell, T.; and Efros, A., 2016a. Context encoders: Feature learning by inpainting. (cited on pages xxii, 160, 167, 170, 172, and 198)

Pathak, D.; Krahenbuhl, P.; Donahue, J.; Darrell, T.; and Efros, A. A., 2016b. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2536–2544. (cited on pages xxii, 83, 84, 85, 87, 104, and 172)

Pineda, F. J., 1987. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59, 19 (1987), 2229. (cited on page 205)

Plaut, D. C.; McClelland, J. L.; Seidenberg, M. S.; and Patterson, K., 1996. Understanding normal and impaired word reading: computational principles in quasi-regular domains. *Psychological review*, 103, 1 (1996), 56. (cited on page 6)

Portilla, J. and Simoncelli, E. P., 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, (2000). (cited on page 137)

Prashnani, E.; Cai, H.; Mostofi, Y.; and Sen, P., 2018. Pieapp: Perceptual image-error assessment through pairwise preference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1808–1817. (cited on pages 174 and 180)

Prenger, R.; Valle, R.; and Catanzaro, B., 2019. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3617–3621. IEEE. (cited on page 117)

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1, 2 (2017), 4. (cited on pages xix, 25, 50, 57, 61, 62, 65, 75, 76, and 143)

Qi, C. R.; Su, H.; Niessner, M.; Dai, A.; Yan, M.; and Guibas, L. J., 2016. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5648–5656. (cited on pages 25, 52, and 148)

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 5099–5108. (cited on pages xix, 25, 61, 62, 65, and 76)

Radford, A.; Metz, L.; and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, (2015). (cited on page 16)

Rajan, V. T.; Klinkner, S. R.; and Farouki, R. T., 1988. Root isolation and root approximation for polynomials in Bernstein form. Technical Report RC14224, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, N.Y. 10598. (cited on page 124)

Ramasinghe, S.; Fernando, K.; Khan, S.; and Barnes, N., 2021. Robust normalizing flows using bernstein-type polynomials. *arXiv preprint arXiv:2102.03509*, (2021). (cited on pages 2 and 20)

Ramasinghe, S.; Khan, S.; and Barnes, N., 2019a. Volumetric convolution: Automatic representation learning in unit ball. *arXiv preprint arXiv:1901.00616*, (2019). (cited on pages 24, 63, 64, 65, 66, 67, 77, and 195)

Ramasinghe, S.; Khan, S.; Barnes, N.; and Gould, S., 2019b. Blended convolution and synthesis for efficient discrimination of 3d shapes. *arXiv preprint arXiv:1908.10209*, (2019). (cited on page 25)

Ramasinghe, S.; Khan, S.; Barnes, N.; and Gould, S., 2019c. Representation learning on unit ball with 3d roto-translational equivariance. *International Journal of Computer Vision*, (2019), 1–23. (cited on pages 2, 11, 64, 190, and 207)

Ramasinghe, S.; Khan, S.; Barnes, N.; and Gould, S., 2019d. Spectral-gans for high-resolution 3d point-cloud generation. *arXiv preprint arXiv:1912.01800*, (2019). (cited on pages 90 and 196)

Ramasinghe, S.; Khan, S.; Barnes, N.; and Gould, S., 2020a. Blended convolution and synthesis for efficient discrimination of 3d shapes. In *The IEEE Winter Conference on Applications of Computer Vision*, 21–31. (cited on page 190)

Ramasinghe, S.; Ranasinghe, K.; Khan, S.; Barnes, N.; and Gould, S., 2020b. Conditional generative modeling via learning the latent space. *arXiv preprint arXiv:2010.03132*, (2020). (cited on pages 83, 84, 85, and 87)

Rangan, S.; Fletcher, A. K.; Goyal, V. K.; and Schniter, P., 2012. Hybrid generalized approximate message passing with applications to structured sparsity. In *2012 IEEE International Symposium on Information Theory Proceedings*, 1236–1240. IEEE. (cited on page 4)

Rao, Y.; Lu, J.; and Zhou, J., 2019. Spherical fractal convolutional neural networks for point cloud recognition. In *CVPR*, 452–460. (cited on page 11)

Redlich, A. N., 1993. Supervised factorial learning. *Neural Computation*, 5, 5 (1993), 750–766. (cited on page 20)

Reininghaus, J.; Huber, S.; Bauer, U.; and Kwitt, R., 2015. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4741–4748. (cited on page 59)

Reiss, T., 1992. Features invariant to linear transformations in 2d and 3d. In *11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis*,, 493–496. IEEE. (cited on pages 24 and 64)

Rezende, D. and Mohamed, S., 2015. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 1530–1538. PMLR. (cited on pages 2, 18, 90, 117, 159, and 200)

Rico-Martinez, R. and Kevrekidis, I. G., 1993. Continuous time modeling of nonlinear systems: A neural network-based approach. In *IEEE International Conference on Neural Networks*, 1522–1525. IEEE. (cited on page 205)

Riegler, G.; Osman Ulusoy, A.; and Geiger, A., 2017. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3577–3586. (cited on page 61)

Rippel, O. and Adams, R. P., 2013. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, (2013). (cited on page 18)

Robbins, H. E., 2007. A stochastic approximation method. *Annals of Mathematical Statistics*, 22 (2007), 400–407. (cited on page 161)

Ronchi, C.; Iacono, R.; and Paolucci, P. S., 1996. The "cubed sphere": a new method for the solution of partial differential equations in spherical geometry. *Journal of Computational Physics*, 124, 1 (1996), 93–114. (cited on page 21)

Ronneberger, O.; Fischer, P.; and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer. (cited on page 104)

Rosenblatt, M., 1952. Remarks on a multivariate transformation. *The annals of mathematical statistics*, 23, 3 (1952), 470–472. (cited on page 20)

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M., 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, (2020). (cited on page 208)

Royer, A.; Bousmalis, K.; Gouws, S.; Bertsch, F.; Mosseri, I.; Cole, F.; and Murphy, K., 2020. Xgan: Unsupervised image-to-image translation for many-to-many mappings. In *Domain Adaptation for Visual Understanding*, 33–49. Springer. (cited on page 14)

Russell, S. and Norvig, P., 2002. Artificial intelligence: a modern approach. (2002). (cited on page 7)

Rustamov, R. M., 2007. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, 225–233. Eurographics Association. (cited on page 64)

Ryan, K.; Lengyel, J.; and Shatruk, M., 2018. Crystal structure prediction via deep learning. *Journal of the American Chemical Society*, 140, 32 (2018), 10158–10168. (cited on page 4)

Sagong, M.-c.; Shin, Y.-g.; Kim, S.-w.; Park, S.; and Ko, S.-j., 2019. Pepsi : Fast image inpainting with parallel decoding network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 199)

Saito, M. and Saito, S., 2018. Tganv2: Efficient training of large models for video generation with multiple subsampling layers. (2018). (cited on page 14)

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X., 2016. Improved techniques for training gans. In *NeurIPS*, 2234–2242. (cited on page 180)

Sanchez-Gonzalez, A.; Bapst, V.; Cranmer, K.; and Battaglia, P., 2019. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, (2019). (cited on page 206)

Sauder, J. and Sievers, B., 2019. Self-supervised deep learning on point clouds by reconstructing space. In *Advances in Neural Information Processing Systems*, 12942–12952. (cited on page 196)

Schank, R. C., 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive psychology*, 3, 4 (1972), 552–631. (cited on page 3)

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural networks*, 61 (2015), 85–117. (cited on page 3)

Schmidt, V.; Luccioni, A.; Mukkavilli, S. K.; Balasooriya, N.; Sankaran, K.; Chayes, J.; and Bengio, Y., 2019. Visualizing the consequences of climate change using cycle-consistent adversarial networks. *arXiv preprint arXiv:1905.03709*, (2019). (cited on page 4)

Sedaghat, N.; Zolfaghari, M.; Amiri, E.; and Brox, T., 2016. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, (2016). (cited on page 52)

Seung, H. S.; Sompolinsky, H.; and Tishby, N., 1992. Statistical mechanics of learning from examples. *Physical review A*, 45, 8 (1992), 6056. (cited on page 3)

Shaham, T. R.; Dekel, T.; and Michaeli, T., 2019. Singan: Learning a generative model from a single natural image. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 154)

Shanahan, M., 2006. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and cognition*, 15, 2 (2006), 433–449. (cited on page 7)

Shanahan, M., 2010. *Embodiment and the inner life: Cognition and Consciousness in the Space of Possible Minds*. Oxford University Press, USA. (cited on page 7)

Shanahan, M., 2012. The brain's connective core and its role in animal cognition. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367, 1603 (2012), 2704–2714. (cited on page 7)

Shao, H.; Kumar, A.; and Thomas Fletcher, P., 2018. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 315–323. (cited on pages 84, 88, and 90)

Sharma, A.; Grau, O.; and Fritz, M., 2016. Vconv-dae: Deep volumetric shape learning without object labels. In *ECCV*, 236–250. Springer. (cited on pages 147 and 196)

Shi, B.; Bai, S.; Zhou, Z.; and Bai, X., 2015. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22, 12 (2015), 2339–2343. (cited on pages 49, 50, and 75)

Shu, D. W.; Park, S. W.; and Kwon, J., 2019. 3d point cloud generative adversarial network based on tree structured graph convolutions. *arXiv preprint arXiv:1905.06292*, (2019). (cited on pages 135, 137, 138, and 149)

Shuman, D. I., 2020. Localized spectral graph filter frames: A unifying framework, survey of design considerations, and numerical comparison. *IEEE Signal Processing Magazine*, 37, 6 (2020), 43–63. (cited on page 207)

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al., 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529, 7587 (2016), 484–489. (cited on page 1)

Simonovsky, M. and Komodakis, N., 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proc. CVPR*. (cited on pages 50, 75, and 147)

Smith, E. and Meger, D., 2017. Improved adversarial systems for 3d object generation and reconstruction. *arXiv preprint arXiv:1707.09557*, (2017). (cited on pages 138 and 152)

Smolensky, P., 1986. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science. (cited on page 4)

Sohn, K.; Lee, H.; and Yan, X., 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*. (cited on page 198)

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*, (2020). (cited on page 206)

Sosnovik, I.; Szmaja, M.; and Smeulders, A., 2019. Scale-equivariant steerable networks. *arXiv preprint arXiv:1910.11093*, (2019). (cited on page 11)

Souza, R. and Frayne, R., 2018. A hybrid frequency-domain/image-domain deep network for magnetic resonance image reconstruction. *arXiv preprint arXiv:1810.12473*, (2018). (cited on page 137)

Spelke, E. S., 1990. Principles of object perception. *Cognitive science*, 14, 1 (1990), 29–56. (cited on page 3)

Spencer, M. R., 1994. *Polynomial Real Root Finding in Bernstein Form*. Ph.D. thesis, Brigham Young University. (cited on pages 124 and 126)

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15, 1 (2014), 1929–1958. (cited on pages 2 and 83)

Stokes, J. M.; Yang, K.; Swanson, K.; Jin, W.; Cubillos-Ruiz, A.; Donghia, N. M.; MacNair, C. R.; French, S.; Carfrae, L. A.; Bloom-Ackermann, Z.; et al., 2020. A deep learning approach to antibiotic discovery. *Cell*, 180, 4 (2020), 688–702. (cited on page 208)

Su, H.; Jampani, V.; Sun, D.; Maji, S.; Kalogerakis, E.; Yang, M.-H.; and Kautz, J., 2018. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2530–2539. (cited on page 59)

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, 945–953. (cited on pages xix, 50, 62, and 75)

Suk, T. and Flusser, J., 1996. Vertex-based features for recognition of projectively deformed polygons. *Pattern Recognition*, 29, 3 (1996), 361–367. (cited on pages 24 and 64)

Sutskever, I.; Vinyals, O.; and Le, Q. V., 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, (2014). (cited on page 1)

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A., 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31. (cited on page 1)

Tabak, E. G. and Turner, C. V., 2013. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66, 2 (2013), 145–164. (cited on page 18)

Tabia, H.; Laga, H.; Picard, D.; and Gosselin, P.-H., 2014. Covariance descriptors for 3d shape matching and retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4185–4192. (cited on pages 52 and 78)

Tabia, H.; Picard, D.; Laga, H.; and Gosselin, P.-H., 2013. Compact vectors of locally aggregated tensors for 3d shape retrieval. In *Eurographics workshop on 3D object retrieval*. (cited on page 52)

Talagrand, M., 1996. Transportation cost for gaussian and other product measures. *Geometric & Functional Analysis GAFA*, 6, 3 (1996), 587–600. (cited on page 20)

Tatsuma, A. and Aono, M., 2009. Multi-fourier spectra descriptor and augmentation with spectral clustering for 3d shape retrieval. *The Visual Computer*, 25, 8 (Aug 2009), 785–804. (cited on pages 52 and 80)

Thanh-Tung, H.; Tran, T.; and Venkatesh, S., 2019. Improving generalization and stability of generative adversarial networks. *arXiv preprint arXiv:1902.03984*, (2019). (cited on page 160)

Thomas, N.; Smidt, T.; Kearnes, S.; Yang, L.; Li, L.; Kohlhoff, K.; and Riley, P., 2018. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, (2018). (cited on page 24)

Tieng, Q. M. and Boles, W. W., 1995. An application of wavelet-based affine-invariant representation. *Pattern Recognition Letters*, 16, 12 (1995), 1287–1296. (cited on pages 24 and 64)

Tombari, F.; Salti, S.; and Di Stefano, L., 2010. Unique signatures of histograms for local surface description. In *European conference on computer vision*, 356–369. Springer. (cited on page 25)

Tramel, E. W.; Gabrié, M.; Manoel, A.; Caltagirone, F.; and Krzakala, F., 2018. Deterministic and generalized framework for unsupervised learning with restricted boltzmann machines. *Physical Review X*, 8, 4 (2018), 041006. (cited on page 4)

Tulyakov, S.; Liu, M.-Y.; Yang, X.; and Kautz, J., 2018. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1526–1535. (cited on page 14)

TURING, A. M., 1950. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX, 236 (10 1950), 433–460. doi:10.1093/mind/LIX.236.433. https://doi.org/10.1093/mind/LIX.236.433. (cited on page 3)

Tyleček, R. and Šára, R., 2013. Spatial pattern templates for recognition of objects with regular structure. In *Proc. GCPR*. Saarbrucken, Germany. (cited on page 181)

Uria, B.; Côté, M.-A.; Gregor, K.; Murray, I.; and Larochelle, H., 2016. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17, 1 (2016), 7184–7220. (cited on page 20)

Valiant, L. G., 1984. A theory of the learnable. *Communications of the ACM*, 27, 11 (1984), 1134–1142. (cited on page 3)

Valsesia, D.; Fracastoro, G.; and Magli, E., 2018. Learning localized generative models for 3d point clouds via graph convolution. (2018). (cited on pages 135, 137, 138, 149, and 157)

Veselkov, K.; Gonzalez, G.; Aljifri, S.; Galea, D.; Mirnezami, R.; Youssef, J.; Bronstein, M.; and Laponogov, I., 2019. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Scientific reports*, 9, 1 (2019), 1–12. (cited on page 208)

Vidaurre, R.; Santesteban, I.; Garces, E.; and Casas, D., 2020. Fully convolutional graph neural networks for parametric virtual try-on. In *Computer Graphics Forum*, vol. 39, 145–156. Wiley Online Library. (cited on page 207)

Villani, C., 2009. *Optimal Transport: Old and New*. Springer. (cited on page 124)

Vitoria, P.; Raad, L.; and Ballester, C., 2020. Chromagan: Adversarial picture colorization with semantic class distribution. In *The IEEE Winter Conference on Applications of Computer Vision*, 2445–2454. (cited on pages 160, 167, 170, 174, 198, and 199)

Voronovskaya, E., 1932. Détermination de la forme asymptotique d'approximation des fonctions par les polynômes de M. Bernstein. *Doklady Akademii Nauk SSSR*, (1932), 79—-85. (cited on page 121)

Vranic, D. V. and Saupe, D., 2002. Description of 3d-shape using a complex function on the sphere. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, vol. 1, 177–180. IEEE. (cited on pages 25 and 65)

Vranic, D. V.; Saupe, D.; and Richter, J., 2001. Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, 293–298. IEEE. (cited on page 64)

Wang, C.; Samari, B.; and Siddiqi, K., 2018. Local spectral graph convolution for point set feature learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 52–66. (cited on page 76)

Wang, F.; Liu, H.; Samaras, D.; and Chen, C., 2020a. Topogan: A topology-aware generative adversarial network. In *European Conference on Computer Vision*. (cited on pages 88 and 90)

Wang, P.-W.; Donti, P.; Wilder, B.; and Kolter, Z., 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, 6545–6554. PMLR. (cited on pages 205 and 206)

Wang, T.; Liu, M.; Zhu, J.; Tao, A.; Kautz, J.; and Catanzaro, B., 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8798–8807. (cited on pages 83, 85, and 154)

Wang, W.; Axelrod, S.; and Gómez-Bombarelli, R., 2020b. Differentiable molecular simulations for control and learning. *arXiv preprint arXiv:2003.00868*, (2020). (cited on page 206)

Wang, X. and Gupta, A., 2016. Generative image modeling using style and structure adversarial networks. In *ECCV*. (cited on page 154)

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M., 2018a. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, (2018). (cited on pages 57 and 76)

Wang, Y.; Tao, X.; Qi, X.; Shen, X.; and Jia, J., 2018b. Image inpainting via generative multi-column convolutional neural networks. In *Advances in Neural Information Processing Systems 31*. (cited on page 199)

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P., 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13, 4 (2004), 600–612. (cited on page 174)

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 & Computers, 2003*, vol. 2, 1398–1402. Ieee. (cited on page 174)

Ward, P. N.; Smofsky, A.; and Bose, A. J., 2019. Improving exploration in soft-actor-critic with normalizing flows policies. *arXiv preprint arXiv:1906.02771*, (2019). (cited on page 117)

Wei, H.; Zhao, S.; Rong, Q.; and Bao, H., 2018. Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods. *International Journal of Heat and Mass Transfer*, 127 (2018), 908–916. (cited on page 4)

Weiler, M.; Geiger, M.; Welling, M.; Boomsma, W.; and Cohen, T., 2018a. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, (2018). (cited on page 24)

Weiler, M.; Geiger, M.; Welling, M.; Boomsma, W.; and Cohen, T., 2018b. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems 31* (Eds. S. Bengio; H. Wallach; H. Larochelle; K. Grauman; N. Cesa-Bianchi; and R. Garnett), 10381–10392. Curran Associates, Inc. http://papers.nips.cc/paper/8239-3d-steerable-cnns-learning-rotationally-equivariant-features-in-volumetric-data.pdf. (cited on page 65)

Welling, M., 2019. Do we still need models or just more data and compute? *University of Amsterdam, April*, 20 (2019). (cited on pages 1 and 2)

Wellman, H. M. and Gelman, S. A., 1992. Cognitive development: Foundational theories of core domains. *Annual review of psychology*, 43, 1 (1992), 337–375. (cited on page 3)

Weyl, H., 1929. Elektron und gravitation. i. *Zeitschrift für Physik*, 56, 5-6 (1929), 330–352. (cited on page 11)

Willard, J.; Jia, X.; Xu, S.; Steinbach, M.; and Kumar, V., 2020. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, (2020). (cited on page 4)

Wirnsberger, P.; Ballard, A. J.; Papamakarios, G.; Abercrombie, S.; Racanière, S.; Pritzel, A.; Jimenez Rezende, D.; and Blundell, C., 2020. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153, 14 (2020), 144112. (cited on page 117)

Wolpert, D. H., 1996. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8, 7 (1996), 1341–1390. (cited on page 2)

Wong, K. W.; Contardo, G.; and Ho, S., 2020. Gravitational-wave population inference with deep flow-based generative network. *Physical Review D*, 101, 12 (2020), 123005. (cited on page 117)

Worrall, D. E. and Brostow, G. J., 2018. Cubenet: Equivariance to 3d rotation and translation. *European Conference on Computer Vision*, (2018). (cited on page 24)

Worrall, D. E.; Garbin, S. J.; Turmukhambetov, D.; and Brostow, G. J., 2017. Harmonic networks: Deep translation and rotation equivariance. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 7168–7177. IEEE. (cited on page 24)

Wu, J.; Zhang, C.; Xue, T.; Freeman, B.; and Tenenbaum, J., 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, 82–90. (cited on pages 25, 52, 61, 65, 75, 135, 138, 147, 148, and 152)

Wu, Y. and He, K., 2018. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19. (cited on page 74)

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920. (cited on pages 25, 49, 50, 75, 76, 135, 144, and 147)

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*, 3485–3492. IEEE. (cited on page 151)

Xie, J.; Fang, Y.; Zhu, F.; and Wong, E., 2015. Deepshape: Deep learned shape descriptor for 3d shape matching and retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 1275–1283. IEEE. (cited on pages 52, 64, and 78)

Xie, J.; Wang, M.; and Fang, Y., 2016. Learned binary spectral shape descriptor for 3d shape correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3309–3317. (cited on page 64)

Xie, J.; Zheng, Z.; Gao, R.; Wang, W.; Zhu, S.-C.; and Nian Wu, Y., 2018a. Learning descriptor networks for 3d shape synthesis and analysis. In *CVPR*, 8629–8638. (cited on pages 135, 147, and 148)

Xie, Y.; Franz, E.; Chu, M.; and Thuerey, N., 2018b. tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow. *ACM Transactions on Graphics (TOG)*, 37, 4 (2018), 95. (cited on page 198)

Yang, B.; Flusser, J.; and Suk, T., 2015. 3d rotation invariants of gaussian–hermite moments. *Pattern Recognition Letters*, 54 (2015), 18–26. (cited on page 24)

Yang, C.; Lu, X.; Lin, Z.; Shechtman, E.; Wang, O.; and Li, H., 2017a. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6721–6729. (cited on page 83)

Yang, C.-N. and Mills, R. L., 1954. Conservation of isotopic spin and isotopic gauge invariance. *Physical review*, 96, 1 (1954), 191. (cited on page 11)

Yang, D.; Hong, S.; Jang, Y.; Zhao, T.; and Lee, H., 2019a. Diversity-sensitive conditional generative adversarial networks. *arXiv preprint arXiv:1901.09024*, (2019). (cited on pages xix, 84, 85, 87, 88, 90, 96, and 170)

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B., 2019b. Pointflow: 3d point cloud generation with continuous normalizing flows. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 135)

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B., 2019c. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4541–4550. (cited on page 206)

Yang, G.; Yu, S.; Dong, H.; Slabaugh, G.; Dragotti, P. L.; Ye, X.; Liu, F.; Arridge, S.; Keegan, J.; Guo, Y.; et al., 2017b. Dagan: deep de-aliasing generative adversarial networks for fast compressed sensing mri reconstruction. *IEEE transactions on medical imaging*, 37, 6 (2017), 1310–1321. (cited on page 137)

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D., 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 206–215. (cited on page 196)

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; Guibas, L.; et al., 2016. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35, 6 (2016), 210. (cited on page 148)

Yi, L.; Su, H.; Guo, X.; and Guibas, L. J., 2017. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2282–2290. (cited on page 207)

Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; and Huang, T. S., 2018a. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5505–5514. (cited on page 170)

Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; and Huang, T. S., 2018b. Generative image inpainting with contextual attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 199)

Yu, T.; Meng, J.; and Yuan, J., 2018c. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 186–194. (cited on pages 75 and 76)

Yu, W.; Tan, J.; Liu, C. K.; and Turk, G., 2017. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, (2017). (cited on page 2)

Zdeborová, L. and Krzakala, F., 2016. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65, 5 (2016), 453–552. (cited on page 3)

Zeinali, H.; Wang, S.; Silnova, A.; Matějka, P.; and Plchot, O., 2019. But system description to voxceleb speaker recognition challenge 2019. *arXiv preprint arXiv:1910.12592*, (2019). (cited on page 1)

Zeng, H.; Zhang, R.; Wang, X.; Fu, D.; and Wei, Q., 2018. Dempster–shafer evidence theory-based multi-feature learning and fusion method for non-rigid 3d model retrieval. *IET Computer Vision*, 13, 3 (2018), 261–266. (cited on page 78)

Zeng, Y.; Fu, J.; Chao, H.; and Guo, B., 2019a. Learning pyramid-context encoder network for high-quality image inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1486–1494. (cited on page 83)

Zeng, Y.; Fu, J.; Chao, H.; and Guo, B., 2019b. Learning pyramid-context encoder network for high-quality image inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1486–1494. (cited on pages 90, 160, 167, 170, 198, and 199)

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D., 2017a. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, (2017). (cited on page 2)

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A., 2019. Self-attention generative adversarial networks. In *International conference on machine learning*, 7354–7363. PMLR. (cited on page 16)

Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; and Metaxas, D. N., 2017b. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 154)

Zhang, L.; Zhang, L.; Mou, X.; and Zhang, D., 2011. Fsim: A feature similarity index for image quality assessment. *IEEE transactions on Image Processing*, 20, 8 (2011), 2378–2386. (cited on page 174)

Zhang, R.; Isola, P.; and Efros, A. A., 2016. Colorful image colorization. In *European conference on computer vision*, 649–666. Springer. (cited on pages 90, 159, 160, 170, 174, 180, 198, and 199)

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O., 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*. (cited on pages 174 and 180)

Zhang, S. Y., Zhifei and Qi, H., 2017. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. (cited on pages 101 and 181)

Zhang, X.; Wang, X.; Kong, B.; Yin, Y.; Song, Q.; Lyu, S.; Lv, J.; Shi, C.; and Li, X., 2020. Domain embedded multi-model generative adversarial networks for image-based face inpainting. *ArXiv*, abs/2002.02909 (2020). (cited on page 199)

Zhi, S.; Liu, Y.; Li, X.; and Guo, Y., 2017. Lightnet: A lightweight 3d convolutional neural network for real-time 3d object recognition. In *3DOR*. (cited on page 147)

Zhong, Y. D.; Dey, B.; and Chakraborty, A., 2019. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, (2019). (cited on page 206)

Zhu, B.; Liu, J. Z.; Cauley, S. F.; Rosen, B. R.; and Rosen, M. S., 2018. Image reconstruction by domain-transform manifold learning. *Nature*, 555, 7697 (2018), 487. (cited on page 137)

Zhu, J.-Y.; Krähenbühl, P.; Shechtman, E.; and Efros, A. A., 2016. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*. (cited on page 199)

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A., 2017a. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2223–2232. (cited on page 83)

Zhu, J.-Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A. A.; Wang, O.; and Shechtman, E., 2017b. Toward multimodal image-to-image translation. In *Advances in neural information processing systems*, 465–476. (cited on pages xix, 85, 87, 90, 96, and 170)

Zhu, J.-Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A. A.; Wang, O.; and Shechtman, E., 2017c. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems 30*. (cited on pages 174 and 198)

Zitnik, M.; Agrawal, M.; and Leskovec, J., 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34, 13 (2018), i457–i466. (cited on page 208)