DEEP GENERATIVE MODELS: PITFALLS AND FIXES

A Dissertation

by

MOHAMMADREZA ARMANDPOUR

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jianhua Z. Huang
Co-Chair of Committee,	Debdeep Pati
Committee Members,	Xia (Ben) Hu
	Yang Ni
Head of Department,	Brani Vidakovic

May 2022

Major Subject: Statistics

ABSTRACT

The promise of deep learning is to discover rich, hierarchical models that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. In recent years, the most striking successes in deep learning have involved generative models. However, in their vanilla forms, generative models have a number of shortcomings and failure modes that can hinder their application: they can be difficult to train on high dimensional data, and they can fail in tasks such as the generation of realistic artificial data. In this thesis, we first explore the reasons for these failures in the adversarial-based generative models and propose a novel approach to alleviate these shortfalls. Then, we discuss how a learned generative model can be employed for a downstream task such as speech recognition.

DEDICATION

To my mother, father, sister (Anna), and friend (Ali).

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my advisor Dr. Jianhua Huang. Back in 2016, I stepped into the statistics realm and was very new to the independent research. I will never forget the patience he had when I was in difficulties, the firm confidence he gave when I was hesitating and the moments we cheered together for new discoveries.

I am also grateful to Dr. Debdeep Pati, for introducing me to the deep learning and nonpramteric Bayes and giving me fundamental problems to work on. He made me to be more confident that I also can tackle important problems and taught me how can I become an independent researcher.

This work would not be possible without the supervision of Dr. Mingyuan Zhou. I am lucky to have him as a mentor and friend. I have grown as a statistician, and as a person, thanks to his guidance and encouragement. I will miss having meeting with a world-class machine learning expert, but I will miss my friend and mentor even more.

I owe a debt of gratitude to Dr. Xia Hu, for his guidance and encouragement. Dr. Hu is acclaimed for his contributions to the field of machine learning, but he deserves equal recognition for his ability to inspire and mentor budding ML scientists, and challenge them to accomplish more than they thought possible.

My committee member, Dr. Yang Ni, deserves extra thanks for always been extremely generous with his time, being friendly, and introducing me to several interesting topics in statistics. I would also like to thank my colleagues, Yabo Niu, Daniel Zilber, Xiaomeng Yan, Hanxuan Ye, Jacob Helwig, and Weiwei Wang for sharing their time and wisdom, and being friend. There is no room to name everyone, but I thank you all.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Dr. Jianhua Z. Huang as committee chair, Dr. Debdeep Pati as committee co-chair, and Dr. Yang Ni of the Department of Statistics and Dr. Xia (Ben) Hu of the Department of Computer Science.

The data analyzed in this thesis are all publicly available as benchmark data sets. The second chapter is the result of collaboration with Dr. Mingyuan Zhou from The University of the Texas at Austin and Chunyuan Li from Microsoft research that is published by CVPR 2021 [1]. And the third chapter is the results of my collaboration with Apple (MIND team) during my internship that is accepted by ICASSP 2022 [2].

Funding Sources

Graduate study was supported by the Department of Statistics at Texas A&M University.

NOMENCLATURE

GAN(s)	Generative Adversarial Networks
ASR	Automatic Speech Recognition
PGMGAN	Partition-Guided Mixture of GAN
IS	Inception Score
FID	Frechet Inception Distance
NMI	Normalized Mutual Information
JSD	Jensen–Shannon distance
ELU	Exponential Linear Unit
LeakyReLU	Leaky Rectified Linear Unit
ReLU	Rectified Linear Unit
SGD	Stochastic gradient descent
BN	Batch Normalization
TTS	Text-to-Speech
CE	Cross-entropy
AUC	Area under the curve
FAR	False accept rate
DET	Detection error tradeoff
FRR	False reject rate
WER	Word error rate
DNN	Deep neural network

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Why do we need generative models?

One of the core aspirations at Machine learning community is to develop algorithms and techniques that endow computers with an understanding of our world. It's easy to forget just how much you know about the world: you understand that it is made up of 3D environments, objects that move, collide, interact; people who walk, talk, and think; animals who graze, fly, run, or bark; monitors that display information encoded in language about the weather, who won a basketball game, or what happened in 1970.

This tremendous amount of information is out there and to a large extent easily accessible — either in the physical world of atoms or the digital world of bits. The only tricky part is to develop models and algorithms that can analyze and understand this treasure trove of data.

Generative models are one of the most promising approaches towards this goal. To train a generative model we first collect a large amount of data in some domain (e.g., think millions of images, sentences, or sounds, etc.) and then train a model to generate data like it. The intuition behind this approach follows a famous quote from Richard Feynman:

"What I cannot create, I do not understand."
—Richard Feynman

The trick is that the neural networks we use as generative models have a number of parameters significantly smaller than the amount of data we train them on, so the models are forced to discover and efficiently internalize the essence of the data in order to generate it. Generative models have many short-term applications. But in the long run, they hold the potential to automatically learn the natural features of a dataset, whether categories or dimensions or something else entirely.

In this thesis, we first study the shortcomings of the generative models. And we continue with

how efficiently one can employ a generative model in a real world problem more specifically in the speech recognition task. To begin let's start with the formal definition of generative adversarial networks (GANs)[4] which is one of the most studied type of generative models.

1.2 Generative Adversarial Networks

While most deep generative models are trained by maximizing log likelihood or a lower bound on log likelihood, GANs take a radically different approach that does not require inference or explicit calculation of the data likelihood. Instead, two models are used to solve a minimax game: a generator which samples data, and a discriminator which classifies the data as real or generated. In theory these models are capable of modeling an arbitrarily complex probability distribution. When using the optimal discriminator for a given class of generators, the original GAN proposed by Goodfellow et al. minimizes the Jensen-Shannon divergence between the data distribution and the generator, and extensions generalize this to a wider class of divergences.

The ability to train extremely flexible generating functions, without explicitly computing likelihoods or performing inference, and while targeting more mode-seeking divergences as made GANs extremely successful in image generation, and image super resolution. The flexibility of the GAN framework has also enabled a number of successful extensions of the technique, for instance for structured prediction [5], training energy based models, and combining the GAN loss with a mutual information loss.

The GAN learning problem is to find the optimal parameters θ_G^* for a generator function $G\left(z;\theta_G\right)$ in a minimax objective,

$$\theta_G^* = \underset{\theta_G}{\operatorname{argmin}} \max_{\theta_D} f\left(\theta_G, \theta_D\right) \tag{1.1}$$

$$= \underset{\theta_G}{\operatorname{argmin}} f\left(\theta_G, \theta_D^*\left(\theta_G\right)\right) \tag{1.2}$$

$$\theta_D^*\left(\theta_G\right) = \underset{\theta_D}{\operatorname{argmax}} f\left(\theta_G, \theta_D\right), \tag{1.3}$$

where f is commonly chosen to be

$$f\left(\theta_G, \theta_D\right) = \mathbb{E}_{x \sim p_{data}}\left[\log\left(D\left(x; \theta_D\right)\right)\right] + \mathbb{E}_{z \sim \mathcal{N}(0,I)}\left[\log\left(1 - D\left(G\left(z; \theta_G\right); \theta_D\right)\right)\right]. \tag{1.4}$$

Here $x \in \mathcal{X}$ is the data variable, $z \in \mathcal{Z}$ is the latent variable, p_{data} is the data distribution, the discriminator $D\left(\cdot; \theta_D\right) : \mathcal{X} \to [0, 1]$ outputs the estimated probability that a sample x comes from the data distribution, θ_D and θ_G are the discriminator and generator parameters, and the generator function $G\left(\cdot; \theta_G\right) : \mathcal{Z} \to \mathcal{X}$ transforms a sample in the latent space into a sample in the data space.

1.3 Outline

Despite the success of Generative Adversarial Networks (GANs), their training suffers from several well-known problems, including mode collapse and having difficulties learning a disconnected set of manifolds. In the next chapter, we break down the challenging task of learning complex high dimensional distributions supporting diverse data samples to simpler sub-tasks. Our solution relies on designing a partitioner that breaks the space into smaller regions, each having a simpler distribution, and training a different generator for each partition. We formulate two desired criteria for the space partitioner that aids the training of our mixture of generators: 1) it produces connected partitions and 2) provides a proxy of distance between partitions and data samples along with a direction to reduce that distance. These criteria are developed to not only avoid producing samples from places with non-existent data density but also facilitate training by providing direction to the generators from both the space partitioner and discriminator. We develop theoretical constraints for a space partitioner to satisfy the above criteria. Guided by our theoretical analysis, we further design an effective neural architecture for the space partitioner that empirically assures these conditions. Experimental results on various standard benchmarks show that the proposed unsupervised model outperforms several recent methods.

Then we continue with how one can employ synthetic data (produced by a generative model) as a viable alternative to real data for training speech recognition models. However, machine learning with synthetic data is not trivial due to the gap between the synthetic and the real data distributions.

Synthetic datasets may contain artifacts that do not exist in real data such as structured noise, content errors, or unrealistic speaking styles. Moreover, the synthesis process may introduce a bias due to uneven sampling of the data manifold. We propose two novel techniques during training to mitigate the problems due to the distribution gap: (i) a rejection sampling algorithm and (ii) using separate batch normalization statistics for the real and the synthetic samples. We show that these methods significantly improve the training of speech recognition models using synthetic data. We evaluate the proposed approach on keyword detection and Automatic Speech Recognition (ASR) tasks, and observe up to 18% and 13% relative error reduction, respectively, compared to naively using the synthetic data.

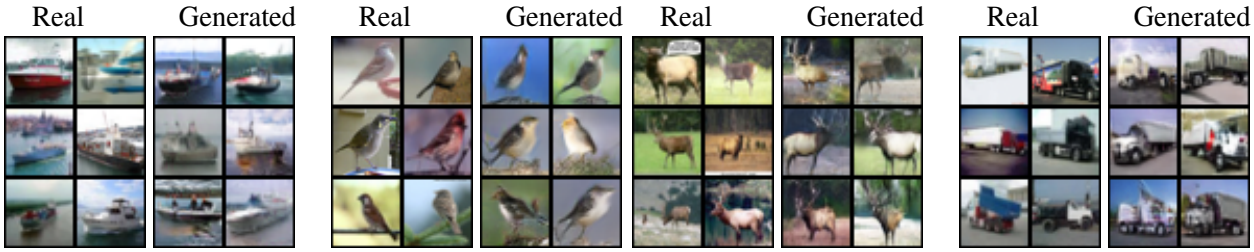
2. PARTITION-GUIDED GANS

2.1 Introduction

Generative adversarial networks (GANs) [4] have gained remarkable success in learning the underlying distribution of observed samples. However, their training is still unstable and challenging, especially when the data distribution of interest is multimodal. This is particularly important due to both empirical and theoretical evidences that suggest real data also conforms to such distributions [6, 7].

Improving the vanilla GAN, both in term of training stability and generating high fidelity images, has been the subject of great interest in the machine learning literature [8, 9, 10, 11, 12, 13, 14, 15]. One of the main problems is *mode collapse*, where the generator fails to capture the full diversity of the data. Another problem, which hasn't been fully explored, is the *mode connecting* problem [16, 17]. As we explain in detail in Section 2.2, this phenomena occurs when the GAN generates samples from parts of the space where the true data is non-existent; caused by using a continuous generator to approximate a distribution with disconnected support. Moreover, GANs are also known to be hard to train due to unreliable gradient provided by the discriminator.

Our solution to alleviate the aforementioned problems is to introduce an unsupervised space partitioner and train a different generator for each partition. Figure 2.1 illustrates real and generated



(a) Partition 13 (b) Partition 47 (c) Partition 69 (d) Partition 85

Figure 2.1: Examples of unsupervised partitioning and their corresponding real/generated samples on CIFAR-10 dataset.

samples from several inferred partitions.

Having multiple generators, which are focused on different parts/modes of the distribution, reduces the chances of missing a mode. This also mitigates mode connecting, because, the mixture of generators is no longer restricted to be a continuous function responsible to generate from a data distribution with potentially disconnected manifolds. In this context, an effective space partitioner should place disconnected data manifolds in different partitions. Therefore, assuming semantically similar images are in the same connected manifold, we use contrastive learning methods to learn semantic representations of images and partition the space using these embeddings.

We show that the space partitioner can be utilized to define a distance between points in the data space and partitions. The gradient of this distance can be used to encourage each generator to focus on its corresponding region by providing a direction to guide it there. In other words, by penalizing a generator when its samples are far from its partition, the space partitioner can *guide* the generator to its designated region. Our partitioner's guide is particularly useful where the discriminator does not provide reliable gradient, as it can steer the generator in the right direction.

However, for a reliable *guide*, the distance function is required to follow certain characteristics, which are challenging to achieve. For example, to avoid misleading the GANs's training the distance should have no local optima outside of the partition. In Section 2.4.2, we formulate sufficient theoretical conditions for a desirable metric, and attain them by enforcing constraints on the architecture of the space partitioner. As a by product, this also guarantees connected partitions in the data space which further mitigates the issue of mode connecting.

We perform comprehensive experiments on stackedMNIST [18, 19, 20], CIFAR-10 [21], and STL-10 [22]. We show that our method, *Partition-Guided Mixture of GAN* (PGMGAN), successfully recovers all the modes and achieves higher Inception Score (IS) [15] and Frechet Inception Distance (FID) [23] than a wide range of supervised and unsupervised methods.

Our contributions can be summarized as:

- Providing a theoretical lower bound on the total variational distance of real and estimated

density using a single generator.

- Introducing a novel differentiable space partitioner and demonstrating that simply training a mixture of generators on its partitions, improves mode collapse/connecting.
- Providing a practical way (with theoretical guarantees) to guide each generator to produce samples from its designated region, further improving mode collapse/connecting. Our experiments show significant improvement in terms of FID and IS confirming the efficacy of our model.
- Elaborating on the design of our loss and architecture by making connection to supervised GANs that employ a classifier. We explain how PGMGAN avoids their short comings.

2.2 Mode connecting problem

Suppose the data distribution is supported on a set of disconnected manifolds embedded within a higher-dimensional space. Since continuous functions preserve the space connectivity [24], one can never expect to have an exact approximation of this distribution by applying a continuous function ($G_{\boldsymbol{\theta}}$) to a random variable with a connected support. Furthermore, if we restrict $G_{\boldsymbol{\theta}}$ to the class of c-Lipschitz functions, the distance between the true density and approximated will always remain more than a certain positive value. In fact, the generator would either have to discard some of the data manifolds or connect the manifolds together. The former can be considered as a form of *mode collapse* and we refer to the latter as the *mode connecting* problem.

The following theorem formally describes the above statement and provides a lower bound for the total variation distance between the true and estimated densities.

Theorem 1. *Suppose p_{data} is a distribution supported on a set of disjoint manifolds $\mathcal{M}_1, \dots, \mathcal{M}_k$ in $\mathbb{R}^d$, and $[\pi_1, \dots, \pi_k]$ are the probabilities of being from each manifold. Let $G_{\boldsymbol{\theta}}$ be a c-Lipschitz function, and p_{model} be the distribution of $G_{\boldsymbol{\theta}}(\mathbf{z})$, where $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I}_n)$, then:*

$$d_{TV}(p_{\text{data}}, p_{\text{model}}) \geq \sum |\pi_i - p_i| \geq \delta$$

where d_{TV} is the total variation distance and:

$$\begin{aligned}
\pi_i^* &:= \min(\pi_i, 1-\pi_i) \\
p_i &:= p_{\text{model}}(\mathcal{M}_i) \\
\delta &:= \max_i \{\pi_i^* - \Phi(\Phi^{-1}(\pi_i^*) - d_i/c)\} \\
d_i &:= \inf\{||\boldsymbol{x}-\boldsymbol{y}|| \mid \boldsymbol{x} \in \mathcal{M}_i, \boldsymbol{y} \in \mathcal{M}_j, j \neq i\}
\end{aligned}$$

d_i is the distance of manifold $\mathcal{M}_i$ from the rest, and Φ is the CDF of the univariate standard normal distribution. Note δ is strictly larger than zero iff $\exists i:\ d_i, \pi_i^* \neq 0$.

Proof. We begin by re-stating the definition of Minkowski sum and proceed by proving the theorem for the case where the number of manifold is 2. To extend the theorem form $k = 2$ to the general case, one only needs to consider manifold $\mathcal{M}_i$ as $\mathcal{M}_1$, and $\bigcup_{j\in[1:k]\setminus i} \mathcal{M}_j$ as $\mathcal{M}_2$.

Definition 1 (Minkowski sum)**.** *The Minkowski sum of two sets $U, V \in \mathbb{R}^d$ defined as*

$$U + V := \{u + v | u \in U, v \in V\}$$

and when V is a d dimensional ball with radius r and centered at zero, we use the notation U_r to refer to their Minkowski sum.

If we let $U^{(1)} := G_\theta^{-1}(\mathcal{M}_1), U^{(2)} := G_\theta^{-1}(\mathcal{M}_2)$, then:

$$\forall r_1, r_2 \in \mathbb{R}_+, \text{ if } \ r_1 + r_2 < d_1/c \implies U_{r_1}^{(1)} \cap U_{r_2}^{(2)} = \emptyset$$

that is because if there exists an $\mathbf{x} \in U_{r_1}^{(1)} \cap U_{r_2}^{(2)}$, there would be $\mathbf{u}_1 \in U^{(1)}, \mathbf{u}_2 \in U^{(2)}$ such that:

$$||\mathbf{x} - \mathbf{u}_1|| \leq r_1, \quad ||\mathbf{x} - \mathbf{u}_2|| \leq r_2 \implies ||\mathbf{u}_1 - \mathbf{u}_2|| < r_1 + r_2 < d_1/c \tag{2.1}$$

However, due to lipsitz condition of G_θ:

$$||G_\theta(\mathbf{u}_1) - G_\theta(\mathbf{u}_2)|| < c||\mathbf{u}_1 - \mathbf{u}_2|| < c \cdot d_1/c = d_1$$

which contradicts with our assumption that the distance between $\mathcal{M}_1, \mathcal{M}_2$ is d_1. Therefore there is no point in the intersection of $U_{r_1}^{(1)}$ and $U_{r_2}^{(2)}$. The disjointness of this two sets provides us:

$$\gamma_n(U_{r_1}^{(1)}) + \gamma_n(U_{r_2}^{(2)}) \leq \gamma_n(\mathbb{R}^n) = 1$$

where $\gamma_n(.)$ of any set is the probability of a random draw of $\mathcal{N}(0, \boldsymbol{I}_n)$ being from that set. We proceed by using a remark from theorem 1.3 of [25] which restated below:

If U is a Borel set in $\mathbb{R}^n$, then:

$$p \leq \gamma_n(U) \implies \Phi(\Phi^{-1}(p) + r) \leq \gamma_n(U_r).$$

Based on above lemma if we let:

$$p_1 := \gamma_n(U^{(1)}), \quad p_2 := \gamma_n(U^{(2)})$$

then for $\forall r_1, r_2 \in \mathbb{R}_+$ such that $r_1 + r_2 < d_1/c$, we have:

$$\Phi(\Phi^{-1}(p_1) + r_1) + \Phi(\Phi^{-1}(p_2) + r_2) \leq 1 \tag{2.2}$$

We can now calculate the total variational distance of the marginal distributions of p_{data}, p_{model} on the set $\{G_\theta(U^{(1)}), G_\theta(U^{(2)}), G_\theta\left(\mathbb{R}^n \setminus (U^{(1)} \cup U^{(2)})\right)\}$ as:

$$d_{TV}(p_{\text{data}}^{(marginal)}, p_{\text{model}}^{(marginal)}) = |\pi_1 - p_1| + |\pi_2 - p_2| + |1 - (p_1 + p_2)| \tag{2.3}$$

and since total variational distance takes a smaller value on marginal distributions than the full distribution, we only need to show that for any i the expression in the equation 2.3 is larger than $g(\pi_i, d_i, c)$ to prove the theorem 1 for $k = 2$. Here $g(\pi_i, d_i, c) = \pi_i^* - \Phi(\Phi^{-1}(\pi_i^*) - d_i/c)$.

Assume $p_1 \leq \pi_1$, and define $\Delta_1 := \pi_1 - p_1 \geq 0$, based on equation 2.2 if $r_1 = d_1/c,\ \ r_2 = 0$, we have:

$$\Phi(\Phi^{-1}(\pi_1 - \Delta_1) + d_1/c) + p_2 \leq 1$$

which based on equation 2.2 implies:

$$r(\Delta_1) := \Phi(\Phi^{-1}(\pi_1 - \Delta_1) + d_1/c) - (\pi_1 - \Delta_1) \leq D_{TV}$$

which D_{TV} refers to total variational distance between the marginal distributions of the data and model. We also know from the equation 2.2, that $\Delta_1 \leq D_{TV}$, therefore:

$$\max(r(\Delta_1), \Delta_1) \leq D_{TV}$$

for a $\Delta_1 \in [0, \pi_1]$. Therefore

$$\min_{\delta_1 \in [0, \pi_1]} \{\max(r(\delta_1), \delta_1)\} \leq D_{TV}$$

To find the δ_1 which minimize the above equation, we need to check endpoints of the interval $[0, \pi_1]$, points where the curve of two functions $r(\delta_1), \delta_1$ intersects with each other, and points that are the local minimaum of each of them. It can be shown the function $r(\delta_1)$ does not have any local minima when $0 < \pi_1 < 1$ because:

$$r(\delta_1) = P(\mathbf{z} \in [\Phi^{-1}(\pi_1 - \delta_1), \Phi^{-1}(\pi_1 - \delta_1) + d_1/c]) \tag{2.4}$$

where z is univarite standard normal random variable. Therefore $r(\delta_1)$ is the probablity of a univarite

normal being in a fixed length interval d_1/c, and δ_1 only changes the starting point of the interval. By using this fact, it can be easily shown this function does not have any local optima in the open interval $(0, \pi_1)$. Also the identity function δ_1 also has no local optima inside the interval. The endpoints values are:

$$\max(r(0), 0) = \Phi(\Phi^{-1}(\pi_1) + d_1/c) - \pi_1$$

$$\max(r(\pi_1), \pi_1) = \max(\Phi(\Phi^{-1}(0) + d_1/c), \pi_1) = \pi_1$$

The function curves of r and identity also intersects only when:

$$\Phi(\Phi^{-1}(\pi_1 - \delta_1^*) + d_1/c) = \pi_1$$

which only happens when:

$$\pi_1 - \Phi(\Phi^{-1}(\pi_1) - d_1/c) = \delta_1^*$$

where for this point, $\max(r(\delta_1^*), \delta_1^*) = \delta_1^*$. Therefore based on the above calculations:

$$\min \left\{ \underbrace{\Phi(\Phi^{-1}(\pi_1) + d_1/c) - \pi_1}_{\text{I}}, \pi_1, \right.$$
$$\left. \underbrace{\pi_1 - \Phi(\Phi^{-1}(\pi_1) - d_1/c)}_{\text{II}} \right\} \leq D_{TV}$$

Note, π_1 is always smaller than term II, and term I (II) is equal to probability of a univariate standard normal random variable being inside the interval $[\Phi^{-1}(\pi_1), \Phi^{-1}(\pi_1) + d_1/c]$ ($[\Phi^{-1}(\pi_1) - d_1/c, \Phi^{-1}(\pi_1)]$). This observation implies that term II is smaller than term I, if and only if $\pi_1 \leq \pi_2$. Based on this fact and symmetry of Φ with respect to zero, it can be easily shown that:

$$g(\pi_1, d_1, c) \leq D_{TV} \tag{2.5}$$

which proves the theorem. However, we made an assumption that $p_1 \leq \pi_1$, this does not harm the argument because otherwise we would have $p_2 \leq \pi_2$, and we can restate all the above arguments for π_2 instead of π_1. And, since $\pi_2 = 1 - \pi_1$ and $d_1 = d_2$, therefore we can have $g(\pi_1, d_1, c) =$

$g(\pi_2, d_2, c)$, which proves equation 2.5. □

According to Theorem 1, the distance between the estimated density and the data distribution can not converge to zero when $G_{\boldsymbol{\theta}}$ is a Lipschitz function. It is worth noting that this assumption holds in practice for most neural architectures as they are a composition of simple Lipschitz functions. Furthermore, most of the state of the art GAN architectures (e.g., BigGAN [26] or SAGAN [27]) use spectral normalization in their generator to stabilize their training which promotes Lipschitzness.

2.3 Related work

Apart from their application in computer vision [28, 29, 30, 31, 32, 33, 34, 35], GANs have also been been employed in natural language processing [36, 37, 38], medicine [39, 40] and several other fields [41, 42, 43]. Many of recent research have accordingly focused on providing ways to avoid the problems discussed in Section 2.2 [18, 44].

Mode collapse For instance, Metz et al. [44] unrolls the optimization of the discriminator to obtain a better estimate of the optimal discriminator at each step, which remedies mode collapse. However due to high computational complexity, it is not scalable to large datasets. VEEGAN [20] adds a reconstruction term to bi-directional GANs [45, 46] objective which does not depend on the discriminator. This term can provide training signal to the generator even when the discriminator does not. PacGAN [18] changes the discriminator to make decisions based on a pack of samples. This change mitigates mode collapse by making it easier for the discriminator to detect lack of diversity and naturally penalizing the generator when mode collapse happens. Lucic et al. [47], motivated by the better performance of supervised-GANs, propose to use an small set of labels and a semi-supervised method to infer the labels for the entire data. They further improve the performance by utilizing a auxiliary rotation loss similar to that of RotNet [48].

Mode connecting Based on Theorem 1, to avoid mode connecting one has to either use a latent variable $\mathbf{z}$ with a disconnected support, or allow $G_{\boldsymbol{\theta}}$ to be a discontinuous function [49, 16, 50, 19, 51].

To obtain a disconnected latent space, DeLiGAN [52] samples $\mathbf{z}$ from a mixture of Gaussian,

while Odena et al. [53] add a discreet dimension to the latent variable. Other methods dissect the latent space post training using some variant of rejection sampling, for example Azadi et al. [54] perform rejection sampling based on the discriminator's score and Tanielian et al. [17] reject the samples where the generator's Jacobian is higher than a certain threshold.

The discontinuous generator method is mostly achieved by learning multiple generators, with the main motivation being to remedy mode-collapse which also reduces mode connecting. Both MGAN [49] and DMWGAN [16] employ K different generators while penalizing them from overlapping with each other. However, these works do not explicitly address the issue when some of the data modes are not being captured. Also, as show in Liu et al. [19], MGAN is quite sensitive to the choice of K. By contrast, Self-Conditioned GAN [19] clusters the space using the discriminator's final layer and uses the labels as self-supervised conditions. However, in practice their clustering does not seem to be reliable (e.g., in terms of NMI for labeled datasets) and the features highly depend on the choice of the discriminator‘s architecture. In addition, there is no guarantee that the generators will be guided to generate from their assigned clusters. GAN-Tree [50] uses hierarchical clustering to address continuous multi-modal data, with the number of parameters increasing linearly with the number of clusters. Thus it is limited to very few cluster numbers (e.g. 5) and can only capture a few modes.

Another recently expanding direction which explores the benefit of using image augmentation techniques for generative modeling. Some works simply augment the data using various perturbations (e.g. random crop, horizontal flipping) [55]. Others [56, 47, 57] incorporated regularization on top of the augmentations, for example CRGAN [58] enforces consistency for different image perturbations. ADA [59] processes each image using non-leaking augmentations and adaptively tunes the augmentation strength while training. These works are orthogonal to ours and can be combined with our method.

2.4 Method

This section first describes how GANs are trained on a partitioned space using a mixture of generators/discriminators and the unified objective function required for this goal. We then explain

our differentiable space partitioner and how we guide the generators towards the right region. We conclude the section by making connections to supervised GANs, which use an auxiliary classifier [60, 53].

Multi-generator/discriminator objective: Given a partitioning of the space, we train a generator (G_i) and a discriminator (D_i) for each region. To avoid over-parameterization and allow information sharing across different regions, we employ parameter sharing across different G_i (D_i)'s by tying their parameters except the input (last) layer. The mixture of these generators serves as our main generator G. We use the following objective function to train our GANs:

$$\sum_{i}^{k} \pi_i \left[\min_{G_i} \max_{D_i} V(D_i, G_i, A_i) \right] \tag{2.6}$$

where $A_1, A_2, ..., A_k$ be a partitioning of the space, $\pi_i := p_{data}(\mathbf{x} \in A_i)$ and:

$$\begin{aligned} V(D, G, A) = &\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}|\boldsymbol{x} \in A)}[\log D(\boldsymbol{x})] + \\ &\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}|G(\boldsymbol{z}) \in A)}[\log(1 - D(G(\boldsymbol{z})))] \end{aligned} \tag{2.7}$$

We motivate this objective by making connection to the Jensen–Shannon distance (JSD) between the distribution of our mixture of generators and the data distribution in the following Theorem.

Theorem 2. *Let $P = \sum_i^k \pi_i p_i$, $Q = \sum_i^k \pi_i q_i$, and $A_1, A_2, ..., A_K$ be a partitioning of the space, such that the support of each distribution p_i and q_i is A_i. Then:*

$$\text{JSD}(P \parallel Q) = \sum_i \pi_i \text{JSD}(p_i \parallel q_i) \tag{2.8}$$

Proof. Based on the definition of the JSD, we have:

$$\text{JSD}(P \parallel Q) = \frac{1}{2}\text{KL}(P \parallel \frac{P+Q}{2}) + \frac{1}{2}\text{KL}(Q \parallel \frac{P+Q}{2})$$

We also have:

$$\begin{aligned}
&\mathrm{KL}(P \parallel \frac{P+Q}{2}) = \\
&\int_{\mathbb{R}^d} P(x) \log \frac{P(x)}{P(x)+Q(x)}\, dx + \log 2 = \\
&\sum_i \int_{A_i} P(x) \log \frac{P(x)}{P(x)+Q(x)}\, dx + \log 2 = \\
&\sum_i \int_{A_i} \pi_i p_i(x) \log \frac{\pi_i p_i(x)}{\pi_i p_i(x) + \pi_i q_i(x)}\, dx + \log 2 = \\
&\sum_i \pi_i (\int_{A_i} p_i(x) \log \frac{p_i(x)}{p_i(x)+q_i(x)}\, dx + \log 2) = \\
&\sum_i \pi_i \mathrm{KL}(p_i \parallel \frac{p_i+q_i}{2}).
\end{aligned}$$

Therefore:

$$\mathrm{KL}(P \parallel \frac{P+Q}{2}) = \sum_i \pi_i \mathrm{KL}(p_i \parallel \frac{p_i+q_i}{2}),$$

and similarly:

$$\mathrm{KL}(Q \parallel \frac{P+Q}{2}) = \sum_i \pi_i \mathrm{KL}(q_i \parallel \frac{p_i+q_i}{2})$$

Adding these two terms completes the proof. □

2.4.1 Partition GAN

Space Partitioner: Based on theorem 1, an ideal space partitioner should place disjoint data manifolds in different partitions to avoid mode connecting (and consequently mode collapse). It is also reasonable to assume that semantically similar data points lay on the same manifold. Hence, we train our space partitioner using semantic embeddings.

We achieve this goal in two steps: 1) Learning for each data point an unsupervised representation which is invariant to transformations that do not change the semantic meaning. 2) Training a partitioner based on these features, where data points with similar features are placed in the same

partition.

Learning representations: We follow the self-supervised literature [61, 62, 63] to construct image representations. These methods generally train the networks via maximizing the agreement between augmented views (e.g., random crop, color distortion, rotation, etc.) of the same scene, while minimizing the agreement of views from different scenes. To that end, they optimize the following contrastive loss:

$$\sum_{(i,j)\in P} \log \frac{\exp(\text{sim}(\mathbf{h}_i, \mathbf{h}_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{k\neq i} \exp(\text{sim}(\mathbf{h}_i, \mathbf{h}_k)/\tau)} \tag{2.9}$$

where $\mathbf{h}$ is the embedding for image $\mathbf{x}$, (i, j) is a positive pair (i.e., two views of the same image) and (i, k) refers to negative pairs related to two different images. We refer to this network as pretext, implying the task being solved is not of real interest, but is solved only for the true purpose of learning a good data representation.

Learning partitions: To perform the partitioning step, one can directly apply K-means on theses semantic representations. However, this may result in degenerated clusters where one partition contains most of the data [64, 65]. Inspired by Van Gansbek et al. [65], to mitigate this challenge, we first make a k-nearest neighbor graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ based on the representations $\mathbf{h}$ of the data points. We then train an unsupervised model that motivates connected points to reside in the same cluster and disconnected points to reside in distinct clusters. More specifically, we train a space partitioner $S : \mathbb{R}^d \to [0, 1]^k$ to maximize:

$$\begin{aligned} &\sum_{(i,j)\in\mathcal{E}} \log \left(S(\mathbf{x}_i)^T \cdot S(\mathbf{x}_j)\right) - \\ &\qquad \alpha \sum_{i\in\mathcal{V}} H_C(S(\mathbf{x}_i))) + \beta H_C(\sum_{i\in\mathcal{V}} \frac{S(\mathbf{x}_i)}{N}) \end{aligned} \tag{2.10}$$

Where $H_C(.)$ is the entropy function of the categorical distribution based on its probability vector. The first term in Equation 2.10 motivates the neighbors in $\mathcal{G}$ to have similar class probability vectors, with the $\log$ function used to significantly penalize the classifier if it assigns dissimilar

probability vector to two neighboring points. The last term is designed to avoid placing all the data points in the same category by motivating the average cluster probability vector be similar to uniform. The middle term is intended to promote the probability vector for each data point to significantly favor one class over the other. This way, we can be more confident about the cluster id of each data point. Furthermore, if the average probability of classes has homogeneous mean (because of the last term), we can expect the number of data points in each class to not degenerate.

To train S efficiently, both in term of accuracy and computational complexity, we initialize S using the already trained network of unsupervised features. More specifically, for:

$$\mathbf{h} = W_2^{\textit{pretext}} \sigma(W_1^{\textit{pretext}} \phi_0(\mathbf{x}))$$

we initialize S as follows:

$$S^{\textit{init}}(\mathbf{x}) = \textit{softmax}(W_0^{\textit{partitioner}} \phi_0(\mathbf{x}))$$

where σ is an activation function, and $W_0^{\textit{partitioner}}$ is a randomly initialized matrix; we ignore the bias term here for brevity. We drop the sub index 0, from $W_0^{\textit{partitioner}}$ and ϕ_0 to refer to their post-training versions. Given a fully trained S, each point $\mathbf{x}$ is assigned to partition A_i, based on the argmax of the probability vector of $S(\mathbf{x})$.

2.4.2 Partition-Guided GAN

In this section we describe the design of *guide* and its properties. As stated previously, we want to guide each generator G_i to its designated region A_i by penalizing it the farther its current generated samples are from A_i.

A simple, yet effective proxy of measuring this distance can be attained using the already trained space partitioner. Let f_is denote the partitioner's last layer logits, expressed as

$$[f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T := W^{\textit{partitioner}} \phi(\mathbf{x}).$$

and define the desired distance as:

$$R_i(\mathbf{x}) := \sum_c (f_c(\mathbf{x}) - f_i(\mathbf{x}))_+ \tag{2.11}$$

Property 1. It is easy to show that for any generated sample $\mathbf{x}$, $R_i(\mathbf{x})$ achieves a larger value, the less likely S believes $\mathbf{x}$ to be from partition A_i. This is clear from how we defined R_i, the more probability mass $S(\mathbf{x})$ assigns any class $c \neq i$, the larger the value of $R_i(\mathbf{x})$.

Property 2. It is also straight forward see that $R_i(\mathbf{x})$ is always non-negative and obtains its minimum (zero) only on the $A_i{}^{\text{th}}$ partition:

$$\begin{aligned} \mathbf{x} \in A_i &\iff f_i(\mathbf{x}) \geq f_c(\mathbf{x}); \quad \forall c \in [1:k] \\ &\iff R_i(\mathbf{x}) = \sum_c (f_c(\mathbf{x}) - f_i(\mathbf{x}))_+ = 0 \end{aligned} \tag{2.12}$$

Therefore, we guide each generator G_i to produce samples from its region by adding a penalization term to its canonical objective function:

$$\begin{aligned} \min_{G_i} \sum_{j=1}^{n} \log(1 - D_i(G_i(\boldsymbol{z}^{(j)}))) \\ + \lambda \sum_j R_i(G_i(\boldsymbol{z}^{(j)}))/n. \end{aligned} \tag{2.13}$$

Intuitionally, G_i needs to move its samples towards partition A_i in order to minimize the newly added term. Fortunately, given the differentiability of $R_i(.)$ with respect to its inputs and *property* 1, R_i can provide the direction for G_i to achieve that goal.

It is also worth noting that R_i should not interfere with the generator/discriminator as long as G_i's samples are within A_i. Otherwise, this may lead to the second term favoring parts of A_i over others and conflicting with the discriminator. *Property* 2. assures that learning the distribution of p_{data} over A_i remains the responsibility of D_i. We also use this metric to ensure each trained generator G_i only draws samples from within its region by only accepting samples with $R_i(\mathbf{x})$ is

equal to zero. A critical point left to consider is the possibility of G_i getting fooled to generate samples from outside A_i, by falling in local optima of $R_i(\mathbf{x})$. In the remaining part of this section, we will explain how the architecture design of the space partitioner S avoids this issue. In addition, it will also guarantee the norm of the gradient provided by R_i to always be above a certain threshold.

Avoiding local optima: We can easily obtain a guide R_i with no local optima, if achieving a good performance for the partitioner was not important. For instance, a simple single-linear-layer neural network as S would do the trick. The main challenge comes from the fact that we need to perform well on partitioning which usually requires deep neural networks while Avoiding local optima. We fulfill this goal by first finding a sufficient condition to have no local optima and then trying to enforce that condition by modifying the ResNet [66] architecture.

The following theorem states the sufficient condition:

Theorem 3. *Let $\phi(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ be a C^1 (differentiable with continuous derivative) function, $W^{partitioner} \in \mathbb{R}^{k \times d}$, and R_i as defined in Eq 2.11. If there exists $c_0 > 0$, such that:*

$$\forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, \quad c_0||\mathbf{x} - \mathbf{y}|| \leq ||\phi(\mathbf{x}) - \phi(\mathbf{y})||,$$

then for every $i \in [1 : k]$, every local optima of R_i is a global optima, and there exists a positive constant $b_0 > 0$ such that:

$$\forall \mathbf{x} \in \mathbb{R}^d \setminus A_i, \quad b_0 \leq ||\nabla R_i(\mathbf{x})||$$

where $A_i = \{\mathbf{x} | \mathbf{x} \in \mathbb{R}^d, R_i(\mathbf{x}) = 0\}$. Furthermore A_i is a connected set for all i's.

Proof. We start by proving that the Jacobian matrix of function ϕ is invertible for any $\mathbf{x} \in \mathbb{R}^d$. Since $\phi \in C^1$, based on Taylor's expansion theorem for multi-variable vector-valued function ϕ, we can write:

$$\phi(\mathbf{y}) - \phi(\mathbf{x}) = \boldsymbol{J}_\phi(\mathbf{x})\,(\mathbf{y} - \mathbf{x}) + o\,(||\mathbf{y} - \mathbf{x}||)$$

Were $o(\cdot)$ is the Little-o notation. By taking norm from both sides and using triangle inequality, we

have:

$$||\phi(\mathbf{y}) - \phi(\mathbf{x})|| \leq ||\boldsymbol{J}_\phi(\mathbf{x})\,(\mathbf{y}-\mathbf{x})\,|| + ||o\,(||\mathbf{y}-\mathbf{x}||)\,||$$

Also because:

$$c_0||\mathbf{y}-\mathbf{x}|| \leq ||\phi(\mathbf{y}) - \phi(\mathbf{x})||$$

$$\implies c_0||\mathbf{y}-\mathbf{x}|| \leq ||\boldsymbol{J}_\phi(\mathbf{x})\,(\mathbf{y}-\mathbf{x})\,|| + ||o\,(||\mathbf{y}-\mathbf{x}||)\,|| \tag{2.14}$$

thus for any fixed $\mathbf{x}$: $\exists\, \epsilon > 0$ such that $\forall\, \mathbf{y} \in \mathbb{R}^d$ where $||\mathbf{y}-\mathbf{x}|| \leq \epsilon$ then:

$$||o\,(||\mathbf{y}-\mathbf{x}||)\,|| \leq \frac{c_0}{2}||\mathbf{y}-\mathbf{x}||$$

which combined with the inequality 2.14, results in:

$$\frac{c_0}{2}||\mathbf{y}-\mathbf{x}|| \leq ||\boldsymbol{J}_\phi(\mathbf{x})\,(\mathbf{y}-\mathbf{x})\,||$$

For $\mathbf{y} \neq \mathbf{x}$, let $\mathbf{u} := (\mathbf{y}-\mathbf{x})/||\mathbf{y}-\mathbf{x}||$, then by dividing both sides of the above inequality to $||\mathbf{y}-\mathbf{x}||$ we have:

$$\forall \mathbf{u} \in \mathbb{R}^d, ||\mathbf{u}|| = 1 \implies \frac{c_0}{2} \leq ||\boldsymbol{J}_\phi(\mathbf{x})\mathbf{u}||$$

which shows the Jacobian matrix of ϕ is invertiable for any $\mathbf{x}$ and all of its singular values are larger than $c_0/2$. If there is no $\mathbf{x} \in \mathbb{R}^d \setminus A_i$ the proof is complete. Otherwise, consider any $\mathbf{x} \in \mathbb{R}^d \setminus A_i$, for this $\mathbf{x}$ we have:

$$0 < R_i(\mathbf{x}) = \sum_c (f_c(\mathbf{x}) - f_i(\mathbf{x}))_+ = \sum_c ((\boldsymbol{w}_c - \boldsymbol{w}_i)\phi(\mathbf{x}))_+$$

where $\boldsymbol{w}_j$ is the j'th row of the matrix $\boldsymbol{W}^{partitioner}$. Let:

$$I(\mathbf{x}) := \{c | f_c(\mathbf{x}) > f_i(\mathbf{x}), \quad c \in [1:k]\}$$

which is an non-empty set, because $0 < R_i(\mathbf{x})$ and we have

$$0 < R_i(\mathbf{x}) = \left[\sum_{c \in I(\mathbf{x})} (\boldsymbol{w}_c - \boldsymbol{w}_i)\right] \phi(\mathbf{x})$$
$$\implies \boldsymbol{v} := \sum_{c \in I(\mathbf{x})} (\boldsymbol{w}_c - \boldsymbol{w}_i) \neq 0$$

Taking the gradient of the new formulation of R_i, we have:

$$\nabla R_i(\mathbf{x}) = \left[\sum_{c \in I(\mathbf{x})} (\boldsymbol{w}_c - \boldsymbol{w}_i)\right] \nabla(\phi(\mathbf{x})) = \boldsymbol{v}\boldsymbol{J}_\phi(\mathbf{x})$$

but since we showed earlier that all of the singular values of the Jacobian matrix is larger than $c_0/2$, the Jacobian matrix is $d \times d$, and $\boldsymbol{v}$ is not equal to zero, it can be easily shown:

$$||\nabla R_i(\mathbf{x})|| > ||\boldsymbol{v}|| \frac{c_0}{2} := b_0$$

Now, we also need to show A_i is connected for any i to complete the proof. To that end, we first show ϕ is a surjective function, which means its image is $\mathbb{R}^d$. To show the ϕ is surjective, we prove its image is both an open and closed set, then since the only sets which are both open and closed (in $\mathbb{R}^d$) are $\mathbb{R}^d, \emptyset$, we can conclude the surejective property. The image of ϕ is an open set due to Inverse Function Theorem [67] for ϕ. We are allowed to use Inverse Function Theorem, since ϕ satisfies both C^1 condition and non zero determinant for all the points in the domain. We will also show that the image of ϕ is a closed set by showing it contains all of its limit points. Let $\boldsymbol{y}$ be a limit point in the image of ϕ, that is there exists $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots\}$ such that $\phi(\boldsymbol{x}_r) \to \boldsymbol{y}$. Since $\mathbb{R}^d$ is complete and we have $c_0||\boldsymbol{x}_r - \boldsymbol{x}_s|| \leq ||\phi(\boldsymbol{x}_r) - \phi(\boldsymbol{x}_s)||$, then $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots\}$ is a Cauchy sequence. Finally since ϕ is a continues function $\phi(\boldsymbol{x}^*) = \boldsymbol{y}$, completing the proof.

The function ϕ is also an invertible function because if $\phi(\boldsymbol{x}) = \phi(\boldsymbol{y})$ then

$$c_0||\mathbf{x} - \boldsymbol{y}|| \leq ||\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})|| = 0$$

which implies $\boldsymbol{x} = \boldsymbol{y}$. Therefore ϕ is in fact a continuous bijecitve function, which means it has a continuous inverse defined on all the space $\mathbb{R}^d$. Furthermore, it can be easily shown R_i for a datapoint is zero iff its transformation by ϕ lies in a polytope (where each of its facets is a hyperplance perpendicular to a $\boldsymbol{w}_c - \boldsymbol{w}_i$). Since convex polytope is a connected set, and by applying ϕ^{-1} (it is well defined everywhere because of bijective property of ϕ) to it, we would have a connected set. That is because a continuous function does not change the connectivity and ϕ^{-1} is continuous. □

Next we describe how to satisfy this constrain in practice.

Motivated by the work of Behrmann et al. [68] who design an invertible network without significantly sacrificing their classification accuracy, we implement ϕ by stacking several residual blocks, $\phi(\mathbf{x}) = B_T \circ B_{T-1} \circ \cdots \circ B_1(\mathbf{x})$, where:

$$B_{t+1}(\mathbf{x}^{(t)}) = \mathbf{x}^{(t+1)} := \mathbf{x}^{(t)} + \psi_t(\mathbf{x}^{(t)})$$

and $\mathbf{x}^{(t)}$ refers to the out of the t^{th} residual block. Figure 2.2, gives an overview of the proposed architecture.

We model each ψ_t as a series of m convolutional layers, each having spectral norm $L < 1$ intertwined by 1-Lipschitz activation functions (e.g., ELU, ReLU, LeakyReLU). Thus it can be easily shown for all $\mathbf{x}^{(t)}, \mathbf{y}^{(t)} \in \mathbb{R}^d$:

$$(1 - L^m)||\mathbf{x}^{(t)} - \mathbf{y}^{(t)}|| \leq ||B_t(\mathbf{x}^{(t)}) - B_t(\mathbf{y}^{(t)})||.$$

This immediately results in the condition required in Theorem 3 by letting $c_0 := (1 - L^m)$.

2.4.3 Connection to supervised GANs

In this section, we make a connection between our unsupervised GAN and some important work in the supervised regime. This will help provide better insight into why the mentioned properties

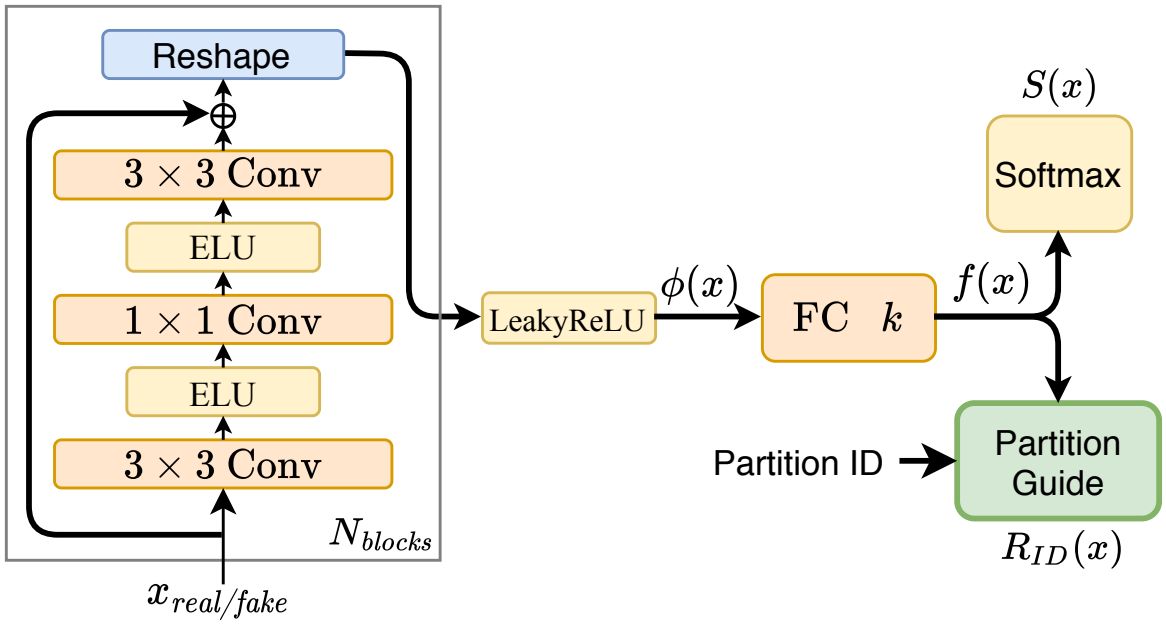


Figure 2.2: Diagram of proposed partitioner and guide. We employ spectral normalization for each convolutional layer to make each layer (as a function) have Lipschitz constant of less than one. The details of our architecture is provided in the Appendix.

of guide are important. Auxiliary Classifier GAN [53] has been one of the well-known supervised GAN methods which uses the following objective function:

$$\min_{G,C} \max_{D} \mathcal{L}_{\text{AC}}(G, D, C) =$$

$$\underbrace{\underset{X\sim P_X}{E}[\log D(X)] + \underset{Z\sim P_Z, Y\sim P_Y}{E}[\log(1 - D(G(Z, Y)))]}_{\text{(a)}}$$

$$- \lambda_c \underbrace{\underset{(X,Y)\sim P_{XY}}{E}[\log C(X, Y)]}_{\text{(b)}} - \lambda_c \underbrace{\underset{Z\sim P_Z, Y\sim P_Y}{E}[\log(C(G(Z, Y), Y))]}_{\text{(c)}}$$

It simultaneously learns an auxiliary classifier C as well as D/G. Other works have also tried fine-tuning the generator using a pre-trained classifier [60]. The term a is related to the typical supervised conditional GAN, term b motivates the classifier to better classify the real data. The term c encourages G to generate images for each class such that the classifier considers them to belong to that class with a high probability.

The authors motivate adding this term as it can provide further gradient to the generator $G(\cdot|Y)$ to generate samples from the correct region $P_X(\cdot|Y)$. However, recent works [69, 70] show this tends to motivate G to down-sample data points near the decision boundary of the classifier. It has also been shown to reduce sample diversity and does not behave well when the classes share overlapping regions [69].

Our space partitioner acts similar to the classifier in these GANs, with the term c sharing some similarity with our proposed *guide*. In contrast, our novel design of $R_i(\cdot)$ enjoys the benefits of the classifier based methods (providing gradient for the generator) but alleviates its problems. Mainly because 1) It provides gradient to the generator to generate samples from its region. At the same time, due to having no local optima (only global optima), it does not risk the generator getting stuck where it is not supposed to. 2) Within regions, our guide does not mislead the generator to favor some samples over others. 3) Since the space partitioner uses the partition labels as "class" id, it does not suffer from the overlapping classes problem, and naturally, it does not require supervised labels. We believe our construction of the modified loss can also be applied to the supervised regime to avoid putting the data samples far from the boundary. In addition, combining our method with the supervised one, each label itself can be partitioned into several segments. We leave the investigation of this modification to future research.

2.5 Experiments

This section provides an empirical analysis of our method on various datasets[1]. We adopt the architecture of SN-GAN [71] for our generators and discriminators. We use a Lipschitz constant of 0.9 for our space partitioner that consists of 20 residual blocks, resulting in 60 convolutional layers. We use Adam optimizer [72] to train the proposed generators, discriminators, and space partitioner, and use SGD for training the pretext model. Please refer to the Appendix for complete details of hyper-parameters and architectures used for each component of our model.

Datasets and evaluation metrics We conduct extensive experiments on CIFAR-10 [21] and

[1] The code to reproduce experiments is available at `https://github.com/alisadeghian/PGMGAN`

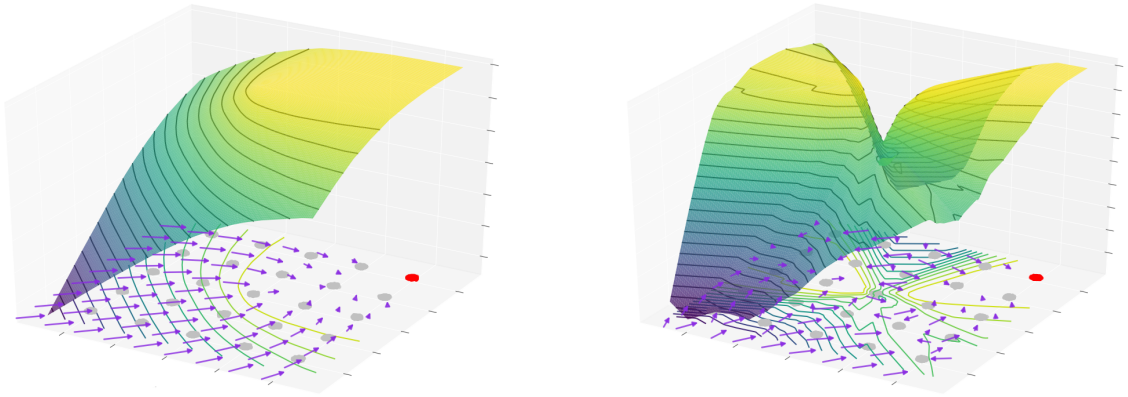


Figure 2.3: Left/right: the graph of $-R_i(\mathbf{x})$ with/without assumption on the architecture, where the data points of the i^{th} partition are shown in red.

STL-10 [22] (48×48), two real image datasets widely used as benchmarks for image generation. To see how our method fares against large dataset, we also applied our method on ILSVRC2012 dataset (ImageNet) [73] which we compressed to 128×128 pixel.To evaluate and compare our results, we use Inception Score (IS) [15] and Frechet Inception Distance (FID) [23]. It has been shown that IS may have many shortcomings, especially on non-ImageNet datasets. FID can detect mode collapse to some extent for larger datasets [74, 75, 76]. However, since FID is not still a perfect metric, we also evaluate our models using *reverse-KL* which reflects both mode dropping and spurious modes[76]. All FIDs and Inception Scores (IS) are reported using 50k samples. No truncation trick is used to sample from the generator.

We also conduct experiments on three synthetic datasets: Stacked-MNIST [18, 19, 20] that has up to 1000 modes, produced by stacking three randomly sampled MNIST [77] digits into the three channels of an RGB image, and the 2D-grid dataset described in Section 2.5.1 as well as 2D-ring dataset. The empirical result of the two later datasets are presented in the Appendix.

2.5.1 Toy dataset

This section aims to illustrate the importance of having a proper guide with no local optima. We also provide intuition about how our method helps GAN training. To that end, we use the canonical 2D-grid dataset, a mixture of 25 bivariate Gaussian with identical variance, and means covering the

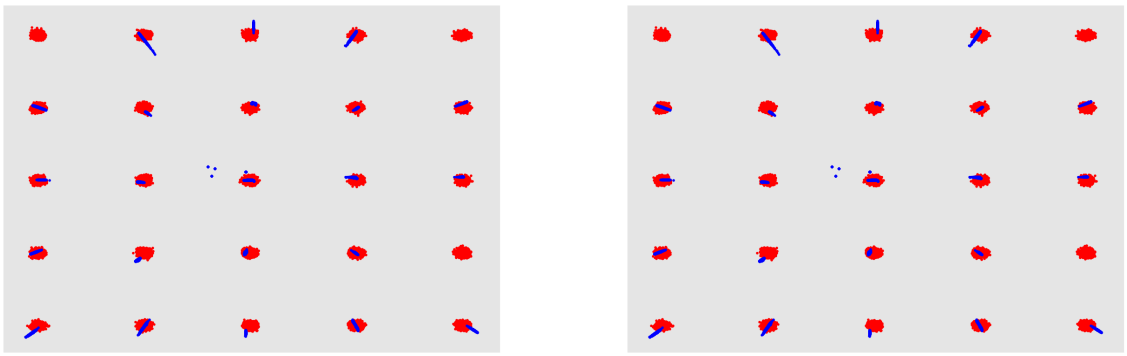


Figure 2.4: Left/right: visual comparison of generated samples on the 2D-grid dataset using PGMGAN, with/without architecture restriction for the space partitioner. The red/blue points illustrate the real/generated data samples. In the right plot, some modes are missed and their corresponding generators focus on the wrong area.

vertices of a square lattice.

For this toy example the data points are low dimensional, thus we skip the feature learning step and directly train the space partitioner S. We train our space partitioner using two different architectures: one that sets its neural network architecture to a multi-layer fully connected network with ReLU activations; while, the other follows the properties of architecture construction in Section 2.4.2. Once the two networks are trained, both successfully learn to put each Gaussian component in a different cluster, i.e., both get perfect clustering accuracy. Nonetheless, the *guide* functions obtained from each architecture behave significantly different.

Figure 2.3 provides the graph of $-R_i(\boldsymbol{x})$ for the two space partitioners, where i is the partition ID for the red Gaussian data samples in the corner. The right plot shows that $-R_i(\mathbf{x})$ can have undesired local optima when the conditions of Section 2.4.2 are not enforced. Therefore, a universal reliable gradient is not provided to move the data samples toward the partition of interest. On the other hand, when the guide's architecture follows these conditions (left plot), taking the direction of $\nabla - R_i(\boldsymbol{x})$ guarantees reaching to partition i.

Figure 2.4 shows the effect of both these guides in the training of our mixture of generators using Equation 2.13. As shown, when R_i has local optima, the generator of that region may get stuck in those local optima and miss the desired mode. As shown in Liu et al. [19], and in the Appendix,

GANs trained with no guide also fail to generate all the modes in this dataset. Furthermore, in contrast to standard GANs, we don't generate samples from the space between different modes due to our partitioning and mixture of generators, mitigating the mode connecting problem. Our quantitative results on the (2D-ring, 2D-grid) toy dataset [18] are: Recovered Modes: (8 , 25), high quality samples: (99.8 , 99.8), reverse KL: (0.0006, 0.0034).

2.5.2 Stacked-MNIST, CIFAR-10 and STL-10

In this section, we conduct extensive experiments to evaluate the proposed Partition-Guided Mixture of Generators (PGMGAN) model. We also quantify the performance gains for the different parts of our method through an ablation study. We randomly generated the partition labels in one baseline to isolate the effect of proper partitioning from the architecture choice of G_i/D_i's. We also ablate the benefits of the guide function by making a baseline where $\lambda = 0$. For all experiments, we use $k = 200$ unless specified otherwise.

Tables 2.4.3 and **??** presents our results on Stacked MNIST, CIFAR-10 and STL-10 respectively. From these tables, it is evident how training multiple generators using the space partitioner allows us to significantly outperform the other benchmark algorithms in terms of all metrics. Comparing *Random Partition ID* to *Partition+GAN* clearly shows the importance of having an effective partitioning in terms of performance and mode covering. Furthermore, the substantial gap between *PGMGAN* and *Partition+GAN* empirically demonstrates the value of utilizing the guide term.

We first perform overall comparisons against other recent GANs. As shown in Table **??**, PGMGAN achieves state-of-the-art FID and IS on the STL-10 dataset. Furthermore, PGMGAN outperforms several other baseline models, even over supervised class-conditional GANs, on both CIFAR-10 and Stacked-MNIST, as shown in Table 2.4.3. The significant improvements of FID and IS reflect the large gains in diversity and image quality on these datasets.

Following Liu et al. [19], we calculate the reverse KL metric using pre-trained classifiers to classify and count the occurrences of each mode for both Stacked-MNIST and CIFAR-10. These experiments and comparisons demonstrate that our proposed guide model effectively improves the

performance of GANs in terms of mode collapse.

2.5.3 Image generation on unsupervised ImageNet

To show that our method remains effective on a large high dimensional dataset, we also trained our model on unsupervised ILRSVRC2012 (ImageNet) dataset which contains roughly 1.2 million images with 1000 distinct categories and we down-sample the images to 128 resolution for the experiment. We adopt the architecture of BigGAN[26] for our generators and discriminators, and use $k = 1000$. Please see the Appendix for the details of experimental settings.

Our results are presented in Table **??**. To the best of our knowledge, we achieve a new state of the art (SOTA) in unsupervised generation on ImagNet.

2.5.4 Parameter sensitivity

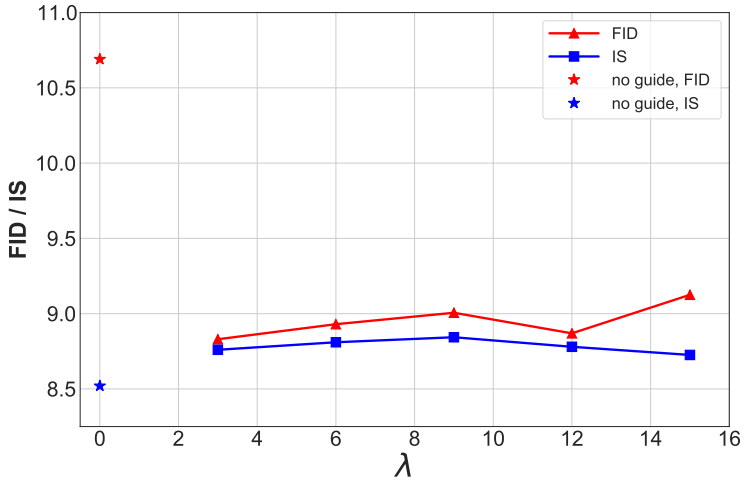
Additionally, we study the sensitivity of our overall PGMGAN method to the choice of guide's weight λ (Eq. 2.13), and number of clusters k. Figure 2.5 shows the results with varying λ, demonstrating that our method is relatively robust to the choice of this hyper-parameter. Next, we change k for a fixed $\lambda = 6.0$ and report the results in Table **??**. we observe that our method performs well for a wide range of k.

2.5.5 Implementation details

We use two RTX 2080 Ti GPUs for experiments on STL-10, eight GPUs for ImageNet and a single GPU for all other experiments.

Space partitioner. For all experiments we use the same architecture for our space partitioner S. We use pre-activation Residual-Nets with 20 convolutional bottleneck blocks with 3 convolution layers each and kernel sizes of 3×3, 1×1, 3×3 respectively and the ELU [86] nonlinearity. The network has 4 down-sampling stages, every 4 blocks where a dimension squeezing operation is used to decrease the spatial resolution. We use 160 channels for all the blocks. We do not use any initial padding due to our theoretical requirements. The negative slope of LeakyReLU is set as 0.2. In fact we can use a soft version of LeakyReLU if it is critical to guarantee the C^1 constraint of ϕ. We train

Figure 2.5: Effect of changing the guide's weight λ in equation 2.13 on PGMGAN performance. $\lambda = 0$ corresponds to the partition+GAN.



our pretext network for 500 epochs with momentum SGD and a weight decay of 3e-5, learning rate of 0.4 with cosine scheduling, momentum of 0.9, and batch size of 400 for CIFAR-10 and 200 for STL-10. The final space partitioner is trained for 100 epochs using Adam [72] with a learning rate of 1e-4 and batch size of 128. The weights in equation 2.10 are set to $\alpha = 5$ and $\beta =$ 1e-3.

Table 2.4: GANs architecture for 32×32 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed($Partition_{ID}$) $\in \mathbb{R}^{128}$
dense, $4 \times 4 \times 256$
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, 3×3 Conv, 3 Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{32 \times 32 \times 3}$
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128
ReLU, Global sum pooling
Embed($Partition_{ID}$)$\cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

Table 2.5: GANs architecture for 48×48 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed(*Partition$_{ID}$*) $\in \mathbb{R}^{128}$
dense, $3 \times 3 \times 1024$
ResBlock up 512
ResBlock up 256
ResBlock up 128
ResBlock up 64
BN, ReLU, 3×3 Conv, 3 Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{48 \times 48 \times 3}$
ResBlock down 64
ResBlock down 128
ResBlock down 256
ResBlock down 512
ResBlock 1024
ReLU, Global sum pooling
Embed(*Partition$_{ID}$*)$\cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

Table 2.6: GANs architecture for 128×128 images. "ch" represents the channel width multiplier and is set to 96.

$z \in \mathbb{R}^{120} \sim \mathcal{N}(0, I)$ Embed(*Partition$_{ID}$*) $\in \mathbb{R}^{128}$
Linear $(20 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock up $2ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down $ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock down $2ch \rightarrow 4ch$
ResBlock down $4ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed(*Partition$_{ID}$*)$\cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

Generative model. Following SN-GANs [71] for image generation at resolution 32 or 48, we use the architectures described in Tables 2.4 and 2.5. Generators/discriminators are different from

each other in first-layer/last-layer by having different partition ID embeddings, (which in fact acts as the condition). We use Adam optimizer with a batch size of 100. For the coefficient of guide λ we utilized linear annealing during training, decreasing form 6.0 to 0.0001. Both G's and D networks are initialized with a normal $\mathcal{N}(0, 0.02\boldsymbol{I})$. For all GAN's experiments, we use Adam optimizer [72] with $\beta_1 = 0$, $\beta_2 = 0.999$ and a constant learning rate 2 for both G and D. The number of D steps per G step training is 4.

For ImageNet experiment, we adopt the full version of BigGAN model architecture [26] described in Table 2.6. In this experiment, we apply the shared class embedding for each CBN layer in G model, and feed noise z to multiple layers of G by concatenating with partition ID embedding vector. We add Self-Attention layer with the resolution of 64. Moreover, we employ orthogonal initialization for network parameters [15]. We use batch size of 256 and set the number of accumulations to 8.

Evaluation. It has been shown that [87, 76] when the sample size is not large enough, both FID and IS are biased, therefor we use N=50,000 samples for computing both IS and FID metrics. We also use the official TensorFlow scripts for computing FID.

Computationally complexity. PGMGAN requires additional computation for training the partitioner, and at GANs' training time for computing the guide. On the other hand, due to the higher quality gradients provided by the guide, requires fewer epoch for the GANs to converge, e.g., over all, compared to conditional model on CIFAR, it takes $\sim$1.6$\times$ in seconds.

2.5.6 Additional Experiments:

Additionally, since SelfCondGAN [19] uses certain features from the discriminator, it is not trivial to adopt to other architectures. Thus, we trained PGMGAN with the same G/D architecture on CIFAR10 yielding an FID of 10.65.

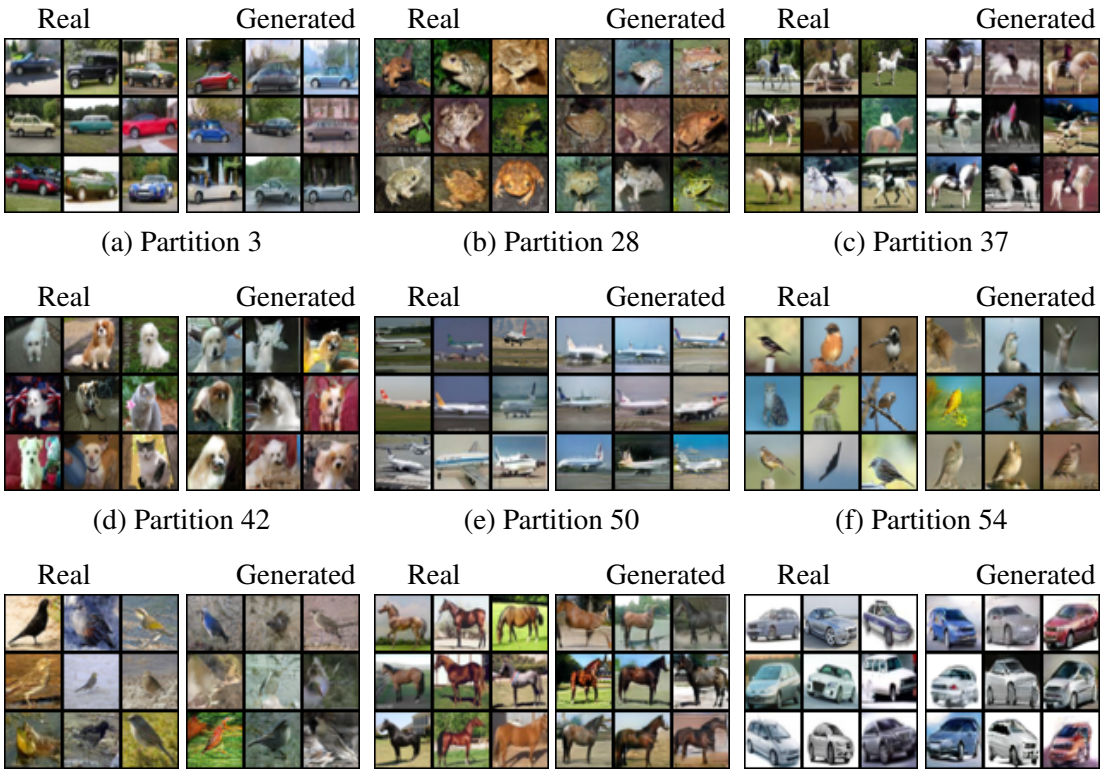
Partitioning method

One way to assess the quality of the space partitioner is by measuring its performance in placing semantically similar images in the same partition. To that end, we use the well-accepted clustering metric Normalized Mutual Information (NMI). NMI is a normalization of the Mutual

Information (MI) between the true and inferred labels. This metric is invariant to permutation of the class/partition labels and is always between 0 and 1, with a higher value suggesting a higher quality of partitioning. Table 2.5.6 compares the clustering performance of our method to the-stat-of-the-art partition-based GAN method Liu et. [19], which clearly shows superiority of our method.

2.5.7 Additional Qualitative Results

Additional qualitative results We present more samples of our method showing both the partitioner and generative model's performance. Figure 2.6, Figure 2.7 and Figure 2.8 visualize the sample diversity and quality of our method on CIFAR-10, STL-10 and ImageNet.

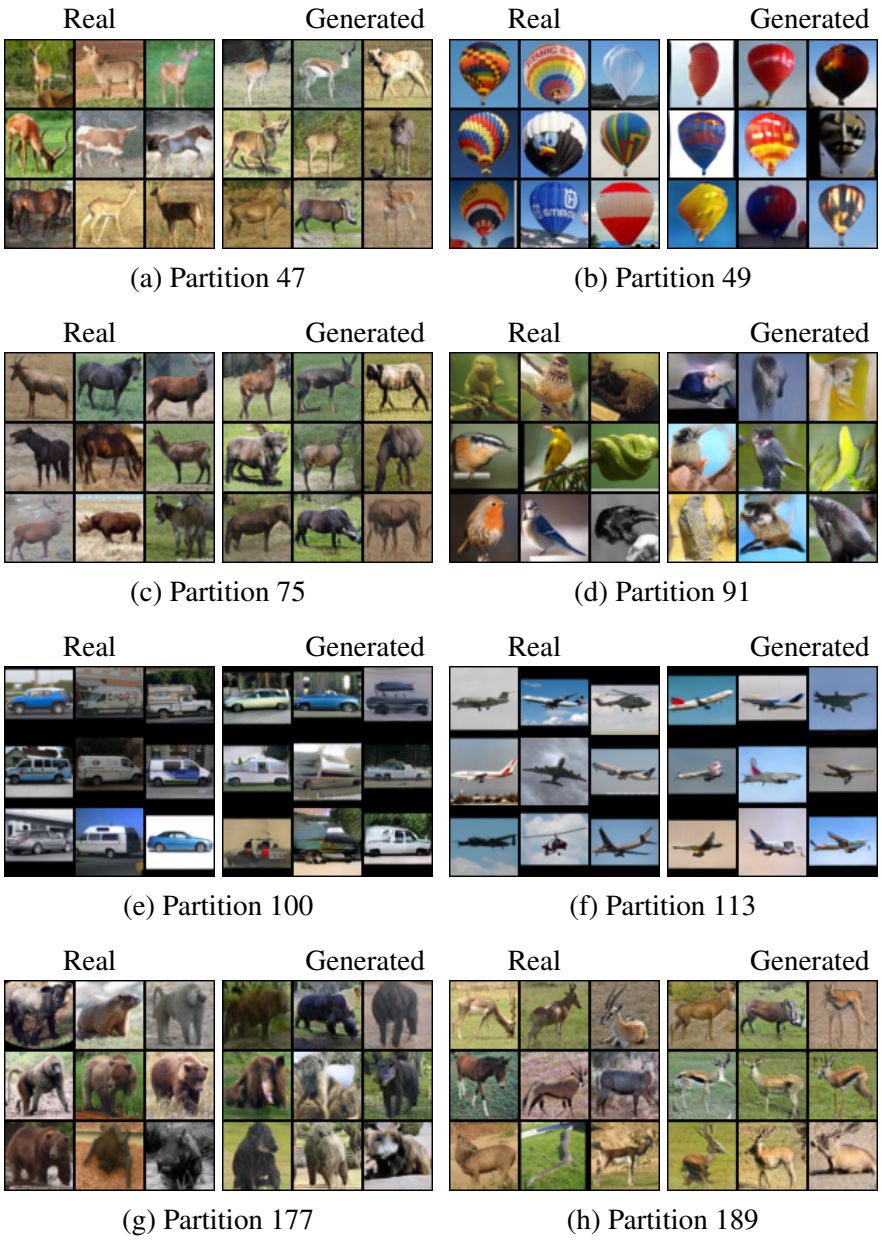


(a) Partition 3 (b) Partition 28 (c) Partition 37

(d) Partition 42 (e) Partition 50 (f) Partition 54

(g) Partition 82 (h) Partition 86 (i) Partition 94

Figure 2.6: Extra examples of unsupervised partitioning and their corresponding real/generated samples on CIFAR-10 dataset.



(a) Partition 47 (b) Partition 49

(c) Partition 75 (d) Partition 91

(e) Partition 100 (f) Partition 113

(g) Partition 177 (h) Partition 189

Figure 2.7: Extra examples of unsupervised partitioning and their corresponding real/generated samples on STL-10 dataset.

(a) Partition 034 (b) Partition 159

(c) Partition 455 (d) Partition 642

Figure 2.8: Examples of generated samples on unsupervised ImageNet 128×128 dataset.

3. SYNT++: UTILIZING IMPERFECT SYNTHETIC DATA TO IMPROVE SPEECH RECOGNITION

3.1 Introduction

Collecting and annotating datasets for training speech recognition models is hard and expensive. An alternative approach is to generate a synthetic dataset using speech synthesis models [88, 89, 90, 91, 92, 93, 3]. However, training recognition models with synthetic data is not trivial. Figure 3.1 shows an illustration of the joint distribution of speech signals and corresponding text labels – black is the true data distribution and green is the synthetic data distribution. In general, we do not have access to the true data distribution, but have finite sample approximation through a collected real dataset. Although the quality of synthetic speech (synthesized via a controllable generative model) has improved significantly, there is still a gap between the synthetic and the true data distributions. We can characterize this gap into 4 different regions as shown in Figure 3.1: (i) artifacts (e.g. structured noise, speech/content mismatch, unrealistic styles in synthetic data) where the synthesized samples are outside the support of true distribution, (ii) over-sampled region where the synthetic distribution has more mass and (iii) under-sampled region where the synthetic distribution has less mass compared to the true distribution, and (iv) missing samples where the synthetic distribution has zero mass inside the support of the true distribution (e.g. missing speaking styles, accents).

In this paper, we present Synt++, an improved algorithm to utilize synthetic data for training speech recognition models. Synt++ combines two novel techniques. The first technique is a rejection sampling algorithm [94, 95, 96] that modifies the sampling process of the synthesis model (the generative model) to make the synthesized data distribution closer to the true data distribution. Specifically, we train a discriminator function to classify samples as real or synthetic. We then use the probability of real measure predicted by this discriminator to stochastically accept/reject synthesized samples. The rejection sampling addresses the first three problems discussed above by

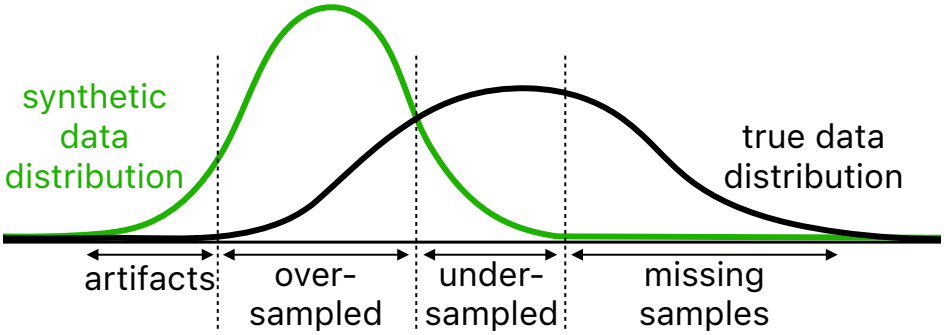


Figure 3.1: The joint distributions of speech and corresponding text. The gap between synthetic and true data distributions can be partitioned into four regions. See text for details.

rejecting artifacts, and under/over sampling data points in over/under-sampled regions, respectively. Note that the rejection sampling cannot correct for the missing samples since the synthesis model does not have support and, hence, cannot synthesize the samples in this region.

The second technique accounts for the remaining gap during training by using separate Batch Normalization (BN) statistics for real and synthetic data. BN is a commonly used technique to normalize the features during training to address covariate shift and improve convergence. However, the BN statistics estimated using a combination of synthetic and real data becomes biased due to the distribution gap (e.g., in Figure 3.1, the mean of the combined real/synthetic distribution is shifted left and the variance is larger than the true distribution). To prevent this problem, we estimate real and synthetic data BN statistics separately, and utilize only the real BN statistics during inference. Moreover, this process helps to bridge the domain gap since synthetic and real data are normalized separately to have similar statistics.

3.2 Related work

State-of-the-art neural Text-to-Speech (TTS) architectures, such as Tacotron 2 [97] and FastSpeech 2 [98], are capable of generating speech indistinguishable from human speech. Controllable speech synthesis extends the TTS models beyond modeling the voice of a single or a few speakers to the entire style space of speech [93, 92, 3, 99]. Using controllable synthesis models, we can expand the acoustic variations of synthetic speech or use a bigger text corpus to add data for unseen content.

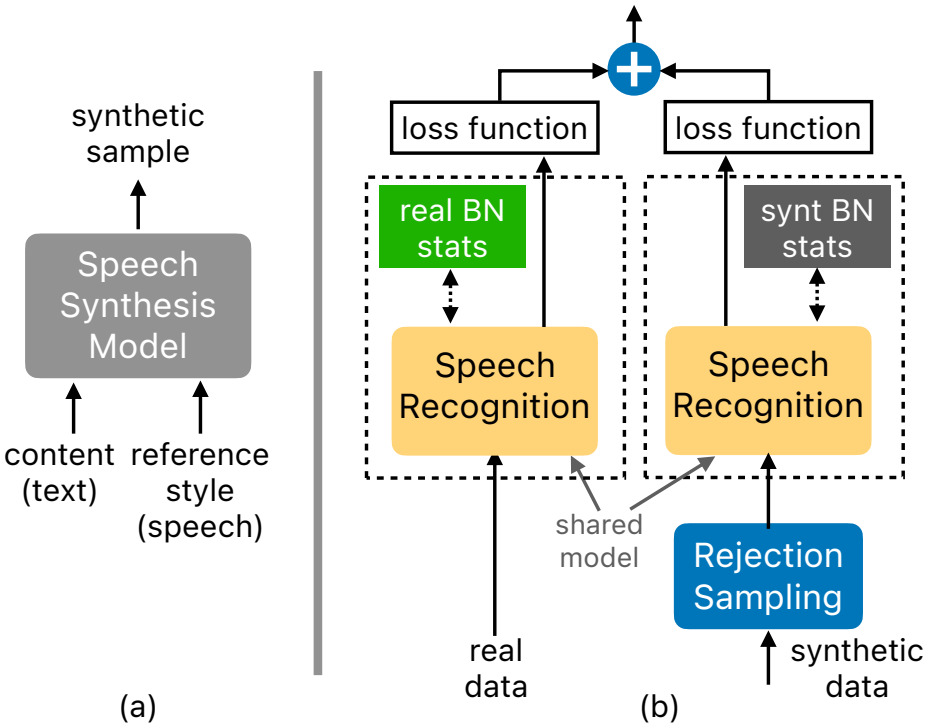


Figure 3.2: Method overview. (a) To synthesize data, we use a controllable generative model [3] that takes a text and a reference speech as input and utters the text in the style of the given speech. (b) During training of the recognition model, we use rejection sampling and separate BN statistics to address the distribution gap between the real and the synthetic data.

Previous works have used TTS models to improve the training of speech recognition models. One group of work relies on the TTS models and self/semi-supervised learning techniques to exploit unpaired speech and text data in the ASR training process [100, 101, 102, 103, 104]. Another group leverages TTS models to create more training data when the amount of real data is limited [105, 106, 107, 108, 109]. Some other works have focused on specific tasks, such as out-of-vocabulary word recognition [110], and keyword detection [111, 112]. These methods, however, do not address the imperfections of the speech generated by the synthesis models.

The idea of utilizing synthetic data to improve a recognition model has also been explored in the computer vision domain. In [113], the authors proposed to refine synthetic images via adversarial training to bridge the distribution gap. In [114], the authors showed that a generative model could be trained to augment images of unseen classes. In [115], authors use adversarial examples to improve image recognition.

3.3 Method

3.3.1 Controllable Speech Generative Model

Controllable speech generative models can generate speech in a given style, where the content is specified by an input text. As shown in Figure 3.1, for effective learning with synthetic data, we need a generative model that has *as small gap as possible* with the true distribution. Therefore, the generative model should be able to synthesize any text in a wide range of speaking styles in various environmental conditions such as background noise, microphone response, etc. without introducing artifacts to the signal. To this end, we utilize a recently introduced auto regressive controllable generative model called Style Equalization [3] that enables learning a controllable generative model in the wild (thousands of speakers recorded in various conditions) without requiring style labels. As shown in Figure 3.2(a), the model takes a text and a reference style speech as input and utters the text in the style of the given reference. In this model, the style is broadly defined, not only the speaker's voice characteristics but also all aspects of the reference speech sample, including background noise, echo, microphone response, etc., which are necessary to have a smaller distribution gap (particularly missing samples). This model also allows sampling styles from a prior distribution that is important for sampling new styles not contained in the dataset. We refer readers to [3] for a detailed description of this model.

3.3.2 Rejection Sampling

Let $\boldsymbol{x}$ be a speech signal and y be the corresponding token label, and $p_d(\boldsymbol{x}, y)$, $p_g(\boldsymbol{x}, y)$ denote the joint true and synthetic data distributions, respectively. We use the synthesizer as proposal distribution and use rejection sampling to make it closer to the true distribution. Following standard rejection sampling [94], a synthetic data $(\boldsymbol{x}, y)$ is accepted with probability $\frac{p_d(\boldsymbol{x},y)}{Mp_g(\boldsymbol{x},y)}$, where scalar $M > 0$ is selected such that $Mp_g(\boldsymbol{x}, y) > p_d(\boldsymbol{x}, y), \forall(\boldsymbol{x}, y)$ in domain of p_d, i.e., $M = \max_{\boldsymbol{x},y} \frac{p_d(\boldsymbol{x},y)}{p_g(\boldsymbol{x},y)}$. Azadi et. al. [95] proposed a rejection sampling method to match the distribution of the images from a generative adversarial network to the true distribution; however, their algorithm does not consider the joint distribution of $(\boldsymbol{x}, y)$ that is needed to train downstream recognizers.

To approximate the ratio $\frac{p_d(\boldsymbol{x},y)}{p_g(\boldsymbol{x},y)}$, we train a discriminator, $D(\boldsymbol{x}, y)$, that takes a speech signal

and the corresponding text and produces probability of $(\boldsymbol{x}, y)$ coming from true distribution, i.e., $D(\boldsymbol{x}, y) \in [0, 1]$. When D is trained to its optimal value D^*, it satisfies ([116, 95, 96])

$$D^*(\boldsymbol{x}, y) = \frac{p_d(\boldsymbol{x}, y)}{p_d(\boldsymbol{x}, y) + p_g(\boldsymbol{x}, y)}, \tag{3.1}$$

which can be re-arranged to yield

$$r(\boldsymbol{x}, y) := \frac{p_d(\boldsymbol{x}, y)}{p_g(\boldsymbol{x}, y)} = \frac{D^*(\boldsymbol{x}, y)}{1 - D^*(\boldsymbol{x}, y)}. \tag{3.2}$$

To train D, we minimize

$$\mathcal{L}_D = \mathbb{E}_{(\boldsymbol{x},y)\sim p_d}(\log(D(\boldsymbol{x}, y))) + \mathbb{E}_{(\boldsymbol{x},y)\sim p_g}(1 - \log(D(\boldsymbol{x}, y))),$$

which is equivalent to training a two class classification model, where the positive class (label 1) corresponds to the real samples and the negative class (label 0) corresponds to the synthetic samples. Note that, we model the joint speech and label distribution, therefore D should also utilize whether the speech signal matches the token labels. To this end, we use a pre-trained ASR model, A to predict the token labels, $\hat{y}$, of the speech signal, i.e., $\hat{y} = A(\boldsymbol{x})$. Then, we characterize the discrepancy between $\hat{y}$ and y by a 5-dimensional feature vector $\Phi(\boldsymbol{x}, y)$, whose elements include the cross-entropy (CE) loss, connectionist temporal classification loss, word error rate (WER), and number of tokens in y and $\hat{y}$. Finally, we train a two-layer DNN, $D_{\boldsymbol{\theta}}$, that takes $\Phi(\boldsymbol{x}, y)$ as input and produces the probability of the sample being real

$$D(\boldsymbol{x}, y) := D_{\boldsymbol{\theta}}(\Phi(\boldsymbol{x}, y)), \tag{3.3}$$

where $\boldsymbol{\theta}$ denotes the parameters of the DNN.

Practically, we need to compute M and address the fact that the rejection sampling can be slow to select a fixed number of samples. First, we estimate an initial value of M by computing the maximum value of p_d/p_g (using D^* and Eq. 3.2) over a few (approximately 200) generated

samples. Following [95], to compute the acceptance probability for each generated sample, $(\boldsymbol{x}, y)$, we first compute the discriminator score $D^*(\boldsymbol{x}, y)$ and then set $M \leftarrow \max(M, p_d(\boldsymbol{x}, y)/p_g(\boldsymbol{x}, y)) = \max(M, \frac{D^*(\boldsymbol{x},y)}{1-D^*(\boldsymbol{x},y)})$. We stop the sampling process after accepting N synthetic samples. Note that, the proposed rejection sampling technique applies only to the domain where the generative model has support and ignores the missing samples region shown in Figure 3.1.

3.3.3 Double Batch Normalization Statistics

A BN layer [117] computes the mean and the standard deviation of the intermediate features and normalizes these features by subtracting the mean and dividing by the standard deviation within a batch. During training, the model uses means and variances computed over the samples in a mini-batch. A running average of the BN statistics, which we call running statistics, is computed to be used during inference.

Naively using BN will update the running statistics with samples from both the real and the synthetic distributions and, due to the distribution gap, the estimates will be biased (i.e. the estimated BN statistics will not match the true data distribution). Instead, we form the mini-batches consisting of only the real samples or only the synthetic samples, and compute two sets of running statistics – one from the real mini-batches and the other from the synthetic mini-batches. Since the test data (during inference) consists of only real samples, we discard the synthetic statistics and use only the real statistics. Note that, we share the affine parameters of the BN between synthetic and real batches.

The proposed technique is similar to Xie et al. [115] that used auxiliary BN layers in an adversarial learning framework where images augmented with adversarial perturbations were used to improve model accuracy. In contrast, we use separate BN statistics to improve model training with synthetic data.

3.4 Experiments

For ASR experiments, we use the LibriSpeech dataset [118], which contains $1,000$ hours of speech from public domain audiobooks. Following the standard protocol, we evaluate our method

on the full training set (LibriSpeech 960h) and a subset containing clean speech with only US English accents (LibriSpeech 100h). To generate synthetic datasets, we use Style Equalization [3] that is trained with LibriTTS dataset [119]. LibriTTS is a subset of LibriSpeech dataset and contains 555 hours of speech data. To study the effect of model size on synthetic data training, we train two ASR models – a large model with 116 million parameters (Large 116M) and a smaller model with 22 million parameters (Regular 22M)

ASR model: We use the implementation from ESPNet [120]) for an end-to-end ASR model. The ASR model is composed of a conformer-based encoder [121] and a transformer-based decoder [89]. Setting the encoder dimension to 144 and 512, we construct the regular (22M) and the large (116M) models. We apply SpecAugment [122] to all speech samples to further enhance the acoustic diversity. The model checkpoint of each epoch is saved, and the final model is produced by averaging the 10 checkpoints with the best validation accuracy. All ASR models are evaluated without a language model.

Baselines: For synthetic-only training, we compare naively using synthetic data with the proposed rejection sampling (Section 3.3.2). Note that when the training data consists of only synthetic samples, we do not have any real data to compute the running BN statistics. Hence, we do not use the double BN when training only with synthetic data. For joint (real,synt) training, our baseline is naively adding the synthetic data to training. We compare this baseline with the proposed method (real, synt++) where we use the rejection sampling algorithm described in Section 3.3.2 and also use separate BN statistics for the real and the synthetic samples (Section 3.3.3).

Synthetic data with the same text corpus as the real data: First, we study the effectiveness of the synthetic data when the synthetic samples are used to increase the acoustic variation in the training data. To this end, we use the same training corpus as the real data in LibriSpeech 960h. Specifically, for each sentence y_i in the training dataset, we take a random real training sample, $\boldsymbol{x}_j$, as the style reference, and generate synthetic sample $\boldsymbol{x}'_i$. For both Synt and Synt++ (using rejection sampling), we obtain a synthetic dataset containing 960 hours of speech.

The WERs of the recognition models are reported in Table 3.1 on the two official test sets

	Real	Synt	Synt++	Real, Synt	Real, Synt++
Regular (22M)					
test-clean	3.7 ± 0.14	7.3 ± 0.17	7.0 ± 0.05	3.4 ± 0.05	$\mathbf{3.2} \pm 0.08$
test-other	9.5 ± 0.12	21.0 ± 0.12	20.0 ± 0.17	9.3 ± 0.21	$\mathbf{8.5} \pm 0.05$
Large (116M)					
test-clean	2.9 ± 0.05	6.3 ± 0.05	5.4 ± 0.12	3.0 ± 0.05	$\mathbf{2.6} \pm 0.05$
test-other	6.9 ± 0.08	18.2 ± 0.17	17.2 ± 0.17	7.4 ± 0.17	$\mathbf{6.5} \pm 0.05$

Table 3.1: ASR results on LibriSpeech 960h dataset. The reported numbers are percentage WER (lower is better). Synt++ significantly improves training with synthetic data.

	LibriSpeech 100h			LibriSpeech 960h		
	Real	Real, Synt	Real, Synt++	Real	Real, Synt	Real, Synt++
test-clean	7.7 ± 0.08	4.3 ± 0.08	$\mathbf{4.0} \pm 0.08$	2.9 ± 0.05	2.7 ± 0.08	$\mathbf{2.4} \pm 0.05$
test-other	20.9 ± 0.64	13.2 ± 0.21	$\mathbf{13.2} \pm 0.08$	6.9 ± 0.08	7.0 ± 0.25	$\mathbf{6.3} \pm 0.05$

Table 3.2: Effect of using an extended text corpus.

(test-clean and test-other). Synt++ gives significant improvement over naively adding synthetic data to the training (Synt).

Synthetic data with extended corpus: We study the effect of synthesizing speech from an extended text corpus in Table 3.2. In addition to LibriSpeech 960h, we also report the results when we have a smaller real dataset (LibriSpeech 100h). For LibriSpeech 100h, we use the text corpus of LibriSpeech 960h and synthesize 960 hours of synthetic data. For LibriSpeech 960h dataset, we double its text corpus by adding the text from the language model corpus of LibriSpeech and synthesize 960 hours of synthetic data. In both cases, extended corpus significantly improves the performance. Particularly, for the smaller dataset, we observe 48% relative reduction in WER (7.7% to 4.0%), compared to real-only training.

Ablation study: To isolate the effect of the rejection sampling and the double BN, we conduct an ablation study in Table 3.3 on LibriSpeech 960h dataset. As seen in this table, both the rejection sampling and the double BN statistics are important to effectively utilize the synthetic samples.

We also evaluate the proposed approach on keyword detection task using the Speech Command

	Regular (22M)		Large (116M)	
	test-clean	test-other	test-clean	test-other
Real, Synt	3.4 ± 0.05	9.3 ± 0.21	3.0 ± 0.05	7.4 ± 0.17
+ rejection	3.4 ± 0.05	9.0 ± 0.12	2.8 ± 0.02	7.0 ± 0.05
+ dbl BN	3.3 ± 0.02	8.5 ± 0.02	2.7 ± 0.05	6.7 ± 0.02
+ rej. + dbl BN	$\mathbf{3.2} \pm 0.08$	$\mathbf{8.5} \pm 0.05$	$\mathbf{2.6} \pm 0.05$	$\mathbf{6.5} \pm 0.05$

Table 3.3: Ablation study on the ASR task with LibriSpeech 960h dataset.

keywords	Real	Synt	Synt++	Real, Synt	Real, Synt++
'down'	9.6 ± 0.8	11.2 ± 0.1	8.9 ± 0.9	5.6 ± 0.2	$\mathbf{5.0} \pm 0.2$
'no'	13.5 ± 0.8	14.5 ± 0.7	11.0 ± 0.2	7.8 ± 0.4	$\mathbf{6.4} \pm 0.4$
'stop'	3.5 ± 0.3	11.2 ± 0.6	9.6 ± 0.4	2.1 ± 0.3	$\mathbf{1.5} \pm 0.1$

Table 3.4: Keyword detection results on Speech Command dataset. The reported numbers are average false accept rate in percentage (lower is better).

dataset [123]. This dataset consists of 35 keywords from multiple speakers, and samples from various background noises. For each keyword, we split the train and test subset based on the speaker identities, i.e. training and test samples did not have common speakers. We chose 3 keywords ('down', 'no', 'stop') to study the improvements with synthetic data. Each of these keywords contains approximately 2500 training samples and 800 test samples. We set up the keyword detection task as a 2-class classification problem, where the positive class is one of the keywords and the negative class contains all the other 34 keywords and the background noise. Each sample consists of approximately 1 second of audio, which is converted to a mel-spectrogram before giving it as input to a ResNet model with approximately 6k parameters. The model is trained with cross-entropy loss.

Sampling synthetic data: For each keyword, we generated 10k samples by randomly selecting the reference styles from the real dataset. For the positive class, we generated an additional 100k samples. For both synthetic and real training, we added approximately 5k samples from background noise to the negative class. Since the detection model predicts the keyword probability (unlike a token sequence in the ASR task), we use only the CE loss in Equation 3.3 to compute D^*.

Metric: Since keyword detection is a 2-class problem, we can compute false reject rate (FRR) and false accept rate (FAR) at different thresholds. The detection error tradeoff (DET) curve is commonly used for keyword detection tasks [124]. Since lower FRR is preferred for keyword detection, we report average FAR for FRR values over the range of 05%, which is equivalent to the area under the curve (AUC) of the DET curve over the FRR range of 05%.

Results: We report the average FAR for keywords in Table 3.4 and observe significant improvements of the proposed approach, compared to naively using the synthetic data.

4. SUMMARY AND CONCLUSIONS

In the first part of the thesis, we introduce a differentiable space partitioner to alleviate the GAN training problems, including mode connecting and mode collapse. The intuition behind how this works is twofold. The first reason is that an efficient partitioning makes the distribution on each region simpler, making its approximation easier. Thus, we can have a better approximation as a whole, which can alleviate both mode collapse and connecting problems. The second intuition is that the space partitioner can provide extra gradient, assisting the discriminator in training the mixture of generators. This is especially helpful when the discriminator's gradient is unreliable. However, it is crucial to have theoretical guarantees that this extra gradient does not deteriorate the GAN training convergence in any way. We identify a sufficient theoretical condition for the space partitioner (in the functional space), and we realize that condition empirically by an architecture design for the space partitioner. Our experiments on natural images show the proposed method improves existing ones in terms of both FID and IS. For future work, we would like to investigate using the space partitioner for the supervised regime, where each data label has its own partitioning. Designing a more flexible architecture for the space partitioner such that its guide function does not have local optima is another direction we hope to explore.

And in the second part, we presented Synt++, a new algorithm to train speech recognition models using synthetic data. We applied Synt++ to ASR and keyword detection tasks and obtained significant improvements over training with real-only and real+synthetic datasets.

And for the future work, we would like to investigate, generative models in Knowledge graphs[125, 125], time series[126, 127], network data[128, 129] and its combination with decision trees[130].

REFERENCES

[1] M. Armandpour, A. Sadeghian, C. Li, and M. Zhou, "Partition-guided gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5099–5109, 2021.

[2] T.-Y. Hu, M. Armandpour, A. Shrivastava, J.-H. R. Chang, H. Koppula, and O. Tuzel, "Synt++: Utilizing imperfect synthetic data to improve speech recognition," *ICASSP*, 2022.

[3] J.-H. R. Chang, A. Shrivastava, H. Koppula, X. Zhang, and O. Tuzel, "Style equalization: Unsupervised learning of controllable generative sequence models," *arXiv preprint arXiv:2110.02891*, 2021.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[5] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *International Conference on Machine Learning*, pp. 1060–1069, 2016.

[6] H. Narayanan and S. Mitter, "Sample complexity of testing the manifold hypothesis," in *Advances in neural information processing systems*, pp. 1786–1794, 2010.

[7] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017.

[9] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, pp. 1486–1494, 2015.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017.

[11] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

[12] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?," in *International Conference on Machine Learning*, pp. 3478–3487, 2018.

[13] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[14] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016.

[16] M. Khayatkhoei, M. Singh, and A. Elgammal, "Disconnected manifold learning for generative adversarial networks," *arXiv preprint arXiv:1806.00880*, 2018.

[17] U. Tanielian, T. Issenhuth, E. Dohmatob, and J. Mary, "Learning disconnected manifolds: a no gans land," 2020.

[18] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," in *Advances in neural information processing systems*, pp. 1498–1507, 2018.

[19] S. Liu, T. Wang, D. Bau, J.-Y. Zhu, and A. Torralba, "Diverse image generation via self-conditioned gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14286–14295, 2020.

[20] A. Srivastava, L. Valkoz, C. Russell, M. U. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," in *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

[21] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[22] A. Coates, A. Ng, and H. Lee, "An Analysis of Single Layer Networks in Unsupervised Feature Learning," in *AISTATS*, 2011. `https://cs.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf`.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

[24] J. L. Kelley, *General topology*. Courier Dover Publications, 2017.

[25] M. Ledoux, "Isoperimetry and gaussian analysis," in *Lectures on probability theory and statistics*, pp. 165–294, Springer, 1996.

[26] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2019.

[27] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*, pp. 7354–7363, PMLR, 2019.

[28] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*, pp. 1989–1998, PMLR, 2018.

[29] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 172–189, 2018.

[30] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, 2017.

[31] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *ICLR*, 2018.

[32] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[33] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *European conference on computer vision*, pp. 318–335, Springer, 2016.

[34] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

[35] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[36] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.

[37] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Advances in Neural Information Processing Systems*, 2017.

[38] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[39] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International conference on information processing in medical imaging*, pp. 146–157, Springer, 2017.

[40] N. Killoran, L. J. Lee, A. Delong, D. Duvenaud, and B. J. Frey, "Generating and designing dna with deep generative models," 2017.

[41] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *ICML*, 2018.

[42] M. Marouf, P. Machart, V. Bansal, C. Kilian, D. S. Magruder, C. F. Krebs, and S. Bonn, "Realistic in silico generation and augmentation of single-cell rna-seq data using generative adversarial networks," *Nature communications*, vol. 11, no. 1, pp. 1–12, 2020.

[43] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network," *Proc. Interspeech 2017*, pp. 3642–3646, 2017.

[44] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *International Conference on Learning Representations*, 2017.

[45] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," in *International Conference on Learning Representations*, 2017.

[46] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *International Conference on Learning Representations*, 2017.

[47] M. Lucic, M. Tschannen, M. Ritter, X. Zhai, O. Bachem, and S. Gelly, "High-fidelity image generation with fewer labels," *International conference on machine learning*, 2019.

[48] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *International Conference on Learning Representations*, 2018.

[49] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "Mgan: Training generative adversarial nets with multiple generators," in *International Conference on Learning Representations*, 2018.

[50] J. N. Kundu, M. Gor, D. Agrawal, and R. V. Babu, "Gan-tree: An incrementally learned hierarchical generative framework for multi-modal data distributions," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8191–8200, 2019.

[51] A. Sage, E. Agustsson, R. Timofte, and L. Van Gool, "Logo synthesis and manipulation with clustered generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5879–5888, 2018.

[52] S. Gurumurthy, R. K. Sarvadevabhatla, and R. V. Babu, "Deligan: Generative adversarial networks for diverse and limited data.," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[53] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International conference on machine learning*, pp. 2642–2651, 2017.

[54] S. Azadi, C. Olsson, T. Darrell, I. Goodfellow, and A. Odena, "Discriminator rejection sampling," *International Conference on Learning Representations*, 2019.

[55] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

[56] T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby, "Self-supervised gans via auxiliary rotation loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12154–12163, 2019.

[57] Z. Zhao, Z. Zhang, T. Chen, S. Singh, and H. Zhang, "Image augmentations for gan training," *arXiv preprint arXiv:2006.02595*, 2020.

[58] H. Zhang, Z. Zhang, A. Odena, and H. Lee, "Consistency regularization for generative adversarial networks," *International Conference on Learning Representations*, 2020.

[59] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *Advances in Neural Information Processing Systems*, 2020.

[60] T. Miyato and M. Koyama, "cgans with projection discriminator," *International Conference on Learning Representations*, 2018.

[61] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[62] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big self-supervised models are strong semi-supervised learners," *arXiv preprint arXiv:2006.10029*, 2020.

[63] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

[64] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

[65] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, "Scan: Learning to classify images without labels," in *European Conference on Computer Vision (ECCV)*, 2020.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[67] W. Rudin *et al.*, *Principles of mathematical analysis*, vol. 3. McGraw-hill New York, 1964.

[68] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," in *International Conference on Machine Learning*, pp. 573–582, 2019.

[69] M. Gong, Y. Xu, C. Li, K. Zhang, and K. Batmanghelich, "Twin auxilary classifiers gan," in *Advances in neural information processing systems*, pp. 1330–1339, 2019.

[70] R. Shu, H. Bui, and S. Ermon, "Ac-gan learns a biased distribution," in *NIPS Workshop on Bayesian Deep Learning*, 2017.

[71] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[72] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[73] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[74] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, 2019.

[75] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," in *Advances in Neural Information Processing Systems*, pp. 5228–5237, 2018.

[76] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," in *Advances in neural information processing systems*, pp. 700–709, 2018.

[77] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[78] Y. Tian, Q. Wang, Z. Huang, W. Li, D. Dai, M. Yang, J. Wang, and O. Fink, "Off-policy reinforcement learning for efficient and effective gan architecture search," *ECCV*, 2020.

[79] T. Nguyen, T. Le, H. Vu, and D. Phung, "Dual discriminator generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2670–2680, 2017.

[80] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," 2016.

[81] H. He, H. Wang, G.-H. Lee, and Y. Tian, "Probgan: Towards probabilistic gan with theoretical guarantees," 2019.

[82] N.-T. Tran, T.-A. Bui, and N.-M. Cheung, "Dist-gan: An improved gan using distance constraints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 370–385, 2018.

[83] W. Wang, Y. Sun, and S. Halgamuge, "Improving MMD-GAN training with repulsive loss function," in *International Conference on Learning Representations*, 2019.

[84] H. Wang and J. Huan, "Agan: Towards automated design of generative adversarial networks," *arXiv preprint arXiv:1906.11080*, 2019.

[85] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "Autogan: Neural architecture search for generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3224–3234, 2019.

[86] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[87] M. J. Chong and D. Forsyth, "Effectively unbiased fid and inception score and where to find them," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6070–6079, 2020.

[88] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," *arXiv preprint arXiv:1906.01083*, 2019.

[89] N. Li, S. Liu, Y. Liu, S. Zhao, and P. Shi, "Neural speech synthesis with transformer network," in *AAAI*, 2019.

[90] S. Ma, D. Mcduff, and Y. Song, "A generative adversarial network for style modeling in a text-to-speech system," in *ICLR*, 2019.

[91] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu, *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Proc. NIPS*, 2018.

[92] T.-Y. Hu, A. Shrivastava, O. Tuzel, and C. Dhir, "Unsupervised style and content separation by minimizing mutual information for speech synthesis," in *Proc. ICASSP*, 2020.

[93] Y. Wang, D. Stanton, Y. Zhang, R.-S. Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *Proc. ICML*, 2018.

[94] C. M. Bishop, *Pattern recognition and machine learning, 5th Edition*. Information science and statistics, Springer, 2007.

[95] S. Azadi, C. Olsson, T. Darrell, I. Goodfellow, and A. Odena, "Discriminator rejection sampling," in *Proc. ICLR*, 2019.

[96] A. Grover, R. Gummadi, M. Lazaro-Gredilla, D. Schuurmans, and S. Ermon, "Variational rejection sampling," in *Proc. AISTATS*, 2018.

[97] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. ICASSP*, 2018.

[98] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *Proc. ICLR*, 2021.

[99] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," in *Proc. ICLR*, 2018.

[100] A. Tjandra, S. Sakti, and S. Nakamura, "End-to-end feedback loss in speech chain framework via straight-through estimator," in *Proc. ICASSP*, 2019.

[101] M. K. Baskar, S. Watanabe, R. Astudillo, T. Hori, L. Burget, and J. Černockỳ, "Semi-supervised sequence-to-sequence asr using unpaired speech and text," *arXiv preprint arXiv:1905.01152*, 2019.

[102] T. Hori, R. Astudillo, T. Hayashi, Y. Zhang, S. Watanabe, and J. Le Roux, "Cycle-consistency training for end-to-end speech recognition," in *Proc. ICASSP*, 2019.

[103] T. Hayashi, S. Watanabe, Y. Zhang, T. Toda, T. Hori, R. Astudillo, and K. Takeda, "Back-translation-style data augmentation for end-to-end asr," in *IEEE Spoken Language Technology Workshop (SLT)*, 2018.

[104] M. K. Baskar, L. Burget, S. Watanabe, R. F. Astudillo, *et al.*, "Eat: Enhanced asr-tts for self-supervised speech recognition," in *Proc. ICASSP*, 2021.

[105] N. Rossenbach, A. Zeyer, R. Schlüter, and H. Ney, "Generating synthetic audio data for attention-based speech recognition systems," in *Proc. ICASSP*, 2020.

[106] C. Du and K. Yu, "Speaker augmentation for low resource speech recognition," in *Proc. ICASSP*, 2020.

[107] A. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. Moreno, Y. Wu, and Z. Wu, "Speech recognition with augmented synthesized speech," in *ASRU*, 2019.

[108] M. Mimura, S. Ueno, H. Inaguma, S. Sakai, and T. Kawahara, "Leveraging sequence-to-sequence speech synthesis for enhancing acoustic-to-word speech recognition," in *IEEE Spoken Language Technology Workshop (SLT)*, 2018.

[109] Z. Chen, A. Rosenberg, Y. Zhang, G. Wang, B. Ramabhadran, and P. J. Moreno, "Improving speech recognition using gan-based speech synthesis and contrastive unspoken text selection.," in *Proc. Interspeech*, 2020.

[110] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *Proc. ICASSP*, 2021.

[111] A. Werchniak, R. B. Chicote, Y. Mishchenko, J. Droppo, J. Condal, P. Liu, and A. Shah, "Exploring the application of synthetic audio in training keyword spotters," in *Proc. ICASSP*, 2021.

[112] J. Lin, K. Kilgour, D. Roblek, and M. Sharifi, "Training keyword spotters with limited and synthesized speech data," in *Proc. ICASSP*, 2020.

[113] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proc. CVPR*, 2017.

[114] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[115] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proc. CVPR*, 2020.

[116] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014.

[117] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015.

[118] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*, 2015.

[119] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[120] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018.

[121] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[122] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Proc. Interspeech*, 2019.

[123] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *ArXiv e-prints*, 2018.

[124] A. Shrivastava, A. Kundu, C. Dhir, D. Naik, and O. Tuzel, "Optimize what matters: Training dnn-hmm keyword spotting model using end metric," in *Proc. ICASSP*, 2021.

[125] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang, "Drum: End-to-end differentiable rule mining on knowledge graphs," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[126] M. Armandpour, B. Kidd, Y. Du, and J. Z. Huang, "Deep personalized glucose level forecasting using attention-based recurrent neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2021.

[127] J. Li and M. Armandpour, "Deep spatio-temporal wind power forecasting," *ICASSP*, 2022.

[128] M. Armandpour, P. Ding, J. Huang, and X. Hu, "Robust negative sampling for network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3191–3198, 2019.

[129] A. Hasanzadeh, M. Armandpour, E. Hajiramezanali, M. Zhou, N. Duffield, and K. Narayanan, "Bayesian graph contrastive learning," *arXiv preprint arXiv:2112.07823*, 2021.

[130] M. Armandpour, A. Sadeghian, and M. Zhou, "Convex polytope trees and its application to vae," *Advances in Neural Information Processing Systems*, vol. 34, 2021.