Data Sharing with Generative Adversarial Networks: From Theory to Practice

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Zinan Lin

B.E., Electronic Engineering, Tsinghua University

Carnegie Mellon University

Pittsburgh, PA

October 2022

Acknowledgments

First of all, I would like to thank my co-advisors, Giulia Fanti and Vyas Sekar. Their incredible support have created a fantastic environment for me to freely explore research directions and ideas, and to fully enjoy the excitement of research. Their endless patience and insightful guidance have helped me to grow significantly in all aspects of my research skills. I have always found Giulia to be very approachable and enjoyable person to talk with, both as an advisor and as a friend. Her high research standards, rigorous research attitude, and caring personality have had a profound impact to me. Similarly, I have always been impressed by Vyas's sharp mind when it comes to making connections across different topics and his brilliant vision for impactful research problems. While I have learned a lot from Vyas's writing, presentation, and communication skills, I still have a long way to go to reach even a fraction of his abilities in these areas. In retrospect, I feel extremely fortunate to have had Giulia and Vyas as my co-advisors during my research journey.

I would like to express my heartfelt gratitude to my dissertation committee members, Sewoong Oh, Jun-Yan Zhu, and Steven Wu. Sewoong Oh introduced me to this fascinating world of generative models, and provided invaluable guidance and mentorship as I embarked on my earliest research projects in Ph.D. His brilliant ideas and mathematical insights have shaped my research path to a great extent. I am also extremely grateful to have Jun-Yan and Steven in my committee. I have long been a fan of their work, particularly Jun-Yan's pioneering work on generative adversarial networks and Steven's seminal work on privacy, which have inspired me greatly throughout my Ph.D. journey. Their insightful comments and feedback have been invaluable to the writing of this dissertation.

I would also like to extend my thanks to all my wonderful collaborators, including Alankar Jain, Ashish Khetan, Carolina Zarate, Charles Kamhoua, Chen Wang, Cho-Yu Jason Chiang, Hao Liang, Jeremy Cohen, Kevin Chan, Kiran Thekumparampil, Minhao Jin, Nandi Leslie, Ritika Mulagalapalli, Sekar Kulandaivel, Shuaiqi Wang, Soo-Jin Moon,

Todd Huster, and Yucheng Yin. Without your help and contributions, this dissertation would not have been possible. I would also like to thank my mentors and colleagues during my research internships at Google and NVIDIA, including Yundi Qian, Xun Huang, Eugene Brevdo, Mircea Trofin, Krzysztof Choromanski, David Li, and Ming-Yu Liu, who have taught me so much beyond the scope of my research. In addition, I would also like to express my appreciation to all of the professors, fellow students, researchers, classmates, friends, and staff of ECE, CyLab, and CMU, who have made my last five years so memorable. Your help, support, and friendship have meant a great deal to me.

Finally, I would like to express my deepest gratitude to my wife Wenyu Wang and my parents Congren Lin and Ruiying Chen for their unconditional love and support throughout my life. I would also like to thank my two lovely cats, Cookie and Doudou, for their "warm" support and companion.

Funding sources. My Ph.D. has been supported by 2022 AAAI scholarship, 2020 CyLab Presidential Fellowship, 2019 Siemens FutureMakers Fellowship, 2017 CMU Presidential Fellowship, and 2017 Carnegie Institute of Technology Dean's Fellow. The dissertation is supported by NSF awards CNS-1527754, CCF-1553452, CCF-1705007, RI1815535, CA-2040675, and RINGS award 2148359. The dissertation is supported in part by faculty research awards from Google, JP Morgan Chase, Intel, and the Sloan Foundation, as well as gift grants from Cisco and Siemens AG. This research was sponsored in part by the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA), the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W911NF20D0002, and the Air Force Office of Scientific Research under award number FA9550-21-1-0090. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and

distribute reprints for Government purposes notwithstanding any copyright notation here on. This dissertation used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575 and ACI-1548562. Specifically, it used the Bridges system, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC). This dissertation is partially supported by the generous research credits on AWS cloud computing resources from Amazon.

Zinan Lin

Abstract

In today's age of big data, data sharing among companies, customers, and researchers has become a critical activity that drives advancements across industry and academia. In these data sharing scenarios, stakeholders want the shared data to have *high fidelity*, meaning that it accurately reflects the important properties of the original data for downstream applications. At the same time, the shared data must be *privacy-preserving*, so that sensitive business and personal information from the data holder is not disclosed during the data sharing process. Unfortunately, achieving both of these goals simultaneously is challenging with existing data sharing techniques such as anonymization, simulation, or simple generative models.

Recent advances in generative adversarial networks (GANs) offer a new opportunity to tackle this long-standing challenge. Given a dataset of images, GANs can synthesize new, random images that are from the same distribution as the original images. Their impressive results in synthesizing photorealistic, high-resolution images suggest the potential of GANs as a building block for a data sharing tool. However, notable challenges remain. On the fidelity front, GANs' generated samples often lack diversity, and GANs are notoriously unstable to train—small changes to hyper-parameters can lead to poor sample fidelity. Moreover, real-world data required in data sharing applications (e.g., long and multi-dimensional time series) has its own unique characteristics different from images, which creates additional fidelity challenges. On the privacy front, the privacy properties of GANs are not well understood, and making them privacy-preserving is still an open question.

In this dissertation, we explore how to *build a high-fidelity and privacy-preserving data sharing tool with GANs*. We tackle this question in a full-stack fashion, from studying and improving the theoretical foundations of GANs to applying these insights in practical data sharing applications. On the fidelity front, we propose theoretical frameworks for

analyzing GAN's sample diversity and training stability problems. Based on these insights, we propose simple and effective fixes to boost GANs' sample diversity and training stability, resulting in better sample fidelity. On the privacy front, we analyze the fundamental privacy properties of GANs, identify the privacy issues, and design new frameworks and approaches to protect sensitive business and personal information in the original data. Finally, based on these insights, we build a practical GAN-based data sharing tool for time series data and demonstrate its fidelity across applications in systems and networking domains. We also package the algorithmic contributions in this dissertation in a modular library for future applications.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

We live in a data-driven world. Data sharing between different parties has become increasingly common in many aspects of industry and academia (Fig. 1.1). For example, network traces are shared from customers to networking vendors, which enables vendors to debug and improve their products [6, 7]. Medical data is shared between hospitals [8, 9], which enables them to collaboratively develop new machine-learning-based diagnosis algorithms [10]. Data shared by researchers allow their research to be reproducible by others [11, 12]. In fact, data sharing has even become a new business in itself, with the emergence of data marketplaces such as Databricks and Snowflake.

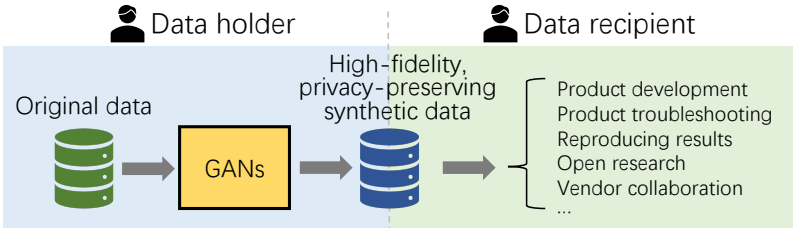


Figure 1.1: Data sharing is crucial in many fields and applications. This dissertation presents fundamental and practical innovations in building a high-fidelity, privacy-preserving data sharing tool with generative adversarial networks (GANs) [1]. This tool allows data holders to share proprietary data without compromising its sensitive information.

In these data sharing scenarios, stakeholders want the shared data to be both *high-fidelity* and *privacy-preserving*. Although its precise definition depends on the application, *fidelity* means how much a piece of shared data preserves the characteristics of the original data. For example, network vendors want the network traces shared from customers to be representative of the true diversity in the original data (in terms of time-space, network protocols, etc.), so that they can use them to debug and improve their products realistically.

Hospitals, on the other hand, want the shared data to be good enough so that diagnostic algorithms trained on it will have high accuracy. *Privacy* refers to the ability of the shared data to hide the information that data owners do not want to be revealed. In practice, data owners have many privacy concerns. For example, when customers share their network traces with the vendors, they may worry that the vendor can infer their business scales from the total amount of traffic in the shared network traces. While there is a good motivation for hospitals to share medical data, they do not want data recipients to be able to reconstruct or infer information about a specific patient (e.g., name, address, disease). Indeed, data protection regulations such as GDPR [13] and HIPAA [14] impose strict rules on how data should be handled and shared to protect privacy.

Unfortunately, *high-fidelity* and *privacy-preserving* data sharing has been a challenge for decades. One extreme solution is to share raw data, which has the highest fidelity but provides no privacy. Instead of sharing raw data, many approaches have been proposed that generate *a different copy* of data to share. *Anonymization* [15] is one common approach, which involves removing or replacing personal identifiable information such as names and zip codes. However, this approach can hurt *fidelity* by making it impossible to make any inferences related to the removed attributes (e.g., the prevalence of a medical condition within a specific zip code). Additionally, anonymization can still have severe *privacy* issues, as the removed attributes can sometimes be recovered from other attributes, as seen in the Netflix data breaches [16]. Another approach that is gaining popularity is called *synthetic data*, which involves generating *a new copy* of data to share instead of directly modifying the original data. One method for generating synthetic data is to use *simulators* to simulate the process of generating ground-truth data and collect such data from the simulation [17, 18, 19, 20, 21, 22, 23]. For example, a network simulator could be used to collect network packets by simulating real servers talking to each other [19]. However, it can be difficult to configure the parameters of the simulation (e.g., network link, queue size) realistically, which can result in *bad fidelity*. Another approach is to use *machine learning models* whose parameters are learned from data [24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 18, 37, 38, 39, 40]. For example, we could compute the mean and standard deviation of the ages of a group of people and release data sampled from a Gaussian distribution with those parameters. Of course, a Gaussian distribution may not be a good representation of the original data. Indeed, this is a common problem with many machine learning models: their representation capabilities may not be sufficient to accurately describe the complicated real-world data, resulting in *bad fidelity*.

1.2 The Promise and Challenges of GANs

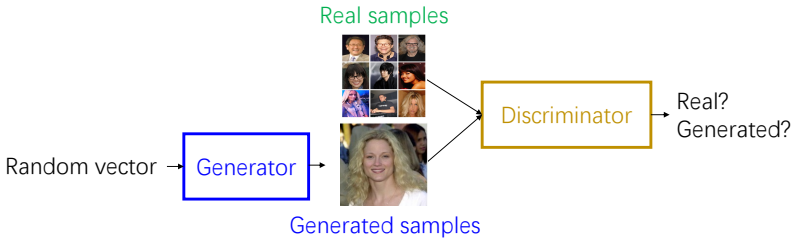


Figure 1.2: GANs [1] consist of two components: a generator, which maps a random vector to a random sample, and a discriminator, which tries to distinguish between real and generated data samples. The images are from CelebA Dataset [2] for illustration purposes.

In 2014, a new machine learning model called Generative Adversarial Networks (GANs) [1] was invented, which offers a new opportunity to tackle the long-standing challenge of generating high-fidelity and privacy-preserving synthetic data. GANs are able to use samples to learn the underlying distribution of a dataset and then generate new samples that follow that distribution. Data holders can then release the generated samples to other parties. GANs consist of two neural networks: a generator and a discriminator (Fig. 1.2). The generator tries to generate samples that the discriminator will classify as being from the real dataset. The discriminator, on the other hand, tries to accurately classify whether the input sample is from the real dataset or from the generator. The two networks are trained alternatively, and it has been shown that given enough data, network capacity, and training time, the generator can learn the real distribution and generate samples that mimic the properties of real data [1]. GANs have generated a lot of excitement in the machine learning community due to their ability to generate high-quality and high-resolution images [41, 42], which was a very challenging task for prior machine learning models. This suggests that GANs have the potential to overcome the long-standing fidelity problems of prior data sharing tools. Motivated by this, the overarching questions we ask in this dissertation is (Fig. 1.1):

Can we build a high-fidelity and privacy-preserving data sharing tool with GANs?

Despite the promise, some challenges remain. (1) *Fidelity.* Although GANs already outperform prior models on fidelity by a great extent, they still have fidelity problems. Indeed, GANs may not always produce samples with good fidelity—slight changes in the

hyper-parameters of the model can lead to poor generated samples [43, 44]. In addition, the diversity of generated samples may be worse than that of the real data [44, 45]. (2) *Privacy.* As GANs are a new model, their privacy properties remain unclear. Understanding the privacy properties of GANs and making GANs more privacy-preserving regarding the properties that data holders care about are important tasks. (3) *Application.* GANs are initially successful in images, but real-world data comes in many other forms. It is an open question how to adapt GANs to work with these more complicated forms of data.

1.3 Contributions

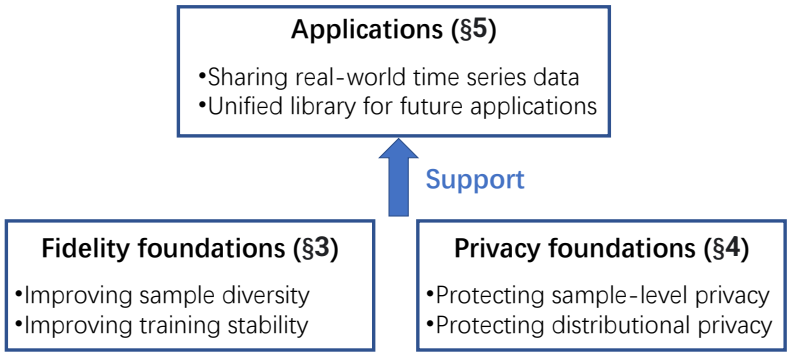


Figure 1.3: The contributions of this dissertation.

In this dissertation, we tackle these questions in a full-stack fashion: ranging from studying the theoretical foundations of GANs' *fidelity* and *privacy* properties, to designing general algorithms for improving GANs' *fidelity* and *privacy*, as well as applying these insights in real-world data sharing applications. Our specific contributions are as follows (Fig. 1.3).

Contribution 1: Fidelity foundations. We conduct theoretical analysis on the cause of the fidelity issues in GANs, and we propose principled and practical solutions to mitigate these issues. More specifically, we study *mode collapse* and *training instability*, the two biggest fidelity issues in GANs [44].

Mode collapse refers to the phenomenon where generated samples only capture a part of the modes in data [44]. For example, when training a GAN on images of handwritten digits 0 to 9, the GAN may only be able to generate some of the digits [46, 44]. We propose the first analytical definition of mode collapse and theoretical analysis of the problem, using

tools from binary hypothesis testing and the seminal results of Blackwell [47]. This provides an important theoretical foundation for understanding and analyzing mode collapse. Based on this analysis, we propose PacGAN, a principled approach for handling mode collapse. Instead of taking either one real sample or one generator sample at a time, PacGAN's discriminator takes a group of m samples, jointly coming from either real data or the generator. We prove why this simple change could mitigate mode collapse—the intuition is that multiple samples make it easier for the discriminator to detect mode collapse and penalize the generator for it. PacGAN is easier to implement than previous approaches and has significant improvements over vanilla GANs in practice. For the first time, PacGAN achieves a full score on Stacked MNIST Dataset [45], a benchmark for mode collapse.

Training stability refers to the phenomenon where the sample fidelity of GANs often fluctuates a lot during training and has large variance across different hyper-parameters [43, 44]. This makes it difficult to get the best fidelity from GANs. A widely used heuristic for stabilizing GAN training is spectral normalization (SN) [43], but little is understood about why it works, so there is little analytical basis for using it or improving it. In this dissertation, we prove that spectral normalization is able to ensure that the gradients of GANs are always in a suitable range that prevents training instability. Our theory also explains why an "incorrect" implementation of SN [43] can result in much better training stability than an correct implementation [48]: the "incorrect" implementation *happens* to control the gradient scales in the right range, while the correct implementation does not. Based on these theoretical insights, we propose *Bidirectional Scaled Spectral Normalization (BSSN)*, an improved version of SN that better conditions the gradients and therefore better stabilizes the training. By simply replacing SN with BSSN, we see better training stability and better sample fidelity across a wide range of datasets.

Contribution 2: Privacy foundations. We conduct theoretical analysis of the privacy properties of GANs, and propose algorithms to make them more privacy-preserving. More specifically, we study two primary privacy concerns that data holders have: leakage of information about *individual samples* and about the *underlying distribution*.

Information about *individual samples* (e.g., the name and address of a patient, the IP address of a client) is highly sensitive. The de facto privacy framework for this privacy concern is *differential privacy* (DP) [3]. DP uses a pair of numbers (ϵ, δ) to describe a theoretical upper bound on the leakage of individual information. We study the inherent DP properties of GANs and show that vanilla GANs do have a DP guarantee, but the guarantee is too weak to provide meaningful practical protection. This means that we

need to modify GANs to make them differentially private. One natural approach is to apply DP-SGD [49], a standard method for making neural networks differentially private. But we empirically show that this approach hurts data fidelity too much. To address this, we propose pretraining GANs on a public dataset and fine-tuning them on the sensitive, private dataset with DP-SGD. This approach improves the privacy-fidelity tradeoff.

The other concern is about the *underlying distribution* of data. For example, the *mean* traffic volume of data from a networking company can reveal the scale of its business, which may be considered a trade secret. Unlike individual privacy concerns, these distributional privacy concerns have been less studied in the community. To address this, we propose *distributional privacy*, a principled framework for analyzing and protecting these distributional privacy concerns in data sharing scenarios. We conduct theoretical analysis of the privacy-fidelity tradeoff under distributional privacy, and propose data release mechanisms for protecting distributional privacy concerns such as mean, quantile, and standard deviation. These data release mechanisms can be applied as a pre-processing or post-processing step in our GAN-based data sharing framework.

Contribution 3: Applications. We want to apply our insights on concrete data sharing applications to make real-world impacts. We focus on time series data, a common type of data in many domains including system usage logs in cloud clusters, packets in computer networks, bank transactions, and health records. Time series data has unique characteristics different to images, such as complicated temporal patterns and complicated correlations between different dimensions. We design new GAN architectures and data normalization techniques tailored to time series data and we show that our synthetic data maintains better fidelity than prior data sharing techniques in real-world applications.

We also package all of our fundamental fidelity and privacy advances, as well as our time series data algorithm, in a Python package [6]. The package is modular and designed to be extensible for supporting future data sharing techniques and applications.

Broader impacts. Our work has already been adopted by several companies in their data sharing products and applications (see their blogs [50, 51, 52]). We hope that our work can help unleash the potential of data through our high-fidelity, privacy-preserving data sharing tools.

Our work could be useful even beyond data sharing. Note that GANs have been widely used in other applications, such as image editing [53], virtual reality [54], content creation [55]. All of these applications are impacted by the fundamental fidelity and privacy challenges in GANs. Our fidelity and privacy techniques can be generally useful for

mitigating these challenges regardless of the application.

1.4 Outline

The reset of the dissertation is structured as follows. Chapter 2 discusses the background and preliminaries. Chapter 3 covers our work on understanding and improving the *fidelity* foundations of GANs. Chapter 4 focuses on our work on understanding and improving the *privacy* foundations of GANs. Chapter 5 presents how we apply our insights in real-world data sharing applications. Chapter 6 provides the conclusion, summarizes other work not included in the dissertation, and discusses future work.

Chapter 2

Background

2.1 Motivating Scenarios

In today's era of big data, data sharing between different parties is increasingly important across many aspects of industry and academia (Fig. 1.1). For example, data sharing is crucial for:

- **Collaboration across stakeholders in industry.** Data sharing between different organizations is an important part of the product development process. For instance, video session data shared between video websites and video analytics vendors enables both to collaboratively improve the video viewing experience for end users [56, 57]. *High fidelity* data is essential to ensure the accuracy of the analytics results. However, *sensitive user information* may be recovered from video viewing data [16]. In another example, medical data shared between hospitals and healthcare service providers (e.g., Nuance [58]) can enable the development of new ML-based solutions that improve the efficiency of doctors and the experience of patients. In this scenario, the healthcare service provider needs *high fidelity* medical data to train better ML models, but hospitals may be concerned about the potential *leakage of patient information*.

- **Reproducible, open research.** Many research proposals rely on datasets to test and develop ideas, and larger datasets often result in breakthroughs in technology. For example, in the fields of computer vision and machine learning, ImageNet [11], a large scale real-world image database, has facilitated the evolution of deep learning algorithms (e.g., [59, 60]). In networking and systems, large-scale datasets like the cloud cluster traces datasets released by Google [15], Microsoft [61], and Alibaba [62]) have also enabled researchers to study job scheduling algorithms, cloud simulators, and data center systems [63] that would otherwise be difficult to study. In these scenarios, researchers need *high fidelity* data to ensure that the research results

are realistic. However, releasing these datasets can also inadvertently *leak sensitive information*. For example, cloud cluster trace datasets can reveal strategic enterprise choices, such as the fraction of server types in use [12]. Such information reflects the company's business strategy and should be kept secret from competitors and vendors. Simply removing the server type from the dataset is not a good option, as this information is important for the downstream applications of the dataset, such as predicting future CPU and memory usage.

Therefore, having a data sharing technique that can share *high-fidelity* and *privacy-preserving* data is essential for fully harnessing the power of data. In the next section, we will review existing techniques on data sharing.

2.2 Techniques for Sharing Data

To address the privacy concerns, a common approach is to generate *a different copy* of data to share. Specifically, given a set of N samples $\mathcal{X} = \{x_1, \ldots, x_N\}$, the goal is to generate a new set of samples $\mathcal{Y} = \{y_1, \ldots, y_M\}$ that preserve the key characteristics of the original data that data users care about (i.e. maintaining good fidelity), while hiding the sensitive information that data owners do not want to leak (i.e., ensuring good privacy). The taxonomy of the techniques for generating such data is visualized in Fig. 2.1.

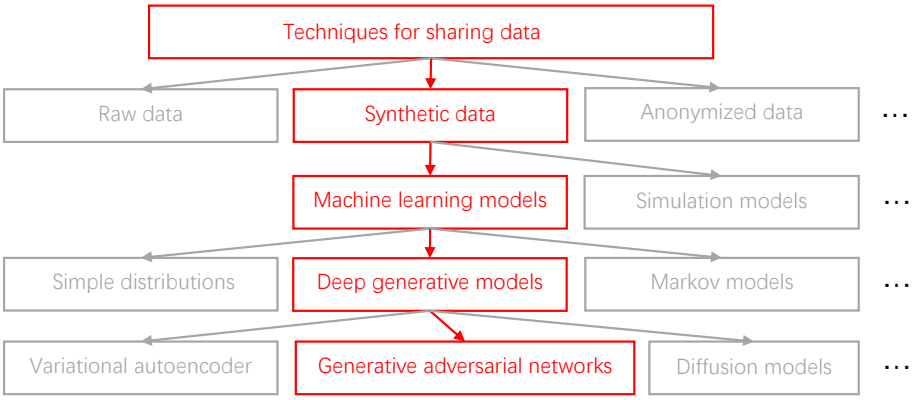


Figure 2.1: Taxonomy of data sharing techniques. Arrows point to subcategories of the technique. Red boxes indicate the techniques that we focus on in this dissertation.

In this section, we discuss related techniques, as well as generative adversarial networks, the focus of this dissertation. These discussions are *not* intended to be a compre-

hensive review of all these techniques, but rather to help readers understand the position of the dissertation within the broader field.

2.2.1 Prior Techniques

Instead of sharing raw data, there are two main alternatives used in practice. One is sharing *anonymized data* [4]. In this approach, there is one-one mapping between the generated sample and raw sample (thus we have $M = N$). Each anonymized sample y_i is created from the raw sample x_i by removing personal identifiable information (e.g., name and address in medical data, IP address in network traces) or replacing it with random strings. This approach is widely used in the release process of public datasets (e.g., [15]) but has drawbacks in terms of both *fidelity* and *privacy*. This approach could greatly hurt the resulting data fidelity due to the anonymization process [64]. At the same time, it does not guarantee the privacy of data. Famous examples of this include Netflix data breaches [16], where attackers were able to recover the identities of individuals in the anonymized movie rating database released by Netflix [16]. The key insight is that the identity of the users in the data can still be inferred from other seemingly non-sensitive attributes like movie ratings when the attackers have side information. The other approach is *synthetic data*, which we will discuss next.

Synthetic data is a *different* set of samples that mimic properties of the raw data. The number of generated samples M is not necessarily equal to the number of original samples N. There are two main models for creating synthetic data. *Simulation models* generate data by building a simulator that mimics a real system that generates the raw data [17, 18, 19, 20, 21, 22, 23]. For example, ns-2 [19] is a widely used simulator for computer networks. Data holders can configure ns-2 according to the data and network, and run the simulators to collect data (i.e., network packets) for sharing. In terms of fidelity, simulation models are good if the simulator is very close to real systems. However, in practice, it is often difficult to configure the parameters to simulate a given target dataset. For example, given a dataset of packet traces, it is difficult to infer the speed of a specific network link, considering the complex behaviors from the application layers all the way down to the physical layers. As a result, these models could have poor *fidelity* in practice. The other class of model used to generate synthetic data is machine learning models, which we discuss next.

Machine learning models are *parametric models* where the parameters can be learned

(trained) from data. Specifically, suppose that the original data samples $x_1, \ldots, x_N \in \mathbb{R}^d$ are from an unknown probability distribution p_X over $\mathbb{R}^d$. These models specify a parametric distribution q_θ to approximate p_X, and use the data samples to compute the optimal parameters (e.g., by maximizing the data likelihood [65]: $\max_\theta \sum_{i=1}^{N} \log q_\theta(x_i)$). Then, new samples can be drawn from this learned model and released to other parties.

These models can be roughly categorized by the amount of domain expertise required in the design phase. Some models in this class require a significant amount of human expertise: domain experts manually examine the data to determine which parameters are important and which parametric model to use [24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 18, 37, 38, 39, 40]. Let's take the networking domain as an example again for comparison to anonymization. Swing [36] extracts statistics of user, application, flow, and link (e.g., packet loss rate, inter-session times) from raw network traffic data, and then generates traffic by sampling from the extracted distributions. The *fidelity* of these models depends on how well the model describes the data. In practice, it can be challenging to come up with a complete list of characteristics that accurately describe the real-world. For instance, BURSE [29] explicitly models the burstiness and self-similarity in cloud computing workloads, but does not consider e.g. nonstationary and long-term correlations [66].

Other more general parametric models require less human expertise to configure. Examples include Markov models for time series data [67] and restricted Boltzmann machine for images [68]. These models have fewer hyper-parameters that users need to configure and are able to generalize across more diverse datasets, thus requiring less human expertise to use. However, due to their limited modeling capability, they usually have poor fidelity on complicated real world datasets.

In the 2010s, during other such breakthroughs in deep learning, a new class of machine learning models called deep generative models emerged as a promising approach, which requires even less human expertise and has high *fidelity* for complex data. We discuss it in the next paragraph.

Deep generative models are based on deep neural networks. The key component of many deep generative models is a neural-network generator $G_\theta : \mathbb{R}^l \to \mathbb{R}^d$ that maps a lower-dimensional random noise $z \in \mathbb{R}^l$ from a standard distribution p_Z (e.g., spherical Gaussian) to the data domain $\mathbb{R}^d$. In other words, the parametric distribution q_θ used to approximate the data distribution p_X is induced by $G_\theta(z)$ where $z \sim p_Z$.

With the representation power of deep neural networks, this class model can better

represent completed data than the other models discussed previously. However, the fact that it is less constrained also makes it harder to train. For example, unlike Markov models where q_θ is analytically tractable, there is no closed form solution for q_θ in deep generative models if G_θ is an arbitrary neural network. As a result, we cannot train it directly with the objectives that work for other models, such as the maximum data likelihood loss $\max_\theta \sum_{i=1}^{N} \log q_\theta(x_i)$ that we discussed earlier.

Starting in the 2010s, many innovative approaches were proposed for training deep generative models effectively. These approaches can be roughly divided into three categories: (1) adding constraints to make q_θ tractable [69, 70], (2) using approximations of the data likelihood [71], or (3) using a completely different loss [1]. As an example of the first category, flow models [69, 70] constraint G_θ to be invertible so that $q_\theta(x)$ can be computed efficiently from θ and x. As an example of the second category, variational autoencoder (VAE) [71] optimizes a lower bound of data likelihood instead of optimizing the intractable data likelihood directly. The lower bound is derived using variational methods (see [71] for details). Diffusion models [72, 73, 74] are a combination of the first two categories. Like VAE, they also optimize a lower bound of data likelihood, but through a different method: instead of directly generating the samples though a single neural network pass of G_θ, diffusion models progressively transform Gaussian noise to the real distribution through a sequence of denoising steps. Each denoising step is constrained to be a Gaussian distribution (whose parameters are predicted by the neural network) so that the lower bound is tractable. Generative adversarial networks (GANs) [1] belong to the third category: it utilizes completely different training losses and paradigms.

Among all these models, GANs were the first one to be able to synthesize high-quality, high-resolution images with dimensions as large as 1024×1024 pixels [41]. As a result, GANs have attracted significant attention in the machine learning and computer vision communities in recent years as a mainstream technique for data synthesis. We discuss its technical details next.

2.2.2 Generative adversarial networks (GANs)

GANs are an innovative idea of training deep generative models. Besides the generator G_θ, GANs have another component called *discriminator* D_η (Fig. 1.2). These two neural networks play a dynamic minimax game against each other. An analogy provides the intuition behind this idea. The generator is acting as a forger trying to make fake coins

(i.e., samples), and the discriminator is trying to detect which coins are fake and which are real. If these two parties are allowed to play against each other long enough, both will become good at their tasks. In particular, the generator will learn to produce coins that are indistinguishable from real coins (but preferably different from the training coins he was given).

Concretely, we search for the parameters of G_θ and D_η (i.e., θ, η) that optimize the following type of minimax objective:

$$\theta^* \in \arg\min_\theta \underbrace{\max_\eta \; \mathbb{E}_{x\sim p_X}\left[\log\left(D_\eta\left(x\right)\right)\right] + \mathbb{E}_{z\sim p_Z}\left[\log\left(1 - D_\eta\left(G_\theta\left(z\right)\right)\right)\right]}_{2\cdot d_{\mathrm{JS}}(p_X \| q_\theta) - \log(4)} . \tag{2.1}$$

where p_X is the distribution of the real data, and p_Z is the distribution of the input code vector z. Critically, [75] shows that the global optimum of Eq. (2.1) is achieved if and only if $p_X = q_\theta$, where q_θ is the generated distribution of $G_\theta\left(z\right)$. The reason is that the optimal value of the inner maximization problem turns out to be the Jensen-Shannon divergence (denoted by $d_{\mathrm{JS}}\left(\cdot\|\cdot\right)$) between the data distribution p_X and the generated distribution q_θ (up to a scaling and shift), when the discriminator D_η is trained to the optimum over the space of all functions. In practice, we search over some parametric family of discriminators, and we can only compute sample average of the losses. This provides an approximation of the Jensen-Shannon divergence between p_X and q_θ. The outer minimization over the generator tries to generate samples such that they are close to the real data in this (approximate) Jensen-Shannon divergence. Therefore, q_θ is pushed to be closer to the target p_X. In practice, we approximate the solution to the minimax problem Eq. (2.1) by iteratively training G_θ and D_η [1]. Each model can be updated individually by backpropagating the gradient of the loss function to each model's parameters.

Since their invention in 2014, GANs have attracted a great deal of interest due to their ability to *generate* realistic, crisp, and original examples of images [75, 76] and text [77]. This has made them useful in various image and video processing tasks, such as frame prediction [78], image super-resolution [79], and image-to-image translation [80]. GANs have also been applied in dialogue systems or chatbots, where artificially generated but realistic data is needed. In addition, GANs implicitly learn a *latent, low-dimensional representation* of arbitrary high-dimensional data. Such embeddings have been hugely successful in the area of natural language processing (e.g. word2vec [81]). GANs have the potential to provide an unsupervised solution for learning representations that capture

semantics of various arbitrary data structures and applications. This can be used in a range of applications, such as image manipulation [82] and defending against adversarial examples [83].

The promise of GANs offers us a new opportunity for solving the data sharing problem. This motives us to ask:

Can we build a high-fidelity and privacy-preserving data sharing tool with GANs?

While we do not claim that GANs are the best solution for data sharing, GANs show promise in addressing the long-standing fidelity problems. The dissertation focuses on exploring how we can leverage the capabilities of GANs to build a more effective data sharing tool, so as to unlock the full potential of data in industry and academia.

2.3 Open Questions

Despite the promise of generative adversarial networks (GANs), there are still some challenges in realizing the goal.

Fidelity. Although GANs have achieved state-of-the-art sample fidelity in image domains, there are still some fidelity problems. The two biggest and well-known fidelity issues are *mode collapse* and *training stability* [44]. Mode collapse refers to the problem where the diversity of the generated samples is worse than that of the real samples. Training stability refers to the phenomenon during which small changes in hyper-parameter or randomness in optimization can cause training to fail altogether, leading to poor samples being generated. These problems are believed to stem from the fundamental training algorithms of GANs. In Chapter 3, we will discuss our theoretical analysis of these problems and the practical algorithms for mitigating them.

Privacy. As GANs are a relatively new technology, their privacy properties are not yet well understood. For example, how well do GANs protect the sensitive information in original data, such as user information or business secrets? If GANs are not privacy-preserving by default, can they be improved to better protect privacy? In Chapter 4, we will discuss our theoretical analysis of these questions and our mitigation approaches for making GANs more privacy-preserving.

Applications. GANs were initially designed for and have been most successful in image domains. However, real-world data is much more diverse and complex. Adjusting GANs to those real-world data sharing applications is an open question. In Chapter 5, we will

discuss how to apply GANs to time series data in various domains such as networking and systems. Then we will conclude by discussing the design and implementation of a unified library, containing all the fundamental and application advances in this dissertation for future applications.

Chapter 3

Fidelity Foundations

In this chapter, we first give an overview on the fundamental fidelity issues of GANs (§ 3.1). We then give our theoretical analysis and algorithmic solutions for the two biggest fidelity issues of GANs: mode collapse (§ 3.2) and training instability (§ 3.3). Finally, we conclude this chapter in § 3.4.

3.1 Overview of Fidelity Challenges

Despite their significant improvement in fidelity compared to prior data sharing tools (as discussed in § 2.2), GANs still suffer from some fidelity issues. The two biggest issues acknowledged in the community [44] are *mode collapse* and *training instability*.

Mode collapse refers to the lack of diversity in generated samples. A common manifestation of mode collapse is the observation that GANs commonly miss some of the modes when trained on multimodal distributions. In Fig. 3.1, we train GANs on a 2D Grid Dataset sampled from a mixture of 25 two-dimensional spherical Gaussians with means $(-4+2i, -4+2j)$ and variances 0.0025 in each dimension for $i, j \in \{0, 1, 2, 3, 4\}$. GANs are only able to generate samples from a subset of modes. Another example is shown in Fig. 3.2, where we train GANs on a Stacked MNIST Dataset in which each image consists of three randomly-selected MNIST images [84] that are stacked into a $28 \times 28 \times 3$ image in RGB. Again, we see that GANs produce many similar and low-quality images. This problem has been widely documented in many GAN applications. For instance, when trained on hand-written digits with ten modes, the generator might fail to produce some of the digits [85]. Similarly, in tasks that translate a caption into an image, generators have been shown to generate series of nearly-identical images [86].

Training instability refers to the phenomenon that the sample fidelity of GANs fluctuates a lot during training, and small changes in hyper-parameters and even randomness in the optimization can cause the training to fail, resulting in poor generated samples. It is believed that this issue is primarily due to the alternating updates of the generator and

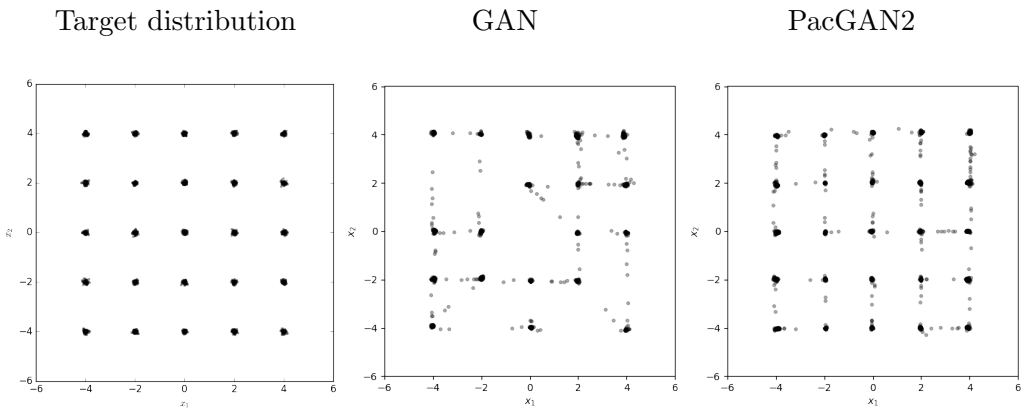


Figure 3.1: Scatter plot of the 2D samples from the true distribution (left) of 2D Grid Dataset and the learned generators using GAN (middle) and PacGAN2 (right). PacGAN2 captures all of the 25 modes.

discriminator in GANs. Such variability makes it challenging to evaluate when training has converged, let alone which model one should choose among those obtained throughout the training process. Empirically, this observation seems to hold even with improved GAN optimization techniques, such as unrolled GANs [87], despite recent work showing that gradient-descent-based optimization of GANs is locally stable [88].

In practice, *mode collapse* and *training instability* sometimes occur simultaneously, or independently of one another [44]. In this dissertation, instead of coupling these two issues together, we study them in isolation in order to gain a deeper understanding of these issues and their causes.

3.2 Improving Sample Diversity

In this section, we first present our theoretical analysis of the mode collapse problem (§ 3.2.1). Based on the theoretical insights, we introduce PacGAN, our practical algorithm for mitigating mode collapse issue (§ 3.2.2), and its experimental results (§ 3.2.3). Finally, we summarize the contributions and discuss future work (§ 3.2.4).

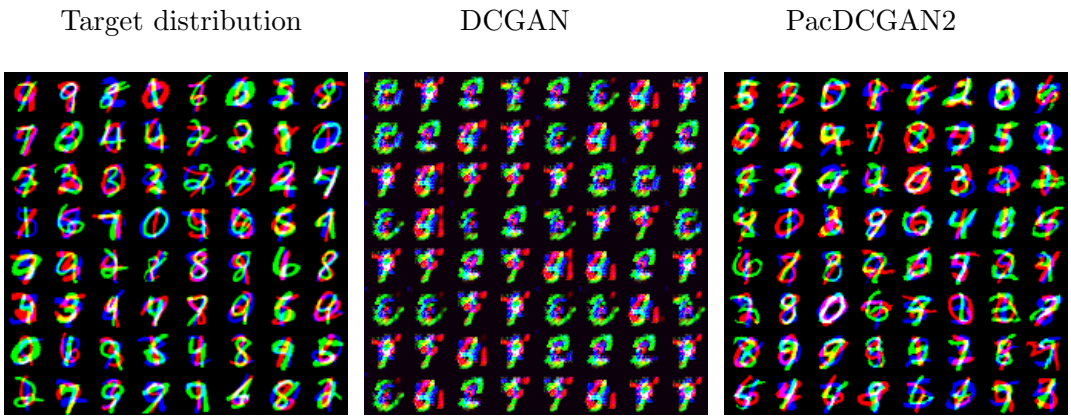


Figure 3.2: True distribution (left), DCGAN generated samples (middle), and PacDCGAN2 generated samples (right) from the Stacked MNIST Dataset show PacDCGAN2 captures more diversity while producing sharper images.

3.2.1 Theoretical Framework for Analyzing Sample Diversity

In this section, we first propose a theoretical framework for understanding mode collapse problem by drawing the connections to hypothesis testing. We then present the theoretical analysis of *packing*, a simple idea to alleviate mode collapse.

Notations and assumptions. For the simplicity of notation, throughout this section, we use P to denote the ground-truth distribution p_X and Q to denote the generated distribution q_θ. We focus on the analysis under the following setting: (a) the optimal discriminator over a family of all measurable functions; (b) the population expectation; and (c) the 0-1 loss function of the form $\max_\eta \mathbb{E}_{x\sim P}\left[\mathbb{I}\left(D_\eta\left(x\right)\right)\right] + \mathbb{E}_{y\sim Q}\left[1-\mathbb{I}\left(D_\eta\left(y\right)\right)\right]$, subject to $D_\eta\left(x\right) \in \{0,1\}$.

This discriminator provides (an approximation of) the total variation distance, and the generator tries to minimize the total variation distance $d_{\mathrm{TV}}\left(P\|Q\right)$, as widely known in the GAN literature [75]. The reason we make this assumption is primarily for clarity and analytical tractability: total variation distance highlights the effect of packing in a way that is cleaner and easier to understand than if we were to analyze Jensen-Shannon divergence.

Mode Collapse Definition

We first provide a formal mathematical definition of mode collapse, which leads to a two-dimensional representation of any pair of distributions (P, Q) as a *mode-collapse region*. This region representation provides not only conceptual clarity regarding mode collapse, but also proof techniques that are essential to proving our main results.

Definition 3.2.1.1. *A target distribution P and a generator Q exhibit* (ε, δ)-mode collapse *for $0 \leq \varepsilon < \delta \leq 1$ if there exists a set $S \subseteq \mathcal{X}$ such that $P(S) \geq \delta$ and $Q(S) \leq \varepsilon$.*

Intuitively, larger δ and smaller ε indicate more severe mode collapse. That is, if a large portion of the target $P(S) \geq \delta$ in some set S in the domain $\mathcal{X}$ is missing in the generator $Q(S) \leq \varepsilon$, we declare (ε, δ)-mode collapse. Similarly, when we change the role of P and Q, and have $P(S) \leq \varepsilon$ and $Q(S) \geq \delta$, we say P and Q exhibit (ϵ, δ)-*mode augmentation*.

A key observation is that *two pairs of distributions can have the same total variation distance while exhibiting very different mode collapse patterns.* To see this, consider a toy example in Fig. 3.3, with a uniform target distribution $P = U([0, 1])$ and a mode collapsing generator $Q_1 = U([0.2, 1])$ and a non mode collapsing generator $Q_2 = 0.6U([0, 0.5]) + 1.4U([0.5, 1])$.

The appropriate way to precisely represent mode collapse is to visualize it through a two-dimensional region we call the *mode collapse region*. For a given pair (P, Q), the corresponding mode collapse region $\mathcal{R}(P, Q)$ is defined as the convex hull of the region of points (ε, δ) such that (P, Q) exhibit (ε, δ)-mode collapse, as shown in Fig. 3.3:

$$\mathcal{R}(P, Q) \triangleq \text{conv}\big(\{ (\varepsilon, \delta) \,|\, \delta > \varepsilon \text{ and } (P, Q) \text{ has } (\varepsilon, \delta)\text{-mode collapse} \} \big)$$

.

Connection to hypothesis testing. We show that the proposed mode collapse region is equivalent to a similar notion in binary hypothesis testing. This allows us to bring powerful mathematical tools from this mature area in statistics and information theory—in particular, the *data processing inequalities* originating from the seminal work of Blackwell [47]. We make this connection precise, which gives insights on how to interpret the mode collapse region, and list the properties and techniques which dramatically simplify the proof.

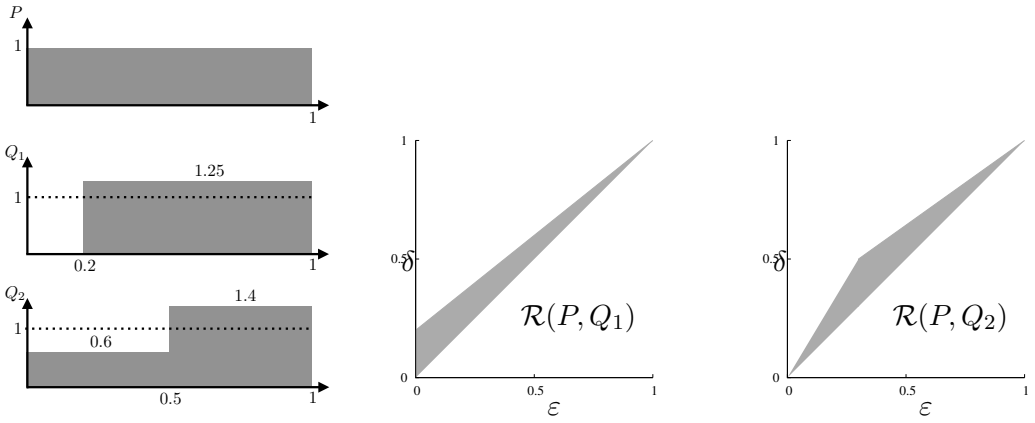


Figure 3.3: A formal definition of (ε, δ)-mode collapse and its accompanying region representation captures the intensity of mode collapse for generators Q_1 with mode collapse and Q_2 which does not have mode collapse, for a toy example distributions P, Q_1, and Q_2 shown on the left. The region of (ε, δ)-mode collapse that is achievable is shown in grey.

More specifically, there is a simple one-to-one correspondence between mode collapse region as we define it in Definition 3.2.1.1 (e.g., Fig. 3.3) and the ROC curve studied in binary hypothesis testing. In the classical testing context, there are two hypotheses, $h = 0$ or $h = 1$, and we make observations via some stochastic experiment in which our observations depend on the hypothesis. Let X denote this observation. One way to visualize such an experiment is using a two-dimensional region defined by the corresponding type I and type II errors. Concretely, an ROC curve of a binary hypothesis testing is obtained by plotting the largest achievable true positive rate (TPR), i.e. 1−probability of missed detection, or equivalently 1− type II error, on the vertical axis against the false positive rate (FPR), i.e probability of false alarm or equivalently type I error, on the horizontal axis.

We can map this binary hypothesis testing setup directly to the GAN context. Suppose the null hypothesis $h = 0$ denotes observations being drawn from the generated distribution Q, and the alternate hypothesis $h = 1$ denotes observations being drawn from the true distribution P. Given a sample X from this experiment, suppose we make a decision on whether the sample came from P or Q based on a rejection region S_{reject}, such that we reject the null hypothesis if $X \in S_{\text{reject}}$. FPR (i.e. Type I error) is when the null hypothesis is true but rejected, which happens with $\mathbb{P}(X \in S_{\text{reject}} | h = 0)$, and TPR (i.e. 1-type II error)

is when the null hypothesis is false but accepted, which happens with $\mathbb{P}(X \in S_{\text{reject}}|h=1)$. Sweeping through the achievable pairs $(\mathbb{P}(X \in S_{\text{reject}}|h=1), \mathbb{P}(X \in S_{\text{reject}}|h=0))$ for all rejection sets, this defines a two-dimensional convex region that we call *hypothesis testing region*. The upper boundary of this convex set is the ROC curve. An example of ROC curves for the two toy examples (P, Q_1) and (P, Q_2) from Fig. 3.3 are shown in Fig. 3.4.

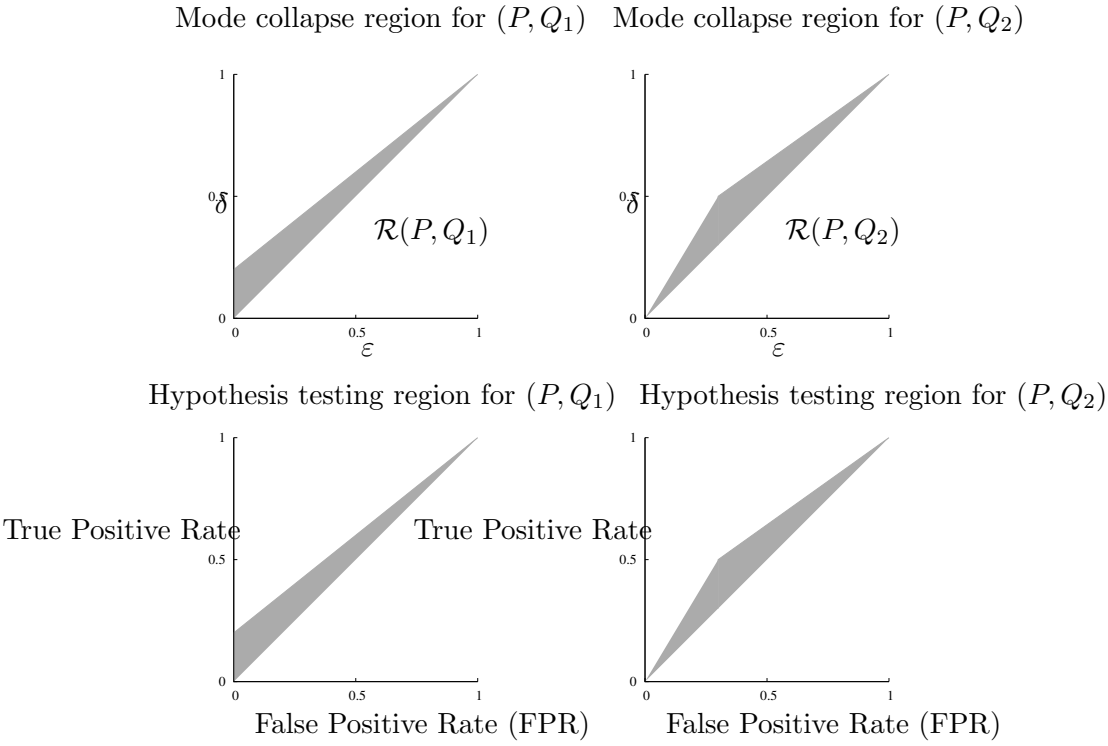


Figure 3.4: The hypothesis testing region of (P, Q) (bottom row) is the same as the mode collapse region (top row). We omit the region above $y = x$ axis in the hypothesis testing region as it is symmetric. The regions for mode collapsing toy example in Fig. 3.3 (P, Q_1) are shown on the left and the regions for the non mode collapsing example (P, Q_2) are shown on the right.

In defining the region, we allow stochastic decisions, such that if a point (x, y) and another point (x', y') are achievable TPR and FPR, then any convex combination of those points are also achievable by randomly choosing between those two rejection sets. Hence, the resulting hypothesis testing region is always a convex set by definition. We also show only the region below the 45-degree line passing through $(0, 0)$ and $(1, 1)$, as the other

region is symmetric and redundant. For a given pair (P, Q), there is a very simple relation between its mode collapse region and hypothesis testing region.

Remark 3.2.1.1 (Equivalence)**.** *For a pair of target P and generator Q, the hypothesis testing region is the same as the mode collapse region.*

This follows immediately from the definition of mode collapse region in Definition 3.2.1.1. If there exists a set S such that $P(S) \geq \delta$ and $Q(S) \leq \varepsilon$, then for the choice of $S_{\text{reject}} = S$ in the binary hypothesis testing, there the point $(\mathbb{P}(X \in S_{\text{reject}}|h = 0) = \varepsilon, \mathbb{P}(X \in S_{\text{reject}}|h = 1) = \delta)$ in the hypothesis testing region is achievable. The converse is also true, in the case we make deterministic decisions on S_{reject}. As the mode collapse region is defined as a convex hull of all achievable points, the points in the hypothesis testing region that require randomized decisions can also be covered. For example, the hypothesis testing regions of the toy examples from Fig. 3.3 are shown below in Fig. 3.4.

Properties of the mode collapse region. Given the equivalence between the mode collapse region and the binary hypothesis testing region, several important properties follow as corollaries. First, the hypothesis testing region is a sufficient statistic for the purpose of binary hypothesis testing from a pair of distributions (P, Q). This implies, among other things, that all f-divergences can be derived from the region. In particular, for the purpose of GAN with 0-1 loss, we can define total variation as a geometric property of the region, which is crucial to proving our main results.

Remark 3.2.1.2 (Total variation distance)**.** *The total variation distance between P and Q is the intersection between the vertical axis and the tangent line to the upper boundary of $\mathcal{R}(P, Q)$ that has a slope of one, as shown in Fig. 3.5.*

This follows from the equivalence of the mode collapse region (Remark 3.2.1.1) and the hypothesis testing region. This geometric definition of total variation allows us to enumerate over all pairs (P, Q) that have the same total variation τ in our proof, via enumerating over all regions that touch the line that has a unit slope and a shift τ (see Fig. A.2).

The major strength of the region perspective, as originally studied by Blackwell [47], is in providing a comparison of stochastic experiments. In our GAN context, consider comparing two pairs of target distributions and generators (P, Q) and (P', Q') as follows. First, a hypothesis h is drawn, choosing whether to produce samples from the true distribution, in which case we say $h = 1$, or to produce samples from the generator, in which case we

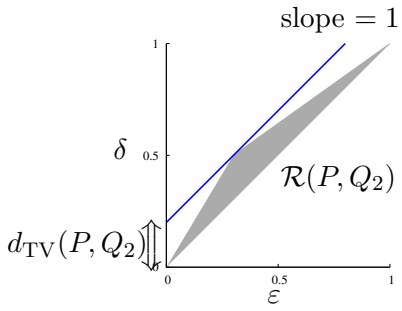


Figure 3.5: Total variation distance is one among many properties of (P, Q_2) that can be directly read off of the region $\mathcal{R}(P, Q)$.

say $h = 0$. Conditioned on this hypothesis h, we use X to denote a random variable that is drawn from the first pair (P, Q) such that $f_{X|h}(x|1) = P(x)$ and $f_{X|h}(x|0) = Q(x)$. Similarly, we use X' to denote a random sample from the second pair, where $f_{X'|h}(x|1) = P'(x)$ and $f_{X'|h}(x|0) = Q'(x)$. Note that the conditional distributions are well-defined for both X and X', but there is no coupling defined between them. Suppose h is independently drawn from the uniform distribution.

Definition 3.2.1.2. *For a given coupling between X and X', we say X* dominates *X' if they form a Markov chain h–X–X'.*

The *data processing inequality* in the following remark shows that if we further *process* the output samples from the pair (P, Q) then the further processed samples can only have less mode collapse. Processing output of stochastic experiments has the effect of smoothing out the distributions, and mode collapse, which corresponds to a *peak* in the pair of distributions, are smoothed out in the processing down the Markov chain.

Remark 3.2.1.3 (Data processing inequality)**.** *The following data processing inequality holds for the mode collapse region. For two coupled target-generator pairs (P, Q) and (P', Q'), if X* dominates *another pair X', then*

$$\mathcal{R}(P', Q') \quad \subseteq \quad \mathcal{R}(P, Q) \; .$$

This is expected, and follows directly from the equivalence of the mode collapse region (Remark 3.2.1.1) and the hypothesis testing region, and corresponding data processing

inequality of hypothesis testing region in [89]. What is perhaps surprising is that the reverse is also true.

Remark 3.2.1.4 (Reverse data processing inequality)**.** *The following reverse data processing inequality holds for the mode collapse region. For two paired marginal distributions X and X', if*

$$\mathcal{R}(P', Q') \quad \subseteq \quad \mathcal{R}(P, Q)\ ,$$

then there exists a coupling of the random samples from X and X' such that X dominates X', i.e. they form a Markov chain h–X–X'.

This follows from the equivalence between the mode collapse region and the hypothesis testing region (Remark 3.2.1.1) and Blackwell's celebrated result on comparisons of stochastic experiments [47] (see [89] for a simpler version of the statement). This region interpretation, and the accompanying (reverse) data processing inequality, abstracts away all the details about P and Q, enabling us to use geometric analysis tools to prove our results. In proving our main results, we will mainly rely on the following remark, which is the corollary of the Remark 3.2.1.3 and Remark 3.2.1.4.

Remark 3.2.1.5. *For all positive integers m, the dominance of regions are preserved under taking m-th order product distributions, i.e. if $\mathcal{R}(P', Q') \subseteq \mathcal{R}(P, Q)$, then $\mathcal{R}((P')^m, (Q')^m) \subseteq \mathcal{R}(P^m, Q^m)$.*

The Effect of "Packing"

As showed previously in Fig. 3.3 , a vanilla GAN discriminator, observing only the TV distance between generator distributions Q and the true distribution P, cannot distinguish between two candidate generators Q_1 and Q_2 with $d_{TV}(P, Q_1) = d_{TV}(P, Q_2)$, but different mode collapse regions. The key insight of this work is that by instead considering *product* distributions, the total variation distance $d_{TV}(P^m, Q^m)$ varies in a way that is closely tied to the mode collapse regions for (P, Q). For instance, Fig. 3.6 (left) shows an achievable range of $d_{\mathrm{TV}}(P^m, Q^m)$ conditioned on that $d_{\mathrm{TV}}(P, Q) = \tau$ for $\tau = 1.1$. Within this achievable range, some pairs (P, Q) have rapidly increasing total variation, occupying the upper part of the region (shown in red, middle panel of Fig. 3.6), and others have slowly increasing total variation, occupying the lower part (shown in blue) in the right panel of Fig. 3.6. We formally show in the following that there is a fundamental connection between total variation distance evolution and degree of mode collapse. Namely, distributions with

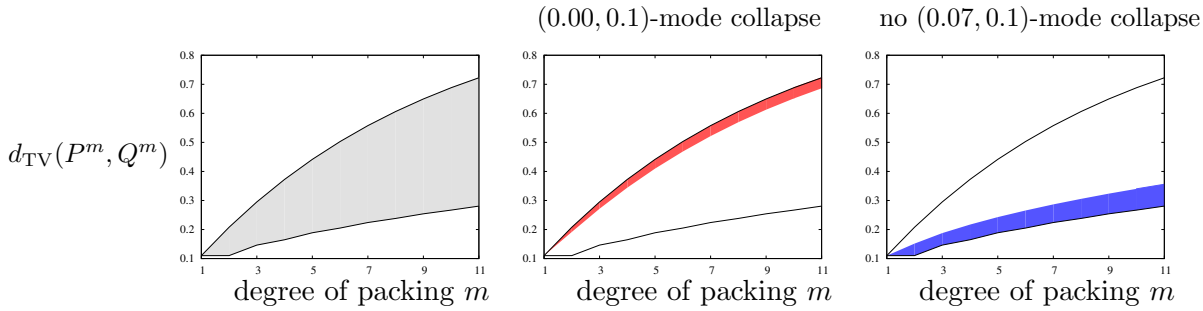


Figure 3.6: The range of $d_{\mathrm{TV}}(P^m, Q^m)$ achievable by pairs with $d_{\mathrm{TV}}(P, Q) = \tau$, for a choice of $\tau = 0.11$, defined by the solutions of the optimization Eq. (A.1) provided in Theorem A.1.1.1 in the Appendix (left panel). The range of $d_{\mathrm{TV}}(P^m, Q^m)$ achievable by those pairs that also have $(\varepsilon = 0.00, \delta = 0.1)$-mode collapse (middle panel). A similar range achievable by pairs of distributions that do not have $(\varepsilon = 0.07, \delta = 0.1)$-mode collapse or $(\varepsilon = 0.07, \delta = 0.1)$-mode augmentation (right panel). Pairs (P, Q) with strong mode collapse occupy the top region (near the upper bound) and the pairs with weak mode collapse occupy the bottom region (near the lower bound).

strong mode collapse occupy the upper region, and hence will be penalized by a packed discriminator.

Evolution of total variation distances with mode collapse. We analyze how the total variation evolves for the set of all pairs (P, Q) that have the same total variation distances τ when unpacked, with $m = 1$, and have (ε, δ)-mode collapse for some $0 \le \varepsilon < \delta \le 1$. The solution of the following optimization problem gives the desired range:

$$\begin{aligned} &\min_{P,Q} \text{ or } \max_{P,Q} \quad d_{\mathrm{TV}}(P^m, Q^m) \\ &\text{subject to} \quad d_{\mathrm{TV}}(P, Q) = \tau \\ &\qquad\qquad (P, Q) \text{ has } (\varepsilon, \delta)\text{-mode collapse} , \end{aligned} \tag{3.1}$$

where the maximization and minimization are over all probability measures P and Q, and the mode collapse constraint is defined in Definition 3.2.1.1. We provide the optimal solution analytically and establish that mode-collapsing pairs occupy the upper part of the total variation region; that is, total variation increases rapidly as we pack more samples together (Fig. 3.6, middle panel).

Theorem 3.2.1.1. *For all $0 \le \varepsilon < \delta \le 1$ and an integer m, if $1 \ge \tau \ge \delta - \varepsilon$ then*

the solution to the maximization in Eq. (3.1) is $1-(1-\tau)^m$*, and the solution to the minimization is*

$$\min\Bigg\{\min_{0\le\alpha\le 1-\frac{\tau\delta}{\delta-\varepsilon}} d_{\mathrm{TV}}\Big(P_{\mathrm{inner1}}(\alpha)^m, Q_{\mathrm{inner1}}(\alpha)^m\Big), \min_{1-\frac{\tau\delta}{\delta-\varepsilon}\le\alpha\le 1-\tau} d_{\mathrm{TV}}\Big(P_{\mathrm{inner2}}(\alpha)^m, Q_{\mathrm{inner2}}(\alpha)^m\Big)\Bigg\},$$

where $P_{\mathrm{inner1}}(\alpha)^m$*,* $Q_{\mathrm{inner1}}(\alpha)^m$*,* $P_{\mathrm{inner2}}(\alpha)^m$*, and* $Q_{\mathrm{inner2}}(\alpha)^m$ *are the* m*-th order product distributions of discrete random variables distributed as*

$$P_{\mathrm{inner1}}(\delta,\alpha) = \Big[\delta,\quad 1-\alpha-\delta,\quad \alpha\Big], \tag{3.2}$$

$$Q_{\mathrm{inner1}}(\varepsilon,\alpha,\tau) = \Big[\varepsilon,\quad 1-\alpha-\tau-\varepsilon,\quad \alpha+\tau\Big], \tag{3.3}$$

$$P_{\mathrm{inner2}}(\alpha) = \Big[1-\alpha,\quad \alpha\Big], \tag{3.4}$$

$$Q_{\mathrm{inner2}}(\alpha,\tau) = \Big[1-\alpha-\tau,\quad \alpha+\tau\Big]. \tag{3.5}$$

If $\tau<\delta-\varepsilon$*, then the optimization in Eq. (3.1) has no solution and the feasible set is an empty set.*

One implication is that distribution pairs (P,Q) at the top of the total variation evolution region are those with the strongest mode collapse. Another implication is that a pair (P,Q) with strong mode collapse (i.e., with larger δ and smaller ε in the constraint) will be penalized more under packing, and hence a generator minimizing an approximation of $d_{\mathrm{TV}}(P^m,Q^m)$ will be unlikely to select a distribution that exhibits such strong mode collapse.

Evolution of total variation distances without mode collapse. We next analyze how the total variation evolves for the set of all pairs (P,Q) that have the same total variations distances τ when unpacked, with $m=1$, and *do not* have (ε,δ)-mode collapse for some $0\le\varepsilon<\delta\le 1$. Because of the symmetry of the total variation distance, mode collapse for (Q,P) is equally damaging as mode collapse of (P,Q), when it comes to how fast total variation distances evolve. Hence, we characterize this evolution for those family of pairs of distributions that do not have either mode collapses. The solution of the following

optimization problem gives the desired range of total variation distances:

$$\begin{aligned} \min_{P,Q} \text{ or } \max_{P,Q} \quad & d_{\mathrm{TV}}(P^m, Q^m) \\ \text{subject to} \quad & d_{\mathrm{TV}}(P, Q) = \tau \\ & (P, Q) \text{ does not have } (\varepsilon, \delta)\text{-mode collapse}, \\ & (Q, P) \text{ does not have } (\varepsilon, \delta)\text{-mode collapse}, \end{aligned} \tag{3.6}$$

We provide thte optimal solution analytically and establish that the pairs (P, Q) with weak mode collapse will occupy the bottom part of the evolution of the total variation distances (see Fig. 3.6 right panel), and also will be penalized less under packing. Hence a generator minimizing (approximate) $d_{\mathrm{TV}}(P^m, Q^m)$ is likely to generate distributions with weak mode collapse.

Theorem 3.2.1.2. *If $\delta + \varepsilon \leq 1$ and $\delta - \varepsilon \leq \tau \leq (\delta - \varepsilon)/(\delta + \varepsilon)$ then the solution to the maximization in Eq. (3.6) is*

$$\max_{\alpha+\beta \leq 1-\tau, \frac{\varepsilon\tau}{\delta-\varepsilon} \leq \alpha, \beta} \; d_{\mathrm{TV}}\Big(P_{\mathrm{outer1}}(\alpha, \beta)^m, Q_{\mathrm{outer1}}(\alpha, \beta)^m \Big) ,$$

where $P_{\mathrm{outer1}}(\alpha, \beta)^m$ and $Q_{\mathrm{outer1}}(\alpha, \beta)^m$ are the m-th order product distributions of discrete random variables distributed as $P_{\mathrm{outer1}}(\alpha, \beta) = [\frac{\alpha(\delta-\varepsilon)-\varepsilon\tau}{\alpha-\varepsilon}, \frac{\alpha(\alpha+\tau-\delta)}{\alpha-\varepsilon}, 1-\tau-\alpha-\beta, \beta, 0]$ and $Q_{\mathrm{outer1}}(\alpha, \beta) = [0, \alpha, 1-\tau-\alpha-\beta, \frac{\beta(\beta+\tau-\delta)}{\beta-\varepsilon}, \frac{\beta(\delta-\varepsilon)-\varepsilon\tau}{\beta-\varepsilon}]$. The solution to the minimization in Eq. (3.6) is

$$\min_{\frac{\varepsilon\tau}{\delta-\varepsilon} \leq \alpha \leq 1-\frac{\delta\tau}{\delta-\varepsilon}} \; d_{\mathrm{TV}}\Big(P_{\mathrm{inner}}(\alpha)^m, Q_{\mathrm{inner}}(\alpha, \tau)^m \Big) ,$$

where $P_{\mathrm{inner}}(\alpha)$ and $Q_{\mathrm{inner}}(\alpha, \tau)$ are defined as in Theorem A.1.1.1 in the Appendix.

We can prove the exact solution of the optimization for all values of ε and δ, which we provide in the Appendix. We refer also to the appendix of more illustrations of regions occupied by various choices of ε and δ for mode collapsing distributions, and non mode collapsing regions.

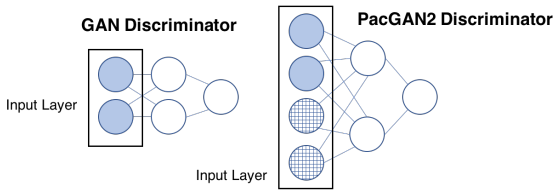


Figure 3.7: PacGAN(m) augments the input layer by a factor of m. The number of weights between the first two layers are increased to preserve the mother architecture's connectivity. Packed samples are concatenated and fed to the input layer; grid-patterned nodes are input nodes for the second sample.

3.2.2 PacGAN for Improving Sample Diversity

There are many ways to implement the idea of packing, each with tradeoffs. In this section, we present a simple packing framework that serves as the basis for our empirical experiments and a concrete example of packing. A primary reason for this architectural choice is to emphasize only the effect of packing in numerical experiments, and isolate it from any other effects that might result from other (more sophisticated) changes to the architecture. However, our analysis in § 3.2.1 is agnostic to the packing implementation, and we discuss potential alternative packing architectures in § 3.2.4, especially those that explicitly impose permutation invariance.

We start with an existing GAN, defined by a generator architecture, a discriminator architecture, and a loss function. We call this triplet the *mother architecture*. The PacGAN framework maintains the same generator architecture, loss function, and hyper-parameters as the mother architecture. However, instead of using a discriminator $D_\eta(x)$ that maps a single sample (either real or generated) to a (soft) label, we use an *augmented* discriminator $D_\eta(x_1, x_2, \ldots, x_m)$ that maps m samples to a single (soft) label. These m samples are drawn independently from the same distribution—either real (jointly labelled $Y = 1$) or generated ($Y = 0$). We refer to the concatenation of samples with the same label as *packing*, the resulting discriminator as a *packed discriminator*, and the number m of concatenated samples as the *degree of packing*. The proposed approach can be applied to any existing GAN architecture and any loss function, as long as it uses a discriminator $D(X)$ that classifies a single input sample. We use the notation "Pac(X)(m)" where (X) is the name of the mother architecture, and (m) is is the packing degree. For example, if we take an original GAN and feed the discriminator three packed samples, we call this "PacGAN3".

We implement packing by keeping all hidden layers of the discriminator identical to the mother architecture, and increasing the number of nodes in the input layer by a factor of m. For example, in Fig. 3.7, we start with a fully-connected, feed-forward discriminator. Each sample x is two-dimensional, so the input layer has two nodes. Under PacGAN2, we multiply the size of the input layer by the packing degree $m = 2$, and the connections to the first hidden layer are adjusted so that the first two layers remain fully-connected, as in the mother architecture. The grid-patterned nodes in Fig. 3.7 represent input nodes for the second sample. Similarly, when packing a DCGAN [90], which uses convolutional neural networks for both the generator and the discriminator, we simply stack the images into a tensor of depth m. For instance, the discriminator for PacDCGAN4 on the MNIST dataset of handwritten images [91] would take an input of size $28 \times 28 \times 4$, since each individual black-and-white MNIST image is 28×28 pixels. Only the input layer and the number of weights in the corresponding first convolutional layer will increase in depth by a factor of 4. As in standard GANs, we train the packed discriminator with a bag of samples from the real data and the generator. However, each minibatch in the stochastic gradient descent now consists of *packed* samples $(x_1, x_2, \ldots, x_m, Y)$, which the discriminator jointly classifies. Intuitively, packing helps the discriminator detect mode collapse because lack of diversity is more obvious in a set of samples than in a single sample.

3.2.3 Experiments

On standard benchmark datasets, we compare PacGAN to several baseline GAN architectures, some explicitly designed to mitigate mode collapse: GAN [75], minibatch discrimination (MD) [85], DCGAN [92], VEEGAN [5], Unrolled GANs [87], and ALI [93]. We also implicitly compare against BIGAN [94], which is conceptually identical to ALI. To isolate the effects of packing, we make minimal choices in the architecture and hyper-parameters of our packing implementation. Our goal is to reproduce experiments from the literature, apply packing to the simplest baseline GAN, and observe how packing affects performance. Whenever possible, we use the exactly same choice of architecture, hyper-parameters, and loss function as a baseline in each experiment; we change only the discriminator to accept packed samples. All code to reproduce our experiments can be found at https://github.com/fjxmlzn/PacGAN.

Metrics. We measure several previously-used metrics. The first is *number of modes* that are produced by a generator [94, 87, 5]. In labelled datasets, this number can be evaluated

using a third-party trained classifier that classifies the generated samples [5]. A second metric used in [5] is the *number of high-quality samples*, which is the proportion of the samples that are within x standard deviations from the center of a mode. Finally, we measure the *reverse Kullback-Leibler divergence* between the induced distribution from generated samples and the induced distribution from the real samples. Each of these metrics has shortcomings—for example, the number of observed modes ignores class imbalance, and all of the metrics assume the modes are known a priori.

Datasets. We use synthetic and real datasets. The **2D Ring Dataset** [5] is a mixture of eight two-dimensional spherical Gaussians with means $(\cos((2\pi/8)i), \sin((2\pi/8)i))$ and variances 10^{-4} in each dimension for $i \in \{1, \ldots, 8\}$. The **2D Grid Dataset** [5] is a mixture of 25 two-dimensional spherical Gaussians with means $(-4 + 2i, -4 + 2j)$ and variances 0.0025 in each dimension for $i, j \in \{0, 1, 2, 3, 4\}$. The MNIST dataset [91] consists of 70K images of handwritten digits, each 28×28 pixels. Unmodified, this dataset has 10 modes (digits). As in [87, 5], we augment the number of modes by *stacking* the images: we generate a new dataset of 128K images where each image consists of three random MNIST images stacked into a $28 \times 28 \times 3$ RGB image. This new **Stacked MNIST Dataset** has (with high probability) $1000 = 10 \times 10 \times 10$ modes. Finally, we include experiments on the **CelebA** dataset, which is a collection of 200K facial images of celebrities [2].

Synthetic data experiments

Our first experiment measures the effect of the number of discriminator parameters on mode collapse. Packed architectures have more discriminator nodes (and parameters) than the mother architecture, which could artificially mitigate mode collapse by giving the discriminator more capacity. We compare this effect to the effect of packing on the 2D grid dataset. In Fig. 3.8, we evaluate the number of modes recovered and reverse KL divergence for ALI, GAN, MD, and PacGAN, while varying the number of *total* parameters in each architecture (discriminator and encoder if one exists). For MD, the metrics first improve and then degrade with the number of parameters. We suspect that this may because MD is very sensitive to experiment settings, as the same architecture of MD has very different performance on 2D Grid Dataset and 2D Ring Dataset. For ALI, GAN and PacGAN, despite varying the number of parameters by an order of magnitude, we do not see significant evidence of the metrics improving with the number of parameters. This suggests that the advantages of PacGAN and ALI compared to GAN do not stem from

having more parameters. Packing also seems to increase the number of modes recovered and decrease the reverse KL divergence. Fig. 3.1 gives visualization of the results.

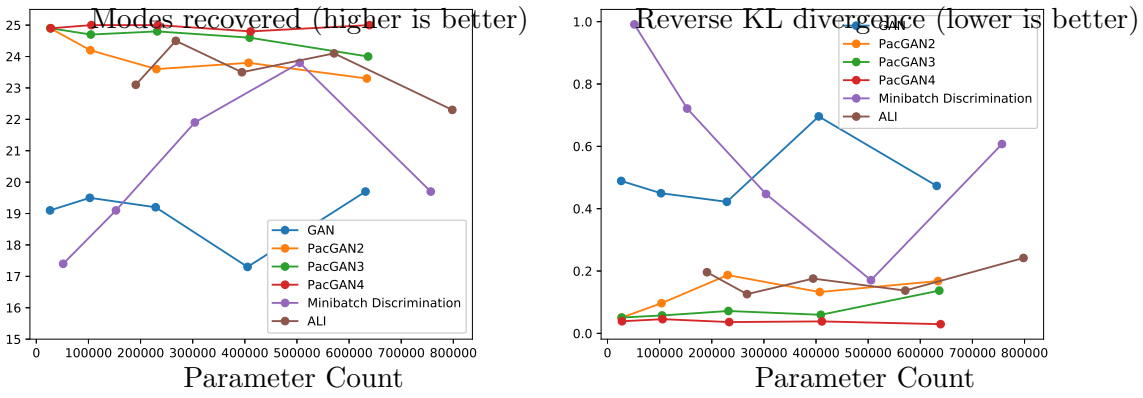


Figure 3.8: Effect of number of parameters on evaluation metrics.

Stacked MNIST experiments

For our stacked MNIST experiments, we generate samples. Each of the three channels in each sample is classified by a pretrained third-party MNIST classifier, and the resulting three digits determine which of the 1000 modes the sample belongs to. We measure the number of modes captured, as well as the KL divergence between the generated distribution over modes and the expected (uniform) one.

In the first experiment, we replicate Table 2 from [5], which measured the number of observed modes in a generator trained on the stacked MNIST dataset, as well as the KL divergence of the generated mode distribution. In line with [5], we used a DCGAN-like architecture for these experiments[1]. Our results are shown in Table 3.1. The first four rows are copied directly from [5]. The last three rows are computed using a basic DCGAN, with packing in the discriminator. We find that packing gives good mode coverage, reaching all 1,000 modes in every trial. Fig. 3.2 also confirms that PacGAN not only increases sample diversity, but also improves sample quality. Again, packing the simplest DCGAN fully captures all the modes in the benchmark test, so we do not pursue packing more complex baseline architectures. We also observe that MD is very unstable throughout training,

[1] https://github.com/carpedm20/DCGAN-tensorflow

which makes it capture even less modes than GAN. One factor that contributes to MD's instability may be that MD requires too many parameters. The number of discriminator parameters in MD is 47,976,773, whereas GAN has 4,310,401 and PacGAN4 only needs 4,324,801.

	Stacked MNIST	
	Modes	KL
DCGAN [92]	99.0	3.40
ALI [93]	16.0	5.40
Unrolled GAN [87]	48.7	4.32
VEEGAN [5]	150.0	2.95
Minibatch Discrimination [85]	24.5±7.67	5.49±0.418
DCGAN (our implementation)	78.9±6.46	4.50±0.127
PacDCGAN2 (ours)	1000.0±0.00	0.06±0.003
PacDCGAN3 (ours)	1000.0±0.00	0.06±0.003
PacDCGAN4 (ours)	1000.0±0.00	0.07±0.005

Table 3.1: Two measures of mode collapse proposed in [5] for the stacked MNIST dataset: number of modes captured by the generator and reverse KL divergence over the generated mode distribution. The DCGAN, PacDCGAN, and MD results are averaged over 10 trials, with standard error reported.

CelebA Experiments

In this experiment, we measure the diversity of images generated from the celebA dataset as proposed by Arora et al. [95]. They suggest measuring the diversity by estimating the probability of collision in a finite batch of images sampled from the generator. If there exists at least one pair of near-duplicate images in the batch it is declared to have a collision. To detect collision in a batch of samples, they select the 20 closest pairs from it according to the Euclidean distance in pixel space, and then visually identify if any of them would be considered duplicates by humans. For visual identification, we take majority vote of three human reviewers for each batch of samples. To estimate the probability we repeat the experiment 20 times.

We use DCGAN- unconditional, with JSD objective as described in [92] as the base architecture. We perform the experiment for different sizes of the discriminator while fixing the other hyper-parameters. The DCGAN- [92] uses 4 CNN layers with the number

of output channels of each layer being $dim \times 1, 2, 4, 8$. Thus the discriminator size is proportional to dim^2. Table 3.2 shows probability of collision in a batch of size 1024 for DCGAN and PacDCGAN2 for $dim \in \{16, 32, 64, 80\}$. Packing significantly improves diversity of samples. If the size of the discriminator is small, then packing also improves quality of the samples. Fig. 3.9 shows samples generated from DCGAN and PacDCGAN2 for $dim = 16$. We note that DCGAN and PacDCGAN2 use approximately same number of parameters, 273K and 274K respectively.

discriminator size	probability of collision	
	DCGAN	PacDCGAN2
d^2	1	0.3
$4\,d^2$	0.4	0
$16\,d^2$	0.8	0
$25\,d^2$	0.6	0.2

Table 3.2: Probability of at least one pair of near-duplicate images being present in a batch of 1024 images generated from DCGAN and PacDCGAN2 on celebA dataset show that PacDCGAN2 generates more diverse images.

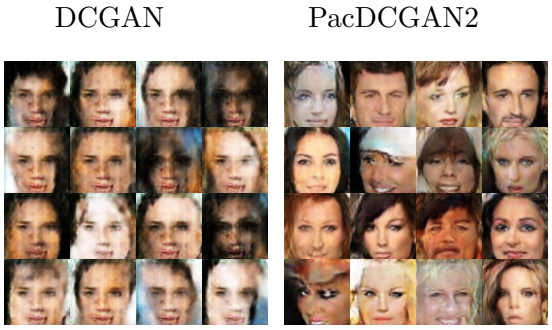


Figure 3.9: CelebA samples generated from DCGAN (left) and PacDCGAN2 (right) show PacDCGAN2 generates more diverse and sharper images.

3.2.4 Discussions

In this section, we propose a packing framework that theoretically and empirically mitigates mode collapse with low overhead. Our analysis leads to several interesting open questions,

including how to apply these analysis techniques to more general classes of loss functions such as Jensen-Shannon divergence and Wasserstein distances. Another important question is what packing architecture to use. For instance, a framework that provides permutation invariance may give better results such as graph neural networks [96, 97, 98] or deep sets [99].

3.3 Improving Training Stability

Many approaches have been proposed for improving the stability of GANs, including different architectures [100, 41, 101], loss functions [102, 103, 104, 105], and various types of regularizations/normalizations [43, 106, 107]. One of the most successful proposals to date is called *spectral normalization* (SN) [43, 108, 48]. SN forces each layer of the discriminator to have unit spectral norm during training. This has the effect of controlling the Lipschitz constant of the discriminator, which is empirically observed to improve the stability of GAN training [43]. Despite the successful applications of SN [101, 109, 110, 111, 112, 113, 114], to date, it remains unclear precisely why this specific normalization is so effective.

In this section, we first present our theoretical analysis of spectral normalization (§ 3.3.1). Based on the theoretical insights, we introduce BSSN, our practical algorithm for further stabilizing GAN training and improving sample fidelity (§ 3.3.2), and its experimental results (§ 3.3.3). Finally, we summarize the contributions and discuss future work (§ 3.3.4).

3.3.1 Theoretical Analysis of Spectral Normalization

We first present the preliminaries on spectral normalization, and then provide our theoretical results.

Preliminaries

The instability of GANs is believed to be predominantly caused by poor discriminator learning [102, 115]. We therefore focus in this work on the discriminator. We adopt the same model as [43]. Consider a discriminator with L internal layers:

$$D_\eta(x) = a_L \circ l_{w_L} \circ a_{L-1} \circ l_{w_{L-1}} \circ \ldots \circ a_1 \circ l_{w_1}(x) \tag{3.7}$$

where x denotes the input to the discriminator and $\eta = \{w_1, w_2, ..., w_L\}$ the weights; $a_i\,(i = 1, ..., L-1)$ is the activation function in the i-th layer, which is usually element-wise ReLU or leaky ReLU in GANs [1]. a_L is the activation function for the last layer, which is sigmoid for the vanilla GAN [1] and identity for WGAN-GP [104]; l_{w_i} is the linear transformation in i-th layer, which is usually fully-connected or a convolutional neural network [1, 100]. Like prior work on the theoretical analysis of (spectral) normalization [43, 48, 116, 117], we do not model bias terms.

Lipschitz regularization and spectral normalization. Prior work has shown that regularizing the Lipschitz constant of the discriminator $\|D_\eta\|_{\text{Lip}}$ improves the stability of GANs [103, 104, 105]. For example, WGAN-GP [104] adds a gradient penalty $(\|\nabla D_\eta\,(\tilde{x})\| - 1)^2$ to the loss function, where $\tilde{x} = \alpha x + (1-\alpha) G_\theta\,(z)$ and $\alpha \sim \text{U}\,(0, 1)$ to ensure that the Lipschitz constant of the discriminator is bounded by 1.

Spectral normalization (SN) takes a different approach. For fully connected layers (i.e., $l_{w_i}(x) = w_i x$), it regularizes the weights w_i to ensure that spectral norm $\|w_i\|_{\text{sp}} = 1$ for all $i \in [1, L]$, where the spectral norm $\|w_i\|_{\text{sp}}$ is defined as the largest singular value of w_i. This bounds the Lipschitz constant of the discriminator since $\|D_\eta\|_{\text{Lip}} \leq \prod_{i=1}^{L} \|l_{w_i}\|_{\text{Lip}} \cdot \prod_{i=1}^{L} \|a_i\|_{\text{Lip}} \leq \prod_{i=1}^{L} \|w_i\|_{\text{sp}} \cdot \prod_{i=1}^{L} \|a_i\|_{\text{Lip}} \leq 1$, as $\|l_{w_i}\|_{\text{Lip}} \leq \|w_i\|_{\text{sp}}$ and $\|a_i\|_{\text{Lip}} \leq 1$ for networks with (leaky) ReLU as activation functions for the internal layers and identity/sigmoid as the activation function for the last layer [43]. Prior work has theoretically connected the generalization gap of neural networks to the product of the spectral norms of the layers [118, 119, 117, 120]. These insights led to multiple implementations of spectral normalization [48, 108, 121, 43, 117, 120], with the implementation of [43] achieving particular success on GANs. SN can be viewed as a special case of more general techniques for enhancing the stability of neural network training by controlling the entire spectrum of a network's input-output Jacobian [122], weight matrices [117], or learned embeddings [120].

In practice, spectral normalization [48, 43] is implemented by dividing the weight matrix w_i by its spectral norm: $\frac{w_i}{u_i^T w_i v_i}$, where u_i and v_i are the left/right singular vectors of w_i corresponding to its largest singular value. As observed by Gouk et al. [108], there are two approaches in the SN literature for instantiating the matrix w_i for convolutional neural networks (CNNs). In a CNN, since convolution is a linear operation, convolutional layers can equivalently be written as a multiplication by an expanded weight matrix $\tilde{w}_i$ that is derived from the raw weights w_i. Hence in principle, spectral normalization should normalize each convolutional layer by $\|\tilde{w}_i\|_{\text{sp}}$ [108, 48]. We call this canonical normalization

SN_{Conv} as it controls the spectral norm of the convolution layer.

However, the spectral normalization that is known to outperform other regularization techniques and improves training stability for GANs [43], which we call SN_{w}, does not implement SN in a strict sense. Instead, it uses $\left\|w_i^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\text{sp}}$; that is, it first reshapes the convolution kernel $w_i \in \mathbb{R}^{c_{out}c_{in}k_wk_h}$ into a matrix $\hat{w}_i$ of shape $c_{out} \times (c_{in}k_wk_h)$, and then normalizes with the spectral norm $\|\hat{w}_i\|_{\text{sp}}$, where c_{in} is the number of input channels, c_{out} is the number of output channels, k_w is the kernel width, and k_h is the kernel height. Miyato et al. showed that their implementation implicitly penalizes w_i from being too sensitive in one specific direction [43]. However, this does not explain why SN_{w} is more stable than other Lipschitz regularization techniques, and as observed in [108], it is unclear how SN_{w} relates to SN_{Conv}. Despite this, SN_{w} has empirically been immensely successful in stabilizing the training of GANs [101, 109, 110, 111, 112, 113, 114]. Even more puzzling, we show in § 3.3.1 that the canonical approach SN_{Conv} has comparatively poor out-of-the-box performance when training GANs.

Hence, two questions arise: (1) Why is SN so successful at stabilizing the training of GANs? (2) Why is SN_{w} proposed by [43] so much more effective than the canonical SN_{Conv}?

In this section, we show that both questions are related to two well-known phenomena: vanishing and exploding gradients. These terms describe a problem in which gradients either grow or shrink rapidly during training [123, 124, 125, 126], and they are known to be closely related to the instability of GANs [102, 101]. To illustrate that gradient explosion and vanishing are closely related to the instability in GANs, we trained a WGAN [104] on the CIFAR10 Dataset with different hyper-parameters leading to stable training, exploding gradients, and vanishing gradients over 40,000 training iterations. Fig. 3.10 shows the resulting inception scores for each of these runs, and Fig. 3.11 shows the corresponding magnitudes of the gradients over the course of training. Note that the stable run has improved sample quality and stable gradients throughout training. This phenomenon has also been observed in prior literature [102, 101]. We will demonstrate that by controlling these gradients, SN (and SN_{w} in particular) is able to achieve more stable training and better sample quality.

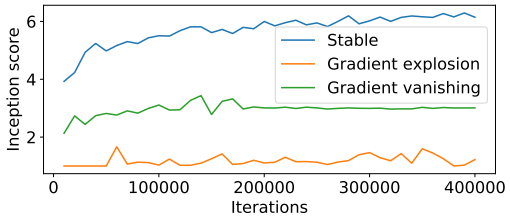


Figure 3.10: Inception score over the course of training. The "gradient vanishing" inception score plateaus as training is stalled.

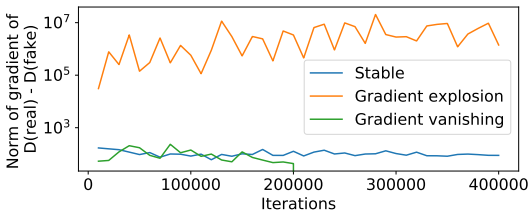


Figure 3.11: Norm of gradient with respect to parameters during training. The vanishing gradient collapses after 200k iterations.

SN Controls Exploding Gradients

In this section, we show that spectral normalization prevents gradient explosion by bounding the gradients of the discriminator. Moreover, we show that the common choice to normalize all layers equally achieves the tightest upper bound for a restricted class of discriminators. We use $\eta \in \mathbb{R}^{d_2}$ to denote a vector containing all elements in $\{w_1, ..., w_L\}$. In the following analysis, we assume linear transformations are fully-connected layers $l_{w_i}(x) = w_i x$ as in [43], though the same analysis can be applied to convolutional layers. Following prior work on the theoretical analysis of (spectral) normalization [43, 48, 116], we assume no bias in the network (i.e., Eq. (3.7)) for simplicity.

To highlight the effects of the spectral norm of each layer on the gradient and simplify the exposition, we will compute gradients with respect to $w_i' = \frac{w_i}{u_i^T w_i v_i}$ in the following discussion. In reality, gradients are computed with respect to w_i; we defer this discussion to Appendix A.2.1, where we show the relevant extension.

How SN controls exploding gradients. The following proposition shows that under this simplifying assumption, spectral normalization controls the magnitudes of the gradients of the discriminator with respect to η. Notice that simply controlling the Lipschitz constant of the discriminator (e.g., as in WGAN [102]) does not imply this property; it instead ensures small (sub)gradients with respect to the input, x.

Proposition 3.3.1.1 (Upper bound of gradient's Frobenius norm for spectral normalization)**.** *If* $\|w_i\|_{sp} \leq 1$ *for all* $i \in [1, L]$*, then we have* $\|\nabla_{w_i} D_\eta(x)\|_F \leq \|x\| \prod_{i=1}^{L} \|a_i\|_{Lip}$*, and the norm of the overall gradient can be bounded by* $\|\nabla_\eta D_\eta(x)\|_F \leq \sqrt{L}\, \|x\| \prod_{i=1}^{L} \|a_i\|_{Lip}$.

Note that under the assumption that internal activation functions are ReLU or leaky

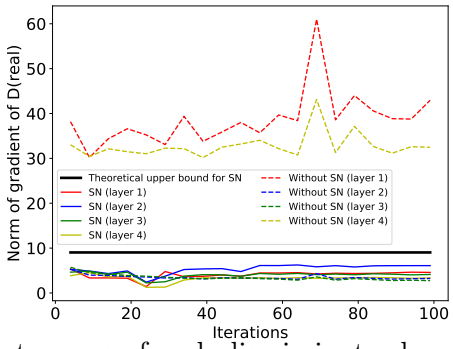


Figure 3.12: Gradient norms of each discriminator layer in MNIST Dataset.

ReLU, if the activation function for the last layer is identity (e.g., for WGAN-GP [104]), the above bounds can be simplified to $\|\nabla_{w_t} D_\eta(x)\|_{\mathrm{F}} \leq \|x\|$ and $\|\nabla_\eta D_\eta(x)\| \leq \sqrt{L}\,\|x\|$, and if the activation for the last layer is sigmoid (e.g., for vanilla GAN [1]), the above bounds become $\|\nabla_{w_t} D_\eta(x)\|_{\mathrm{F}} \leq 0.25\,\|x\|$ and $\|\nabla_\eta D_\eta(x)\| \leq 0.25\sqrt{L}\,\|x\|$.

The bound in Proposition 3.3.1.1 has a significant effect in practice. Fig. 3.12 shows the norm of the gradient for each layer of a GAN trained on MNIST with and without spectral normalization. Without spectral normalization, some layers have extremely large gradients throughout training, which makes the overall gradient large. With spectral normalization, the gradients of all layers are upper bounded as shown in Proposition 3.3.1.1.

Optimal spectral norm allocation. Common implementations of SN advocate setting the spectral norm of *each layer* to the same value [43, 48]. However, the following proposition states that we can set the spectral norms of different layers to different constants, without changing the network's behavior on the input samples, as long as the *product* of the spectral norm bounds is the same.

Proposition 3.3.1.2. *For any discriminator* $D_\eta = a_L \circ l_{w_L} \circ a_{L-1} \circ l_{w_{L-1}} \circ \ldots \circ a_1 \circ l_{w_1}$ *and* $D'_\eta = a_L \circ l_{c_L \cdot w_L} \circ a_{L-1} \circ l_{c_{L-1} \cdot w_{L-1}} \circ \ldots \circ a_1 \circ l_{c_1 \cdot w_1}$ *where the internal activation functions* $\{a_i\}_{i=1}^{L-1}$ *are ReLU or leaky ReLU, and positive constant scalars* $c_1, ..., c_L$ *satisfy that* $\prod_{i=1}^{L} c_i = 1$*, we have*

$$D_\eta(x) = D'_\eta(x)\ \forall x \quad \text{and} \quad \frac{\partial^n D_\eta(x)}{\partial x^n} = \frac{\partial^n D'_\eta(x)}{\partial x^n}\ \forall x, \forall n \in \mathbb{Z}^+ .$$

Given this observation, it is natural to ask if there is any benefit to setting the spectral norms of each layer equal. It turns out that the answer is yes, under some assumptions

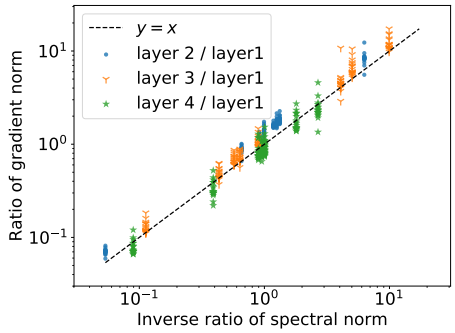


Figure 3.13: Ratio of gradient norm v.s. inverse ratio of spectral norm in MNIST Dataset.

that appear to approximately hold in practice. Let

$$\mathcal{D} \triangleq \left\{ D_\eta = a_L \circ l_{w_L} \circ \ldots \circ a_1 \circ l_{w_1} \;:\; \frac{\|\nabla_{w_i} D_\eta(x)\|_{\mathrm{F}}}{\|\nabla_{w_j} D_\eta(x)\|_{\mathrm{F}}} = \frac{\|w_j\|_{\mathrm{sp}}}{\|w_i\|_{\mathrm{sp}}}, a_i \in \{\text{ReLU, leaky ReLU}\}\ \forall i, j \in [1, L] \right\}. \tag{3.8}$$

This intuitively describes the set of all discriminators for which scaling up the weight of one layer proportionally increases the gradient norm of all other layers; the definition of this set is motivated by our upper bound on the gradient norm (Appendix A.2.2). The following theorem shows that when optimizing over set $\mathcal{D}$, choosing every layer to have the same spectral norm gives the smallest possible gradient norm, for a given set of parameters.

Theorem 3.3.1.1. *Consider a given set of discriminator parameters* $\eta = \{w_1, ..., w_L\}$. *For a vector* $c = \{c_1, \ldots, c_L\}$, *we denote* $\eta_c \triangleq \{c_t w_t\}_{t=1}^{L}$. *Let* $\lambda_\eta = \prod_{i=1}^{L} \|w_i\|_{sp}^{1/L}$ *denote the geometric mean of the spectral norms of the weights. Then we have*

$$\left\{ \frac{\lambda_\eta}{\|w_1\|_{sp}}, \ldots, \frac{\lambda_\eta}{\|w_L\|_{sp}} \right\} = \arg \min_{c:\, D_{\eta_c} \in \mathcal{D},\, \prod_{i=1}^{L} c_i = 1,\, c_i \in \mathbb{R}^+} \|\nabla_{\eta_c} D_{\eta_c}(x)\|_F$$

The key constraint in this theorem is that we optimize only over discriminators in set $\mathcal{D}$ in Eq. (3.8). To show that this constraint is realistic (i.e., SN GAN discriminator optimization tends to choose models in $\mathcal{D}$), we trained a spectrally-normalized GAN with four hidden layers on MNIST, computing the ratios of the gradient norms at each layer and the ratios of the spectral norms, as dictated by Eq. (3.8). We computed these ratios at different epochs during training, as well as for different randomly-selected rescalings of the spectral normalization vector c. Each point in Fig. 3.13 represents the results averaged

over 64 real samples at a specific epoch of training for a given (random) c. Vertical series of points are from different epochs of the same run, therefore their ratio of spectral norms is the same. The fact that most of the points are near the diagonal line suggests that training naturally favors discriminators that are in or near $\mathcal{D}$. This observation, combined with Theorem 3.3.1.1, suggests that it is better to force the spectral norms of every layer to be equal. Hence, existing SN implementations [43, 48] chose the correct, uniform normalization across layers to upper bound discriminator's gradients.

Implications on other normalization/regularization techniques: Note that the analysis in this section can also be used to show the same results for other normalization/regularization techniques that control the spectral norm of weights, like weight normalization [107] and orthogonal regularization [106]. However, these techniques do not necessarily exhibit the more important properties proved in the next section for SN, and have some known drawbacks (see more discussion in § 3.3.4).

SN Controls Vanishing Gradients

An equally troublesome failure mode of GAN training is vanishing gradients [102]. Prior work has proposed new objective functions to mitigate this problem [102, 103, 104], but these approaches are not fully effective (Fig. 3.11). In this section, we show that SN also helps to control vanishing gradients.

How SN controls vanishing gradients: Gradients tend to vanish for two reasons. First, gradients vanish when the objective function *saturates* [127, 102], which is often associated with function parameters growing too large. Common loss functions (e.g., hinge loss) and activation functions (e.g., sigmoid, tanh) saturate for inputs of large magnitude. Large parameters tend to amplify the inputs to the activation functions and/or loss functions, causing saturation. Second, gradients vanish when function parameters (and hence, internal outputs) grow too small. This is because backpropagated gradients are scaled by the function parameters (Appendix A.2.2).

These insights motivated the LeCun initialization technique [127]. The key idea is that to prevent gradients from vanishing, we must ensure that the outputs of each neuron do not vanish or explode. If the inputs to a neural unit are uncorrelated random variables with variance 1, then to ensure that the unit's output also has variance (approximately) 1, the weight parameters should be zero-mean random variables with variance of $\frac{1}{n_i}$, where n_i denote the fan-in (number of incoming connections) of layer i [127]. Hence, LeCun initial-

ization prevents gradient vanishing by controlling the variance of the individual parameters. In the following theorem, we show that SN enforces a similar condition.

Theorem 3.3.1.2 (Parameter variance of SN)**.** *For a matrix $A \in \mathbb{R}^{m\times n}$ with i.i.d. entries a_{ij} from a symmetric distribution with zero mean (e.g., zero-mean Gaussian or uniform), we have*

$$Var\left(\frac{a_{ij}}{\|A\|_{sp}}\right) \leq \frac{1}{\max\{m,n\}} \quad . \tag{3.9}$$

Furthermore, if $m, n \geq 2$ and $\max\{m, n\} \geq 3$, and a_{ij} are from a zero-mean Gaussian, we have

$$\frac{L}{\max\{m,n\}\log(\min\{m,n\})} \leq Var\left(\frac{a_{ij}}{\|A\|_{sp}}\right) \leq \frac{1}{\max\{m,n\}} \quad ,$$

where L is a constant which does not depend on m, n.

Hence, spectral normalization forces zero-mean parameters to have a variance that scales inversely with $\max\{m, n\}$. The proof relies on a characterization of extreme values of random vectors drawn from the surface of a high-dimensional unit ball. Many fully-connected, feed-forward neural networks have a fixed width across hidden layers, so $\max\{m, n\}$ corresponds precisely to the fan-in of any neuron in a hidden layer, implying that SN has an effect like LeCun initialization.

Why $\mathbf{SN_w}$ works better than $\mathbf{SN_{Conv}}$. In a CNN, the interpretation of $\max\{m, n\}$ depends on how SN is implemented. Recall that the implementation $\mathrm{SN_w}$ by [43] does not strictly implement SN, but a variant that normalizes by the spectral norm of $\hat{w}_i = w_i^{c_{out}\times(c_{in}k_wk_h)}$. In architectures like DCGAN [100], the larger dimension of $\hat{w}_i$ for *hidden layers* tends to be $c_{in}k_wk_h$, which is exactly the fan-in. This means that SN gets the right variance for hidden layers in CNN.

Perhaps surprisingly, we find empirically that the strict implementation $\mathrm{SN_{Conv}}$ of [48] does *not* prevent gradient vanishing. Figs. 3.14 and 3.15 shows the gradients of $\mathrm{SN_{Conv}}$ vanishing when trained on CIFAR10, leading to a comparatively poor inception score, whereas the gradients of $\mathrm{SN_w}$ remain stable. To understand this phenomenon, recall that $\mathrm{SN_{Conv}}$ normalizes by the spectral norm of an expanded matrix $\tilde{w}_i$ derived from w_i. Theorem 3.3.1.2 does not hold for $\tilde{w}_i$ since its entries are not i.i.d. (even at initialization); hence it cannot be used to explain this effect. However, Corollary 1 in [128] shows that $\|\hat{w}_i\|_{\mathrm{sp}} \leq \|\tilde{w}_i\|_{\mathrm{sp}} \leq \alpha \|\hat{w}_i\|_{\mathrm{sp}}$, where α is a constant only depends on kernel size, input size,

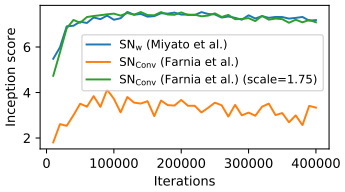


Figure 3.14: Inception score of different SN variants in CIFAR10 Dataset.

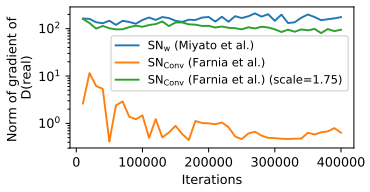


Figure 3.15: Gradient norms of different SN variants in CIFAR10 Dataset.

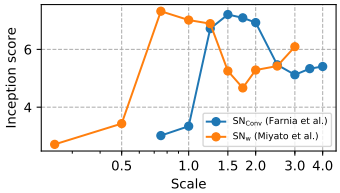


Figure 3.16: Inception score of scaled SN in CIFAR10 Dataset.

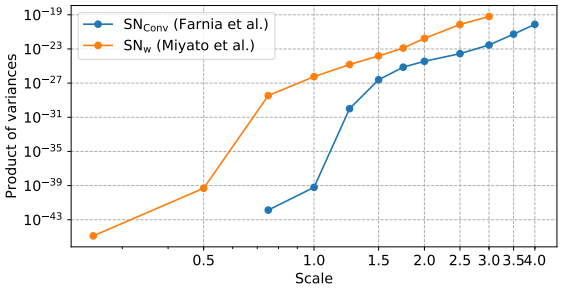


Figure 3.17: The parameter variance of scaled SN in CIFAR10 Dataset.

and stride size of the convolution operation. ([129] also deduced a special case of the second inequality.) This result has two implications:

(1) $\|\tilde{w}_i\|_{\text{sp}} \leq \alpha \|\hat{w}_i\|_{\text{sp}}$: Although SN_w does not strictly normalize the matrix with the actual spectral norm of the layer, it does upper bound the spectral norm of the layer. Therefore, all our analysis in the gradient explosion section still applies for SN_w by changing the spectral norm constant from 1 to $\alpha \|\hat{w}_i\|_{\text{sp}}$. This means that SN_w can still prevent gradient explosion.

(2) $\|\hat{w}_i\|_{\text{sp}} \leq \|\tilde{w}_i\|_{\text{sp}}$: This implies that SN_Conv normalizes by a factor that is at least as large as SN_w. In fact, we observe empirically that $\|\tilde{w}_i\|_{\text{sp}}$ is strictly larger than $\|\hat{w}_i\|_{\text{sp}}$ during training. This means that for the same w_i, a discriminator using SN_Conv will have smaller outputs than the discriminator using SN_w. We hypothesize that the different scalings explain why SN_Conv has vanishing gradients but SN_w does not.

To confirm this hypothesis, for SN_w and SN_Conv, we propose to multiply all the normalized weights by a scaling factor s, which is fixed throughout the training. Fig. 3.16 shows that SN_Conv seems to be a shifted version of SN_w. SN_Conv with $s = 1.75$ has similar inception score (Fig. 3.14) to SN_w, as well as similar gradients (Fig. 3.15) and parameter variances (Fig. 3.17) throughout training. This, combined with Theorem 3.3.1.2, suggests

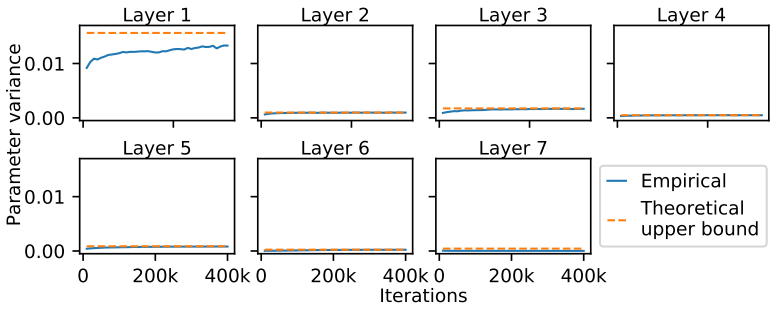


Figure 3.18: Parameter variances throughout training in CIFAR10 Dataset. The blue lines show the parameter variances of different layers when SN is applied, and the original line shows our theoretical upper bound given in Eq. (3.9).

that SN_{w} inherently finds the correct scaling for the problem, whereas "proper" spectral normalization $\mathrm{SN}_{\mathrm{Conv}}$ requires additional hyper-parameter tuning.

SN has good parameter variances throughout training. Our theoretical analysis only applies at initialization, when the parameters are selected randomly. However, unlike LeCun initialization which only controls the variance at initialization, we find empirically that Eq. (3.9) for SN appears to hold *throughout training* (Fig. 3.18). As a comparison, if trained without SN, the variance increases and the gradient decreases, which makes sample quality bad. This explains why in practice GANs trained with SN are stable *throughout training*. Formally extending our theoretical analysis to apply throughout training requires more complicated techniques, which we defer to future work.

3.3.2 Bidirectional Spectral Normalization for Improving Sample Diversity

Given the above theoretical insights, we propose an extension of spectral normalization called Bidirectional Scaled Spectral Normalization (BSSN). It combines two key ideas: bidirectional normalization and weight scaling.

Bidirectional Normalization

Glorot and Bengio [130] built on the intuition of LeCun [127] to design an improved initialization, commonly called *Xavier initialization*. Their key observation was that to limit gradient vanishing (and explosion), it is not enough to control only feed-forward outputs;

we should also control the variance of backpropagated gradients. Let n_i, m_i denote the fan-in and fan-out of layer i. (In fully-connected layers, $n_i = m_{i-1}$ = the width of layer i.) Whereas LeCun chooses initial parameters with variance $\frac{1}{n_i}$, Glorot and Bengio choose them with variance $\frac{2}{n_i+m_i}$, a compromise between $\frac{1}{n_i}$ (to control output variance) and $\frac{1}{m_i}$ (to control variance of backpropagated gradients).

The first component of BSSN is Bidirectional Spectral Normalization (BSN), which applies a similar intuition to improve the spectral normalization of Miyato *et al.* [43]. For fully connected layers, BSN keeps the normalization the same as $\mathrm{SN_w}$ [43]. For convolution layers, instead of normalizing by $\left\|w^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\mathrm{sp}}$, we normalize by $\sigma_w \triangleq \frac{\left\|w^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\mathrm{sp}}+\left\|w^{c_{in}\times(c_{out}k_wk_h)}\right\|_{\mathrm{sp}}}{2}$, where $\left\|w^{c_{in}\times(c_{out}k_wk_h)}\right\|_{\mathrm{sp}}$ is the spectral norm of the reshaped convolution kernel of dimension $c_{in} \times (c_{out}k_wk_h)$. For calculating these two spectral norms, we use the same power iteration method in [43]. The following theorem gives the theoretical explanation.

Theorem 3.3.2.1 (Parameter variance of BSN)**.** *For a convolutional kernel $w \in \mathbb{R}^{c_{out}c_{in}k_wk_h}$ with i.i.d. entries w_{ij} from a symmetric distribution with zero mean (e.g. zero-mean Gaussian or uniform) where $k_wk_h \geq \max\left\{\frac{c_{out}}{c_{in}}, \frac{c_{in}}{c_{out}}\right\}$, and σ_w defined as above, we have*

$$Var\left(\frac{w_{ij}}{\sigma_w}\right) \leq \frac{2}{c_{in}k_wk_h + c_{out}k_wk_h} \ .$$

Furthermore, if $c_{in}, c_{out} \geq 2$ and $c_{in}k_wk_h, c_{out}k_wk_h \geq 3$, and w_{ij} are from a zero-mean Gaussian distribution, there exists a constant L that does not depend on $c_{in}, c_{out}, k_w, k_h$ such that

$$\frac{L}{c_{in}k_wk_h\log(c_{out}) + c_{out}k_wk_h\log(c_{in})} \leq Var\left(\frac{w_{ij}}{\sigma_w}\right) \leq \frac{2}{c_{in}k_wk_h + c_{out}k_wk_h}.$$

In convolution layers, $n_i = c_{in}k_wk_h$ and $m_i = c_{out}k_wk_h$. Therefore, BSN sets the variance of parameters to scale as $\frac{2}{n_i+m_i}$, as dictated by Xavier initialization. Moreover, BSN naturally inherits the benefits of SN discussed in § 3.3.1 (e.g., controlling variance throughout training).

Weight Scaling

The second component of BSSN is to multiply all the normalized weights by a constant scaling factor (i.e., as we did in Fig. 3.16). We call the combination of BSN and this weight scaling *Bidirectional Scaled Spectral Normalization* (BSSN). Note that scaling can also be applied independently to SN, which we call Scaled Spectral Normalization (SSN). The scaling is motivated by the following reasons.

(1) The analysis in LeCun and Xavier initialization assumes that the activation functions are linear, which is not true in practice. More recently, Kaiming initialization was proposed to include the effect of non-linear activations [131]. The result is that we should set the variance of parameters to be $2/(1+a^2)$ times the ones in LeCun or Xavier initialization, where a is the negative slope of leaky ReLU. This suggests the importance of a constant scaling.

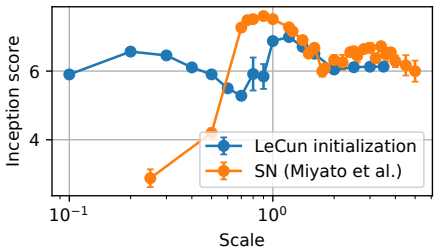


Figure 3.19: Inception score of SSN and scaled LeCun initialization in CIFAR10, with mean and standard error of the best score during training across multiple runs.

(2) However, we found that the scaling constants proposed in LeCun/Kaiming initialization do not always perform well for GANs. Even more surprisingly, there are *multiple modes* of good scaling. Fig. 3.19 shows the sample quality of LeCun initialization with different scaling on the discriminator. We see that there are at least two good modes of scaling: one at around 0.2 and the other at around 1.2. This phenomenon cannot be explained by the analysis in LeCun/Kaiming initialization.

Recall that SN has similar properties as LeCun initialization (§ 3.3.1). Interestingly, we see that SSN also has two good modes of scaling (Fig. 3.19). Although the best scaling constants for LeCun initialization and SN are very different, there indeed exists an interesting mode correspondence in terms of parameter variances. We hypothesize that the shift of good scaling from Kaiming initialization we see here could result from adversarial training, and defer the theoretical analysis to future work. These results highlight the need

	CIFAR10 Dataset		STL10 Dataset		CelebA Dataset	ILSVRC2012 Dataset	
	IS ↑	FID ↓	IS ↑	FID ↓	FID ↓	IS ↑	FID ↓
Real data	11.26	9.70	26.70	10.17	4.44	197.37	15.62
SN	7.12 ± 0.07	31.43 ± 0.90	9.05 ± 0.05	44.35 ± 0.54	9.43 ± 0.09	12.84 ± 0.33	75.06 ± 2.38
SSN	7.38 ± 0.06	29.31 ± 0.23	**9.28 ± 0.03**	43.52 ± 0.26	**8.50 ± 0.20**	12.84 ± 0.33	73.21 ± 1.92
BSN	**7.54 ± 0.04**	**26.94 ± 0.58**	9.25 ± 0.01	42.98 ± 0.54	9.05 ± 0.13	1.77 ± 0.13	265.20 ± 19.01
BSSN	**7.54 ± 0.04**	**26.94 ± 0.58**	9.25 ± 0.01	**42.90 ± 0.17**	9.05 ± 0.13	**13.23 ± 0.16**	**69.04 ± 1.46**

Table 3.3: Inception score (IS) and FID on CIFAR10 Dataset, STL10 Dataset, CelebA Dataset, and ILSVRC2012 Dataset. The last three rows are proposed in this work, with BSSN representing our final proposal—a combination of BSN and SSN. Each experiment is conducted with 5 random seeds except that the last three rows on ILSVRC2012 Dataset is conducted with 3 random seeds. Mean and standard error across these random seeds are reported. We follow the common practice of excluding IS in CelebA Dataset as the inception network is pretrained on ImageNet, which is different from CelebA. Bold font marks best numbers in a column.

for a separate scaling factor.

(3) The bounds in Theorem 3.3.1.2 and Theorem 3.3.2.1 only imply that in SN and BSN the *order* of parameter variance w.r.t. the network size is correct, but constant scaling is unknown.

3.3.3 Experiments

In this section we verify the effectiveness of BSSN with extensive experiments. The code for reproducing the results is at https://github.com/fjxmlzn/BSN.

[43] already compares SN with many other regularization techniques like WGAN-GP [104], batch normalization [132], layer normalization [133], weight normalization [107], and orthogonal regularization [106], and SN is shown to outperform them all. Therefore, we compare BSSN with SN. To isolate the effects of the two components proposed in BSSN, we also compare against bidirectional normalization without scaling (BSN) and scaling without bidirectional normalization (SSN).

We conduct experiments across different public (non-sensitive) datasets (from low-resolution to high-resolution) and network architectures (from standard CNN to ResNets). More specifically, we conducts experiments on CIFAR10 Dataset, STL10 Dataset, CelebA Dataset, and ImageNet Dataest (ILSVRC2012 Dataset), following the same settings in [43]. The results are in Table 3.3.

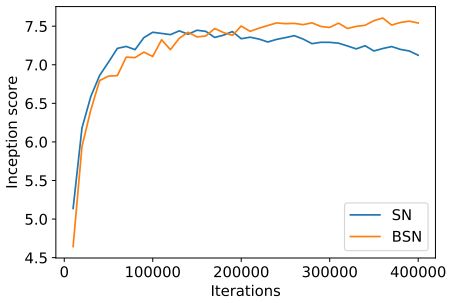


Figure 3.20: Inception score in CIFAR10 Dataset. The results are averaged over 5 random seeds, with $\alpha_g = 0.0001$, $\alpha_d = 0.0001$, $n_{dis} = 1$.

BSN v.s. SN (showing the effect of bidirectional normalization § 3.3.2): (1) By comparing BSN with SN in Table 3.3, we can see that BSN outperforms SN by a large margin in all metrics except in ILSVRC2012 Dataset (discussed later). (2) More importantly, the superiority of BSN is stable across hyper-parameters. We also vary the learning rates (α_g, α_d) and momentum parameters of generator and discriminator, and the number of discriminator updates per generator update (n_{dis}). We see that BSN consistently outperforms SN in most of the cases. (3) Moreover, BSN is more stable in the entire training process. We see that as training proceeds, the sample quality of SN often drops, whereas the sample quality of BSN appears to monotonically increase (Fig. 3.20). BSN generally outperforms SN in final sample quality (i.e., at the end of training), but also in *peak* sample quality. I.e., BSN stabilizes the training process, which is the purpose of SN (and BSN).

SSN v.s. SN (showing the effect of scaling § 3.3.2): Comparing SSN with SN in Table 3.3, we see that scaling consistently improves (or has the same metric) *in all cases*. This verifies our intuition in § 3.3.2 that the inherent scaling in SN is not optimal, and an extra constant scaling is needed for best results.

BSSN v.s. BSN (showing the effect of scaling § 3.3.2): By comparing BSSN with BSN in Table 3.3, we see that in some cases the optimal scale of BSN happens to be 1 (e.g., in CIFAR10 Dataset), but in other cases, scaling is critical. For example, in ILSVRC2012 Dataset, BSN without any scaling has the same gradient vanishing problem we observe for SN_{Conv} [48] in § 3.3.1, which causes bad sample quality. BSSN successfully solves the gradient vanishing problem and achieves the best sample quality.

Summary: In summary, both designs we proposed can effectively stabilize training and achieve better sample quality. Combining them together, BSSN achieves the best sample

quality in most cases. This demonstrates the practical value of the theoretical insights in § 3.3.1.

3.3.4 Discussions

In this section, we theoretically analyze why spectral normalization is able to stabilize GANs, and propose a simple replacement of SN named BSSN to further boost the training stability and sample fidelity.

Other reasons contributing to the stability of SN. This section presents one possible reason (i.e., SN avoids exploding and vanishing gradients), and shows such a correlation through theoretical and empirical analysis. However, there could exist many other parallel factors. For example, [43] points out that SN could speed up training by encouraging the weights to be updated along directions orthogonal to itself. Our results do not shed light on these orthogonal hypotheses.

Implications on other normalization/regularization techniques. As discussed in § 3.3.1, other normalization techniques like weight normalization [107] and orthogonal regularization [106] also control the maximum gradient norm, shown by a simple extension of our results. However, they perform worse in practice [43]. We hypothesize two reasons: (1) They may not have the more important properties proved in gradient vanish § 3.3.1. For example, the official implementation of orthogonal regularization on CNN kernels [134] gives a parameter variance of $\frac{1}{c_{in}k_w}$, which is larger than the one in $\mathrm{SN_w}$; (2) They have known drawbacks like promoting less effective features [43]. Extending our analysis framework to explain these differences would be an interesting future work.

Future directions. Our results suggest that SN stabilizes GANs by controlling exploding and vanishing gradients in the discriminator. However, our analysis also applies to the training of any feed-forward neural network. This may explain why SN also helps train generators [110, 101] and neural networks more broadly [48, 108, 121]. We focus on GANs because SN seems to disproportionately benefit them [43]. Carefully understanding why is an interesting direction for future work.

Related to the weight initialization and training dynamics, recent work [135, 136] has shown that Gaussian weights or ReLU activations cannot achieve dynamical isometry (all singular values of the network Jacobian near 1), a desired property for training stability. Orthogonal weight initialization may be better at achieving the goal. We considered Gaussian weights and ReLU activations as they are the predominant implementations in

GANs, but studying other networks may be useful too.

3.4 Chapter Summary

In this chapter, we study the two biggest fidelity issues in GANs: *mode collapse* and *training instability*. Regarding *mode collapse*, we propose a theoretical framework for analyzing the issue by drawing connections to hypothesis testing. Based on the theoretical insights, we propose *PacGAN*, a simple and effective framework for alleviating mode collapse. Regarding *training instability*, we provide a theoretical analysis of why spectral normalization [43], a widely used heuristic is so successful in stabilizing training. Based on the theoretical insights, we propose *Bidirectional Scaled Spectral Normalization (BSSN)*, a simple weight normalization technique to further improve training stability and sample fidelity.

Broader impacts. As fundamental improvements to GANs, the two algorithms proposed in this chapter (PacGAN and BSSN) are not only useful in data sharing applications, but are also generally applicable in all other GAN applications for boosting the sample fidelity. The theoretical insights (e.g., the effect of spectral normalization § 3.3.1) could be useful for other neural networks, too.

Chapter 4

Privacy Foundations

In this chapter, we first provide an overview of the fundamental privacy issues of GANs and introduce the privacy definitions and metrics (§ 4.1). We then present our theoretical analysis and algorithmic solutions for two important privacy notations: sample-level privacy (§ 4.2) and distributional privacy (§ 4.3). We conclude this chapter in § 4.4.

4.1 Overview of the Privacy Challenges

Since GANs are a relatively new class of models, the privacy properties of GANs have not been extensively studied, and how to make GANs more privacy-preserving is still an open question. In this section, we review the two most important privacy notations: *sample-level privacy* and *distributional privacy*.

Sample-level privacy. Sample privacy is a concern not only for GANs, but for all algorithms that interact with data. To provide a simplified mental model, imagine a table where each row corresponds to a single sample (e.g., a patient's medical information), and each column corresponds to a feature (e.g., gender, disease). Sample-level privacy describes how well the output of an algorithm (in the case of GANs, synthetic data) protects the information about individual samples (rows). In other words, if an attacker can infer information about a single sample in the original data by observing the output of the algorithm, then the algorithm has poor sample-level privacy. To evaluate sample-level privacy, there are two main approaches, which we discuss next.

- *Differential privacy.* Differential privacy (DP) has become the *de facto* formal privacy definition for sample-level privacy in many applications [137, 3]. We say that two databases D_0 and D_1 are *neighboring* if they differ in at most one element (row). A mechanism M is (ϵ, δ)-differentially-private [137, 3] if for any neighboring database D_0 and D_1, and any set $S \subseteq range\,(M)$,

$$\mathbb{P}\left[M(D_0) \in S\right] \leq e^{\epsilon}\mathbb{P}\left[M(D_1) \in S\right] + \delta.$$

Another stronger notion of differential privacy is called *probabilistic differential privacy*. A mechanism M is (ϵ, δ)-probabilistically-differentially-private [138] if for any neighboring database D_0 and D_1, there exists sets $S_0 \subseteq range\,(M)$ where $\mathbb{P}\left[M(D_0) \subseteq S_0\right] \leq \delta$, such that for any set $S \subseteq range\,(M)$

$$\mathbb{P}\left[M(D_0) \in S \setminus S_0\right] \leq e^{\epsilon}\mathbb{P}\left[M(D_1) \in S \setminus S_0\right].$$

This says that $(\epsilon, 0)$-differential-privacy condition holds except over a region of the support with probability mass at most δ.

It is straightforward to show that (ϵ, δ)-probabilistically-differential-privacy implies (ϵ, δ)-differential-privacy. In fact, probabilistic differential privacy is *strictly* stronger than differential privacy. To see this, consider the following example: assume $\mathbb{P}_p, \mathbb{P}_q$ are the distribution function of $M(D_0)$ and $M(D_1)$ respectively. When $e^{\epsilon}(1-\delta) < 1$ and $\delta > 0$, let $\epsilon' = \min\left\{1 - e^{\epsilon}(1-\delta), \delta\right\}$, we can construct the following $\mathbb{P}_p, \mathbb{P}_q$: $\mathbb{P}_q\,(0) = 0, \mathbb{P}_p\,(0) = 1 - (1-\epsilon')e^{-\epsilon}$ and $\mathbb{P}_q\,(1) = 1, \mathbb{P}_p\,(1) = (1-\epsilon')e^{-\epsilon}$. Then M satisfies (ϵ, δ)-differential-privacy, but does not satisfy (ϵ, γ)-probabilistically-differential-privacy for any $\gamma < 1$.

- *Membership inference attacks.* Membership inference attacks are closely related to differential privacy. Given a trained model, a membership inference attack aims to infer whether a given sample x was in the training dataset or not. The main difference between membership inference and differential privacy is that the attacker in differential privacy is assumed to know an adversarially-chosen pair of candidate training databases, whereas in membership inference attacks, the adversary is typically given access only to test samples (of which some are training samples) and the model. In some cases, the attacker is also given side information about the number of training samples in the test set. Hence, in general, the attacker in membership inference is neither strictly weaker nor strictly stronger than the differential privacy attacker.

  There have been many membership inference attacks proposed for discriminative models [139, 140, 141, 142, 143, 144, 145] and generative models (including GANs) [146, 147, 148]. Therefore, understanding robustness of GANs to membership inference attacks is important.

With the above setup in mind, we seek to answer the following questions: *(1) What are the sample-level privacy guarantees of GANs? (2) If these guarantees are not sufficient, how can we make GANs more privacy-preserving with respect to sample-level information?*

Distributional privacy. Distributional privacy is another important privacy concern, particularly in data sharing scenarios. It refers to the issue that the *distributional* properties of shared data may leak sensitive information (e.g., the mean of a data column). For example, a video content provider that shares video session data may wish to hide the total or mean traffic volume, which could be used to infer the company's total revenue [57]. A cloud provider that shares cluster performance traces may not want to reveal the proportions of different server types that it owns, which are regarded as business secrets [12]. At the same time, the cloud provider may also not want to release the system usage information faithfully, as maximum value of memory usage, for example, is close to the memory size of the system, which can be used by adversaries to launch attacks (e.g., denial-of-service attacks). Note that this information (total/mean traffic volume, proportions of server types, maximum or quantiles of system usage) cannot be inferred from any single record, but is inherent to the overall data distribution (or the aggregate dataset). This means that distributional privacy is orthogonal to sample-level privacy. For example, a video content provider has a dataset of daily page views that they want to release, and they are concerned about the *mean* page views (as it implies the revenue). A typical DP algorithm for protecting sample-level privacy [149] would add Gaussian or Laplace noise to the page view values. This process does not change the *mean* of the entire data on expectation. In fact, DP techniques are not designed to protect distributional attributes at all (they are designed to preserve them).

Unlike sample-level privacy, which has well-established frameworks for quantification and evaluation (e.g., differential privacy, membership inference attacks), distributional privacy is less studied in the community. Therefore, we want to explore: *(1) Can we develop a privacy framework for analyzing, evaluating, and protecting distributional privacy concerns? (2) Can we make GANs more privacy-preserving regarding distributional information?*

4.2 Protecting Sample-Level Privacy

In this section, we first show that GANs have inherent sample-level privacy guarantees, but these guarantees are not strong enough to provide meaningful protection (§ 4.2.1). This motivates us to use DP-SGD, a general training method for making neural networks differentially private. However, we observe that DP-SGD sacrifices data fidelity too much (§ 4.2.2). To address this challenge, we propose pretraining GANs on public data and

then fine-tuning them on private data using DP-SGD. We show that this method has the potential to achieve better tradeoff between fidelity and privacy (§ 4.2.3). Finally, we summarize our contributions and discuss future work (§ 4.2.4).

4.2.1 GANs' Inherent Privacy Guarantees

In this section, we will analyze the sample-level privacy guarantees of GANs when trained without any special mechanisms in place. We will examine both sample-level privacy notations discussed in § 4.1: differential privacy and robustness to membership inference attacks.

Differential Privacy Guarantees

We start with some notation, definitions, and assumptions. For a probability measure μ on $\mathcal{X}$, we let ρ_μ denote its density function. We use $\mathcal{G}$ and $\mathcal{D}$ to denote the set of possible generators and discriminators, respectively, where $\mathcal{D}$ is a set of functions $\mathcal{X} \to \mathbb{R}$. Our results rely on three assumptions:

(A1) Our generator set $\mathcal{G}$ and discriminator set $\mathcal{D}$ satisfy $\forall \nu_1, \nu_2 \in \mathcal{G}, \log\left(\rho_{\nu_1}/\rho_{\nu_2}\right) \in span\mathcal{D}$, where $span\mathcal{D}$ is defined as the set of linear combinations of the functions:

$$span\mathcal{D} \triangleq \left\{ w_0 + \sum_{i=1}^{n} w_i f_i : w_i \in \mathbb{R}, f_i \in \mathcal{D}, n \in \mathbb{N} \right\}.$$

(A2) Our discriminator set $\mathcal{D}$ is even, i.e, $\forall f \in \mathcal{D}$, $-f \in \mathcal{D}$, and assume that $\forall f \in \mathcal{D}$, $\forall x \in \mathcal{X}$,

$$\|f(x)\|_\infty \leq \Delta,$$

where $\|f\|_\infty \triangleq \sup_{x \in \mathcal{X}} |f(x)|$.

(A3) The discriminator set $\mathcal{D} = \left\{ f_\eta : \eta \in H \subseteq [-1,1]^{d_2} \right\}$ and

$$\left\| f_\eta - f_{\eta'} \right\|_\infty \leq L \left\| \eta - \eta' \right\|_2.$$

According to Lemma 4.1 in [150], when generators in $\mathcal{G}$ are invertible neural networks (e.g., in [151, 152]) with l layers, then the discriminator set $\mathcal{D}$ of neural networks with $l+2$

layers satisfy (A1). Assumption (A2) is easily satisfied by neural networks with an activation function on the output layer that bounds the output (e.g., sigmoid). (A3) assumes the discriminator is Lipschitz in its (bounded) parameters; several recent works attempt to make network layers Lipschitz in inputs through various forms of regularization (e.g., spectral normalization [43, 153]), which makes the network Lipschitz also in parameters [153].

Given μ, ν, two probability measures on $\mathcal{X}$, and set $\mathcal{D}$ of functions $\mathcal{X} \to \mathbb{R}$, the *integral probability metric* [154] is defined as:

$$d_{\mathcal{D}}(\mu \| \nu) \triangleq \sup_{f \in \mathcal{D}} \left\{ \mathbb{E}_{x \sim \mu}[f(x)] - \mathbb{E}_{x \sim \nu}[f(x)] \right\}.$$

Given function set $\mathcal{D}$, the *$\mathcal{D}$-variation norm* of function g is defined as

$$\|g\|_{\mathcal{D},1} \triangleq \inf \left\{ \sum_{i=1}^{n} |w_i| : g = w_0 + \sum_{i=1}^{n} w_i f_i, \forall n \in \mathbb{N}, w_i \in \mathbb{R}, f_i \in \mathcal{D} \right\},$$

which intuitively describes the complexity of linearly representing g using functions in $\mathcal{D}$ [155]. We define

$$\Gamma_{\mathcal{D},\mathcal{G}} \triangleq \sup_{\nu_1, \nu_2 \in \mathcal{G}} \left\| \log\left(\rho_{\nu_1} / \rho_{\nu_2}\right) \right\|_{\mathcal{D},1}, \tag{4.1}$$

which intuitively bounds the complexity of representing differences in log densities of pairs of generators in $\mathcal{G}$ using functions in $\mathcal{D}$.

We consider the GAN training and sampling mechanism in Algorithm 4.2.1, which adds a sampling processing before the normal GAN training. Note that the sampling process in Algorithm 4.2.1 is commonly used in existing GAN implementations [1], which typically sample i.i.d. from D in each training batch. We have moved this sampling process to the beginning of training in Algorithm 4.2.1 for ease of analysis.

Finally, we define the quantity

$$\tau_{\mathcal{F},\mathcal{G}}(k, \xi, \mu) \triangleq 2 \left(\inf_{\nu \in \mathcal{G}} d_{\mathcal{D}}(\mu \| \nu) + \tau_{\text{opt}} + \frac{C_\xi}{\sqrt{k}} \right), \tag{4.2}$$

Algorithm 4.2.1: Differentially-private GAN mechanism.

Input : D_{original}: A training dataset containing m samples.
k: Number of sampled training samples used in training.
n: Number of generated samples.

Output: $D_{\text{generated}}$: n generated samples.

1 $D_{\text{train}} \leftarrow k$ i.i.d. samples from D_{original} ;
2 $q_\theta \leftarrow$ Trained GAN using D_{train};
3 $D_{\text{generated}} \leftarrow n$ generated samples from the trained q_θ;

where

$$C_\xi = 16\sqrt{2\pi} d_2 L + 2\Delta\sqrt{2\log(1/\xi)}, \tag{4.3}$$

d_2 and L are the constants defined in (A3), Δ is the constant defined in (A2), τ_{opt} is an upper bound on the optimization error, i.e.,

$$d_{\mathcal{D}}\left(p_{\hat{X}k} \| q_\theta\right) - \inf_{\nu \in \mathcal{G}} d_{\mathcal{D}}\left(p_{\hat{X}k} \| \nu\right) \leq \tau_{\text{opt}},$$

p_X is the real distribution, $p_{\hat{X}k}$ be the empirical distribution of p_X on k i.i.d training samples, q_θ is the trained generator from the optimization algorithm. The interpretation of $\tau_{\mathcal{F},\mathcal{G}}\left(k, \xi, p_X\right)$ is described in greater detail in Appendix B.1.1, but intuitively, it will be used to bound a GAN's generalization error arising from approximation, optimization, and sampling of the training datasets in Line 1. To reduce notation, we will henceforth write this quantity as $\tau_{k,\xi}$.

Our main result states that a GAN trained on m samples and used to generate n samples satisfies a $\left(\epsilon, \frac{O(n/m)}{\epsilon(1-e^{-\epsilon})}\right)$-differential-privacy guarantee; moreover, this bound is tight when $n = 1$ and for small ϵ.

Theorem 4.2.1.1 (Achievability)**.** *Consider a GAN trained on m i.i.d. samples from distribution p_X. The mechanism in Algorithm 4.2.1 under assumptions (A1)-(A3) satisfies (ϵ, δ)-differential privacy for any $\epsilon > 0$, $\xi > 0$, and*

$$\delta \quad > \frac{n\ \Gamma_{\mathcal{D},\mathcal{G}}}{\epsilon(1-e^{-\epsilon})} \left(\frac{2\Delta}{m} + \tau_{k,\xi}\right) + 2\xi\ . \tag{4.4}$$

We assume that the terms regarding ξ (2ξ and $\tau_{k,\xi}$) can be made negligible: for any

$\xi > 0$, $\frac{C_\xi}{\sqrt{k}}$ can be arbitrarily small as we get more samples from the sampling phase in Line 1, and we assume negligible approximation error and optimization error. Hence, the dominating term in Eq. (4.4) scales as $O(n/m)$; here we are ignoring the dependency on ϵ. Next, we show that for a special case where $n = 1$ and ϵ scales as $O\left(\frac{1}{m}\right)$, this bound is tight in an order sense (again ignoring dependencies on ϵ).

Proposition 4.2.1.1 (Converse for $n = 1$)**.** *Under the assumptions of Theorem 4.2.1.1, let*

$$\Delta' = \sup_{f \in \mathcal{D}} \sup_{x,y \in \mathcal{X}} |f(x) - f(y)|.$$

Then with probability at least $1-2\xi$*, the GAN mechanism in Algorithm 4.2.1 for generating 1 sample (i.e.,* $n = 1$*) does not satisfy* (ϵ, δ)*-differential-privacy for any*

$$\delta < \frac{(e^\epsilon + 1)}{2\Delta} \left(\frac{\Delta'}{2m} - \tau_{k,\xi} \right) + 1 - e^\epsilon. \tag{4.5}$$

This bound in Eq. (4.5) is non-vacuous (nonnegative) when $\epsilon < \frac{1}{\Delta}\left(\frac{\Delta'}{2m} - \tau_{k,\xi}\right)$ and $m < \Delta'/2\tau_{k,\xi}$. Again assuming $\tau_{k,\xi} \approx 0$ (i.e., ignoring the approximation error and optimization error and taking $\frac{C_\xi}{\sqrt{k}} \to 0$), the latter condition holds trivially. Hence when ϵ scales as $O\left(\frac{1}{m}\right)$, it is not possible to achieve an (ϵ, δ)-probabilistic differential privacy guarantee for $\delta = o\left(\frac{1}{m}\right)$.

Discussions. These results suggest that GAN-generated samples satisfy an inherent differential privacy guarantee, so the influence of any single training sample on the final generated samples is bounded. However, the rate $O(n/m)$ is weak. For comparison, the mechanism that releases n samples uniformly at random from a set of m training samples satisfies $(0, n/m)$-differential privacy, which is of the same rate. Therefore, (ϵ, δ)-differential privacy usually requires $\delta \ll \frac{1}{\text{poly}(m)}$ to be meaningful. To satisfy this condition, we would need ϵ to grow as a function of m (since δ in our results is a function of ϵ), which is not practically viable. This suggests the need for incorporating additional techniques (e.g., DP-SGD) to achieve stronger differential privacy guarantees in practice. This is what we will explore in the next sections.

Robustness to Membership Inference Attacks

In this section, we first derive a general bound on the error of membership inference attacks for generative models. Then, we utilize generalization bounds for GANs to obtain specific

bounds for GANs.

We focus on the black-box attack setting [139, 147, 146], in which the attacker can sample from the generated distribution q_θ, but does not have access to the generator parameters θ. To upper bound the attack performance, we assume that attacker has unlimited resources and can access the trained generator infinite times, so that it can accurately get the generated distribution q_θ.

Our analysis departs from prior analysis of membership inference in discriminative models [139] in two key respects:

- [139] assume that the attacker has access to a dataset $U = \{u_1, \ldots, u_p\}$, which contains all the training samples (i.e., $D_{\text{original}} \subseteq U$) and some other test samples drawn from the ground-truth distribution p_X (so $p > m$). It also assumes that the attacker knows the number of training samples m. We argue that this assumption is too strong, especially in the case of generative models, where training samples are typically proprietary. Therefore, we assume that the attacker makes guesses purely based on a single test sample $x \in \mathcal{X}$, without access to such a dataset. The test sample is either drawn from the ground-truth distribution (i.e., $x \sim p_X$), or from the training dataset (i.e., $x \xleftarrow{\text{i.i.d.}} D_{\text{original}}$).
- The analysis in [139] focuses on the quantity $\mathbb{P}(u \in D_{\text{original}}|\theta)$ for a particular u. This is useful for finding an attack policy, but is not conducive to characterizing the error statistics. Instead, we want to be able to bound the shape of ROC curve. That is, we want to upper bound the true positive rate an attacker can achieve given any desired false positive rate. We show that this problem can be reduced to a clean hypothesis testing problem, whose errors are closely tied to the generalization errors of GANs.

Following prior work on the theoretical analysis of membership inference attacks [139], we assume that the distribution of the generator parameters is

$$\mathbb{P}(\theta|x_1, ..., x_m) \propto e^{-\sum_{i=1}^{m} \ell(\theta, x_i)} \tag{4.6}$$

(setting $T = 1$ in [139]), where $x_1, ..., x_m \sim p_X$ are i.i.d training samples drawn from the ground-truth distribution p_X, and $\ell(\theta, x)$ denotes the loss on sample x and parameter θ. As we are focusing on generative models here, we assume that the loss is Kullback-Leibler (KL) divergence, i.e., $\ell(\theta, x) = \log\left(\rho_{p_X}(x_i)/\rho_{q_\theta}(x_i)\right)$, where ρ_{q_θ} denotes the density of the generator with parameters θ. Note that many generative models are explicitly or implicitly minimizing this KL divergence, including some variants of GANs (more specifically, f-GANs with

a specific loss [156]), Variational Autoencoder (VAE) [71], PixelCNN/PixelRNN [157], and many other methods that are based on maximum likelihood [158]. We use KL divergence also because this simplifies the analysis and highlights key theoretical insights. With this assumption, the parameter distribution becomes

$$\mathbb{P}\left(\theta|x_1,...,x_m\right) \propto \prod_{i=1}^{m} \frac{\rho_{q_\theta}\left(x_i\right)}{\rho_{p_X}\left(x_i\right)} \ .$$

Let $\rho^{\theta}_{\text{train}}$ denotes the density posterior distribution of the training samples given parameter θ. The following proposition shows that this distribution takes a simple form.

Proposition 4.2.1.2 (Posterior distribution of training samples)**.** *The posterior distribution of training samples is equal to the generated distribution, i.e.,* $\rho^{\theta}_{train} = \rho_{q_\theta}$.

This proposition validates prior membership inference attacks that utilize approximations of $\rho_{q_\theta}\left(x\right)$ to make decisions [147, 146, 148].

With this proposition, the problem becomes clear: for a given sample x, the attacker needs to decide whether the sample comes from the training set (i.e., from q_θ) or not (i.e., from p_X). In the following theorem, we outer bound the ROC region for this hypothesis test, which relates the true positive (TP) rate to the false positive (FP) rate.

Proposition 4.2.1.3. *Consider a generative model* q_θ *and a real distribution* p_X*. Define* $r \triangleq d_{\text{TV}}\left(q_\theta \| p_X\right)$ *as the total variation (TV) distance between the two distributions. Define function* $f : [0,1] \to [0,1]$ *as*

$$f(x) = \begin{cases} x + r & (0 \le x \le 1 - r) \\ 1 & (r < x \le 1) \end{cases} ,$$

Then we have that for any membership inference attack policy $\mathcal{A}$*, the ROC curve* $g_{\mathcal{A}} : [0,1] \to [0,1]$ *(mapping FP to TP) satisfies* $g(x) \le f(x), \forall\ 0 \le x \le 1$*, i.e., the ROC curve is upper bounded by* f*. Also, this bound is tight, i.e., there exists two distributions* p'_X, g' *such that* $d_{\text{TV}}\left(p'_X \| g'\right) = r$ *and the ROC curve is exactly* f *at every point.*

As a result, we can directly bound the area under the ROC curve (AUC) as a function of the total variation distance.

Corollary 4.2.1.1 (Bound on the AUC for generative models)**.** *For any attack policy on a generative model, we have*

$$\mathrm{AUC} \leq -\frac{1}{2} d_{\mathrm{TV}}\left(q_\theta \| p_X\right)^2 + d_{\mathrm{TV}}\left(q_\theta \| p_X\right) + \frac{1}{2}.$$

Note that Proposition 4.2.1.3 and Corollary 4.2.1.1 hold for any generative model. For GANs in particular, we can use generalization bounds in Lemma B.1.1.1 to obtain the following result.

Theorem 4.2.1.2. *Consider a GAN model q_θ and a real distribution p_X. Define*

$$\Xi_{\mathcal{F},\mathcal{G},p_X} \triangleq \sup_{\nu \in \mathcal{G}} \left\| \log\left(\rho_{p_X} / \rho_\nu\right) \right\|_{\mathcal{D},1}$$

and

$$\epsilon_{\mathrm{TV}}\left(m, \delta\right) \triangleq \frac{\sqrt{\Xi_{\mathcal{F},\mathcal{G},p_X} \cdot \tau_{m,\delta}}}{2\sqrt{2}}, \tag{4.7}$$

where $\tau_{m,\delta}$ is defined as in Eq. (4.2). Define function $f : [0,1] \to [0,1]$ as

$$f(x) = \begin{cases} x + \epsilon_{\mathrm{TV}}\left(m, \delta\right) & \left(0 \leq x \leq 1 - \epsilon_{\mathrm{TV}}\left(m, \delta\right)\right) \\ 1 & \left(\epsilon_{\mathrm{TV}}\left(m, \delta\right) < x \leq 1\right) \end{cases}.$$

Then we have that for any membership inference attack policy $\mathcal{A}$, the ROC curve $g_{\mathcal{A}} : [0,1] \to [0,1]$ satisfies $g(x) \leq f(x), \forall\ 0 \leq x \leq 1$, and the bound is tight.

One complication is that existing generalization bounds do not directly bound TV distance, so these must be extended. It directly gives the following corollary bounding the AUC for GANs.

Corollary 4.2.1.2 (Bound on AUC for GANs)**.** *For any attack policy on GANs, we have that with probability at least $1 - \delta$ w.r.t. the randomness of training samples,*

$$\mathrm{AUC} \leq -\frac{1}{2} \epsilon_{\mathrm{TV}}\left(m, \delta\right)^2 + \epsilon_{\mathrm{TV}}\left(m, \delta\right) + \frac{1}{2},$$

where $\epsilon_{\mathrm{TV}}\left(m, \delta\right)$ is defined as in Eq. (4.7).

Note that the AUC bound decays as $O(m^{-1/4})$.

Discussions. These results confirm the prior empirical observation that GANs are more robust to membership inference attacks when the number of training samples grows [12, 147]. However, the results heavily rely on the assumption of generator parameter distribution Eq. (4.6), which was introduced in [139]. It is unlikely to strictly hold in practice. Extending the results to more general settings would be an interesting future direction.

4.2.2 Challenge: DG-SGD Gives Bad Fidelity-Privacy Tradeoff on GANs

The above section shows that the inherent privacy guarantees of GANs are not sufficient to provide reasonable guarantees. Therefore, we need to apply additional mechanisms to ensure sample-level privacy. One common way to make neural network models differentially private is to replace the optimizer with DP-SGD algorithm []. On the high-level, DP-SGD algorithm computes per-sample gradients, clip them, and add Gaussian noise to their sum. Theoretically, this process would make sure that the trained GAN model and also its generated samples are differentially private.

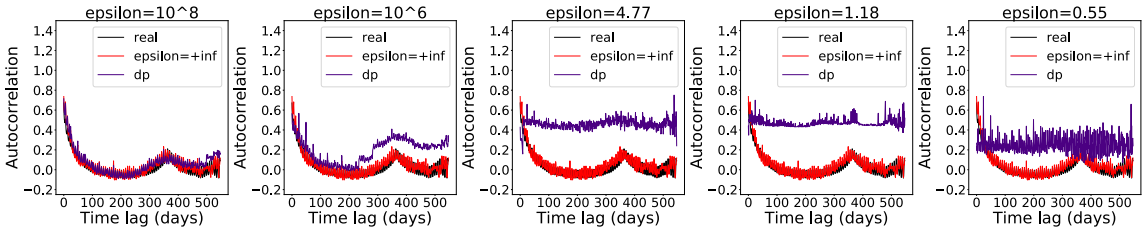


Figure 4.1: Autocorrelation for real, $\epsilon = +\inf$, and DP-GANs with different values of ϵ).

However, empirically we observe that DG-SGD algorithm hurts fidelity too much. We conduct experiments on training GANs on Wikipedia Web Traffic Dataset (WWT) [160], which contains the daily page views of 145,063 Wikipedia web pages in 2015-2016. The blue line in Fig. 4.1 shows the average autocorrelation of all samples in the original data. We see that the original data has weekly and annual correlations. The red line shows the results of GANs without DP-SGD. We can see that GANs can capture both patterns well. The purple lines are from GANs trained with DP-SGD with different DP privacy budget, ϵ. Smaller values of ϵ denote more privacy; $\epsilon \approx 1$ is typically considered a reasonable operating point. We can see that GANs with DP-SGD only work well with very large epsilon values (less private). As ϵ is reduced (stronger privacy guarantees),

autocorrelations become progressively worse. When epsilon is around 1, the GANs are not able to capture the patterns well. One reason for the bad results could be that the GANs are already hard to train (§ 3.3). The added gradient noise from DP-SGD could make the training process more unstable, leading to poor generated samples.

4.2.3 Public Pretraining for Improving Fidelity-Privacy Tradeoff

Observing that public datasets often contain similar patterns as private data, we propose a two-step approach for training GANs on private data. First, we pretrain GANs on public data without using DP-SGD. This allows GANs to learn the common patterns present in both public and private data. Next, we fine tune GANs on private data with DP-SGD. Because GANs have already learned relevant patterns from the public data, they should require fewer DP-SGD iterations to research the desired fidelity level. This can improve fidelity-privacy tradeoff, as the privacy cost (ϵ) increases with the number of DP-SGD iterations.

To verify the idea, we conduct the experiments on (1) CAIDA datasets [7] which contain anonymized traces from high-speed monitors on a commercial backbone link, and (2) Data Center dataset, which is a packet capture from the "UNI1" data center studied in the IMC 2010 paper [161]. Privacy is measured by DP parameter ϵ (we fix $\delta = 10^{-5}$), and fidelity is measured as the mean JSD across all fields in the data (smaller is better). Figure 4.2 shows the results. We see that public pretraining do help the fidelity-privacy tradeoff, but this gain only holds when the public dataset is similar to the private data. When the model is pretrained on a public header trace from a different domain, the privacy-fidelity tradeoff is the same as training from scratch. For example, the 'DP Pretrained-SAME' curve was pretrained on a CAIDA dataset from the Chicago collector in March 2015, and finetuned on "private" CAIDA dataset (New York). Although these datasets likely see different traffic patterns, they are from the same domain, and we observe gains in privacy-fidelity tradeoff. In contrast, the 'DP Pretrained-DIFF' curve was pretrained on the Data Center dataset, and pretraining gives virtually no benefit. This suggests that pretraining can be effective, but care must be taken to select sufficiently close pretraining public datasets.

Note that this idea of public pretraining has been explored in related work from the DP community [162, 163, 164, 165], but it has not been utilized in GANs to the best of our knowledge.

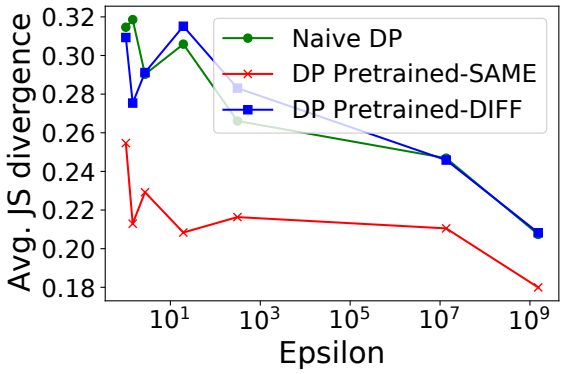


Figure 4.2: Privacy-fidelity tradeoffs: Privacy is measured with (ϵ, δ) in DP (↓) and fidelity is measured as average JSD across metrics (↓).

4.2.4 Discussions

In this section, we provide the theoretical analysis of the sample-level privacy guarantees of GANs, and show that pretraining GANs on public data is a promising approach for improving fidelity-privacy tradeoff.

Limitations and future work. (1) Our theoretical analysis (§ 4.2.1) builds on existing generalization bounds for GANs [155]. Therefore, we inherit their assumptions (e.g., on the classes of generators and discriminators). Some of these assumptions do not apply to all GAN architectures. Extending the results to more general settings would be an interesting direction, potentially with stronger generalization bounds. (2) As discussed in § 4.2.1, the results on membership inference attacks rely on an assumption from [139] regarding the generator parameter distribution; this assumption is unlikely to hold in practice. Relaxing this assumption is an interesting direction for future work. (3) The bounds we give for both privacy notions depend on unknown constants like optimization and approximation errors. Numerically quantifying these bounds in practice remains a challenging and interesting direction. (4) This dissertation focuses on GANs. Similar techniques in our theoretical analysis (§ 4.2.1) could be used to analyze other types of generative models. This could be an interesting direction for future work.

4.3 Protecting Distributional Privacy

In this section, we first present a theoretical framework for analyzing distribution privacy (§ 4.3.1). At a high level, the proposed framework works as follows. A data holder first chooses one or more secrets, which are mathematically defined as functions of the data holder's data distribution. For example, a video analytics company might choose the mean daily observed traffic as a secret quantity. Then, the data holder obfuscates their data according to some *mechanism* and releases the output. Our framework quantifies the *privacy* of this mechanism by analyzing the probability that an attacker can infer the data holder's true secret after observing the output. To capture the utility of released data, we define the *distortion* of a mechanism as the worst-case distance (where the distance metric can be chosen by the data holder or data user) between the original and released data distributions. Our goal is to design data release mechanisms that control tradeoffs between privacy and distortion. Based on this framework, we present a theoretical analysis of the achievable privacy-distortion tradeoffs (§ 4.3.2) and a template of data release mechanisms (§ 4.3.3). We then apply these results to several concrete examples of secrets (§ 4.3.4), and present experimental results on real-world datasets (§ 4.3.5). Finally, we summarize the contributions and discuss future work (§ 4.3.6).

4.3.1 Theoretical Framework for Distributional Privacy

Notation. Random variables are denoted with uppercase English letters or upright Greek letters (e.g., $X, \upmu$), and their realizations are denoted with italicized lowercase letters (e.g., x, μ). For a random variable X, we denote its probability density function (PDF), or, in the case of discrete random variables, its probability mass function (PMF), as f_X, and its distribution measure as p_X. If a random variable X is drawn from a parametric family (e.g., X is Gaussian with specified mean and covariance); the parameters are denoted with a subscript of X, i.e., the above notations become X_θ, f_{X_θ}, p_{X_θ} respectively for parameters $\theta \in \mathbb{R}^q$, where $q \geq 1$ denotes the dimension of the parameters. In addition, the conditional PDF/PMF of X given another random variable Y is denoted as $f_{X|Y}$. We use $\mathbb{Z}, \mathbb{Z}_{>0}, \mathbb{N}, \mathbb{R}, \mathbb{R}_{>0}$, to denote the set of integers, positive integers, natural numbers, real numbers, and positive real numbers respectively.

Fig. 4.3 shows the framework, which we will explain below.

Original data. Consider a data holder who possesses a dataset of n samples $\mathcal{X} =$

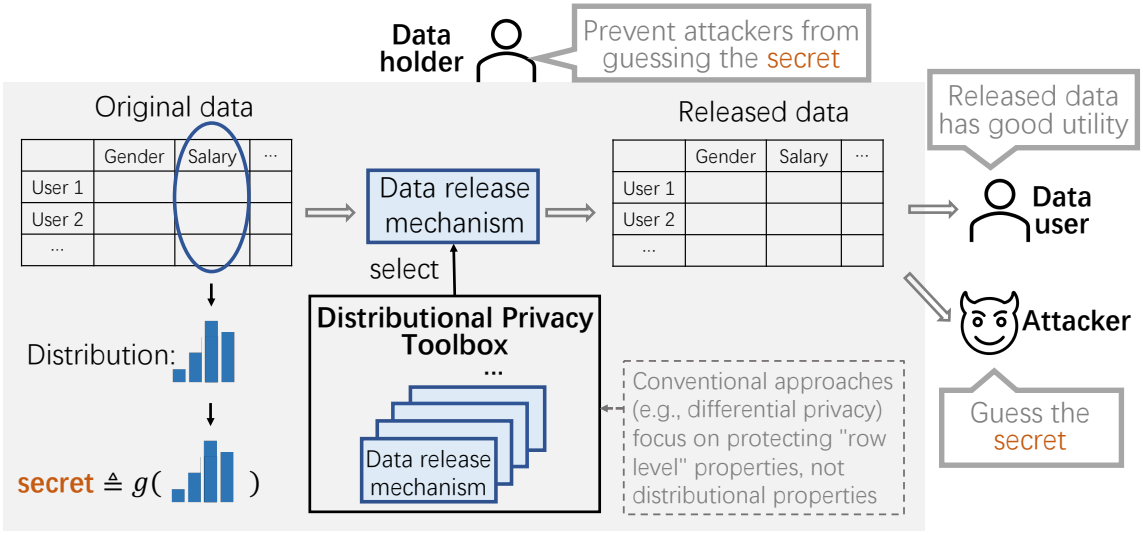


Figure 4.3: Problem overview. The data holder produces released data and wants to hide *distributional secrets* of the original data. The data user requires that the utility of the released data be good. The attacker, who could be the data user, also observes the released data, and wants to guess the *secrets* of the original data. Note that we focus on secrets about the *underlying distribution* (e.g., mean, quantile, standard deviation, of a specific data *column*), whereas many existing frameworks (e.g., differential privacy [3], anonymization [4], sub-sampling [4]) protect information from *individual samples (rows)*. Our goal is to provide a *distributional privacy toolbox* for data holders to use. The toolbox contains data release mechanisms for a set of pre-defined secrets and data distributions. Data holders can choose the mechanism according to the secret they want to hide and the closest data distributions.

$\{x_1, \ldots, x_n\}$, where for each $i \in [n]$, $x_i \in \mathbb{R}^p$ is drawn i.i.d. from an underlying distribution. We assume the distribution comes from a parametric family, and the parameter vector $\theta \in \mathbb{R}^q$ of the distribution fully specifies the distribution. That is, $x_i \sim p_{X_\theta}$, where we further assume that θ is itself a realization of random parameter vector Θ, and p_Θ is the probability measure for Θ. We assume that the data holder knows θ (and hence knows its full data distribution p_{X_θ}); our results and mechanisms can be generalized to the case where the data holder only possesses the dataset $\mathcal{X}$ (see § 4.3.4).

For example, suppose the original data samples come from a Gaussian distribution. We have $\theta = (\mu, \sigma)$, and $X_\theta \sim \mathcal{N}(\mu, \sigma)$. p_Θ (or f_Θ) describes the prior distribution over (μ, σ). For example, if we know a priori that the mean of the Gaussian is drawn from a uniform distribution between 0 and 1, and σ is always 1, we could have $f_\Theta(\mu, \sigma) =$

$\mathbb{I}(\mu \in [0,1]) \cdot \delta(\sigma)$, where $\mathbb{I}(\cdot)$ is the indicator function, and δ is the Dirac delta function. In practice, the underlying distribution can be much more complicated than a Gaussian.

In general, the data can be multi-dimensional (i.e., $p > 1$). We study one-dimensional data as a starting point.

Distributional secrets to protect. We assume the data holder wants to hide $\ell \in \mathbb{Z}_{>0}$ *secrets* from the original data distribution. Since the true data distribution is fully-specificed by parameter vector θ, these secrets can be expressed as a function $g(\theta) : \mathbb{R}^q \to \mathbb{R}^\ell$. In the Gaussian example $X_\theta \sim \mathcal{N}(\mu, \sigma)$, suppose the random variable X_θ represents the traffic volume experienced by an enterprise in a day. The data holder may wish to hide the mean traffic per day, in which case $g(\cdot)$ would be the mean of the distribution, i.e., $g(\mu, \sigma) = \mu$. In this example, we are hiding only one secret (the mean), so $\ell = 1$. *In general, the secret can be any (vector-valued) function that can be deterministically computed from* θ. As shown in Fig. 4.3, the secret could be derived from one feature (e.g., mean salary) or computed from multiple features (e.g., the mean salary of males). The secrets could also be multi-dimensional (e.g., mean salary and the fraction of males). In this section, we present general results for one-dimensional secrets (i.e., $\ell = 1$) and defer the discussion of higher-dimensional secrets to future work (see § 4.3.6).

Data release mechanism. The data holder releases data by passing the private parameter θ through a *data release mechanism* $\mathcal{M}_g$. That is, for a given θ, the data holder first draws internal randomness $z \sim p_Z$, and then releases another distribution parameter $\theta' = \mathcal{M}_g(\theta, z)$, where $\mathcal{M}_g$ is a deterministic function, and p_Z is a fixed distribution from which z is sampled. Note that we assume that both the input and output of $\mathcal{M}_g$ are distribution parameters, but it is straightforward to generalize to the case when the input and/or output are datasets of samples (see § 4.3.4).

For example, in the Gaussian case discussed above, the data release mechanism can be $\mathcal{M}_g((\mu, \sigma), z) = (\mu + z, \sigma)$ where $z \sim \mathcal{N}(0, 1)$. In other words, this mechanism shifts the mean of the Gaussian by a random amount drawn from a standard Gaussian distribution and keeps the variance unchanged.

Threat Model. We assume that the attacker knows the parametric family from which our data is drawn, but does not know the initial parameter θ. The attacker is also assumed to know the data release mechanism $\mathcal{M}_g$ and output θ' but not the realization of the data holder's internal randomness z. The attacker guesses the initial secret $g(\theta)$ based on the released parameter θ' according to estimate $\hat{g}(\theta')$. $\hat{g}$ can be either random or deterministic, and we assume that the adversary has no computational bounds. For instance, in the

running Gaussian example, an attacker may choose $\hat{g}(\mu', \sigma') = \mu'$. When the data holder releases a dataset of samples instead of the parameter θ', this formulation can be used to upper bound the attacker's accuracy in correctly guessing the secret, since the estimation error on released distribution parameter is induced due to the finite samples in the released dataset.

Privacy metric. The data holder wishes to prevent an attacker from guessing its secrets.

We define our privacy metric privacy Π_ϵ as the attacker's probability of guessing the secret(s) within a tolerance ϵ, in the worst-case over all attackers $\hat{g}$:

$$\Pi_\epsilon \triangleq \sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right) , \tag{4.8}$$

where the probability is taken over the randomness of the original data distribution ($\theta \sim p_\Theta$), the data release mechanism ($z \sim p_Z$), and the attacker strategy ($\hat{g}$).

Distortion metric. The main goal of data sharing is to provide useful data; hence, we (and data holders and users) want to understand how much the released data distorts the original data. We define the *distortion* Δ of a mechanism as the worst-case distance between the original distribution and the released distribution:

$$\Delta \triangleq \sup_{\substack{\theta \in \text{Supp}(p_\Theta), \theta', \\ z \in \text{Supp}(p_Z): \mathcal{M}_g(\theta, z) = \theta'}} d\left(p_{X_\theta} \| p_{X_{\theta'}}\right), \tag{4.9}$$

where d is a general distance metric defined over distributions. The choice of the distance metric depends on the data type and potentially on the applications that data holders and users care about. For example, if the data holders or users have concrete metrics that they want to preserve (e.g., the difference between the mean salaries of males and females in Fig. 4.3), they could use this quantity as the distance metric. Otherwise, one can use statistical distance metrics between distributions (e.g., total variation distance, Wasserstein distance). In this dissertation, we adopt Wasserstein-1 distance for continuous distributions and total variation (TV) distance for discrete distributions. These distances are often used for evaluating data quality (e.g., [6, 12]) and as the distance metric in neural network design (e.g., [103, 166]). Note that the definition in Eq. (4.9) can be extended to data release mechanisms that take datasets as inputs and/or outputs.

Objective. To summarize, the data holder's objective is to choose a data release mecha-

nism that minimizes the distortion metric Δ subject to a constraint on privacy Π_ϵ:

$$\min_{\mathcal{M}_g} \quad \Delta \qquad \text{subject to} \quad \Pi_\epsilon \leq T. \tag{4.10}$$

The optimal data release mechanisms for Eq. (4.10) depends on the secrets, the distance metric d used in Eq. (4.9), and the characteristics of the original data. In practice, we envision a *distributional privacy toolbox* (Fig. 4.3) that encodes data release mechanisms for a list of predefined secrets, d, and data distributions. Data holders would specify the secret function they want to protect and the desired distance metric; the toolbox would select the data distribution parametric family that most closely reflects the holder's raw data and use the corresponding data release mechanism to process the raw data for sharing.

4.3.2 Privacy-Distortion Tradeoffs

Given the privacy budget, we present a lower bound on distortion that applies *regardless of the prior distribution of data* p_Θ and *regardless of the secret* g. As discussed in § 4.3.1, we assume that the secret is scalar (i.e., $\ell = 1$).

Theorem 4.3.2.1 (Lower bound of privacy-distortion tradeoff)**.** *Let* $D\left(X_{\theta_1}, X_{\theta_2}\right) \triangleq \frac{1}{2} d\left(p_{X_{\theta_1}} \| p_{X_{\theta_2}}\right)$, *where* $d\left(\cdot\|\cdot\right)$ *is defined in Eq. (4.9). Further, let* $R\left(X_{\theta_1}, X_{\theta_2}\right) \triangleq \left|g(\theta_1) - g(\theta_2)\right|$. *Let* $\gamma \triangleq \inf_{\theta_1,\theta_2 \in Supp(p_\Theta)} \frac{D\left(X_{\theta_1}, X_{\theta_2}\right)}{R\left(X_{\theta_1}, X_{\theta_2}\right)}$. *For any* $T \in (0,1)$, *when* $\Pi_\epsilon \leq T$, *we have* $\Delta > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot 2\gamma\epsilon$.

From Theorem 4.3.2.1 we know that the lower bound of distortion is inversely correlated with the privacy budget and positively correlated with the guess tolerance ϵ. Note that we have not made the dependent quantity γ in the lower bound specific as its exact form depends on the type of the secret. We will instantiate it in the cases studies in § 4.3.4.

4.3.3 Data Release Mechanism Design

In this section, we present a general template of our data release mechanisms that we will use across all case studies in § 4.3.4. We divide the set of possible distribution parameters $\text{Supp}\left(\Theta\right)$ into subsets $\mathcal{S}_i$ such that $\cup_{i\in\mathcal{I}}\mathcal{S}_i \supseteq \text{Supp}\left(\Theta\right)$ and $\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2} = \emptyset$ for $i_1 \neq i_2$, where $\mathcal{I}$ is the set of indices of the subsets. For $\theta \in \text{Supp}\left(\Theta\right)$, $I\left(\theta\right)$ is the index of the set that θ belongs to; in other words, we have $I\left(\theta\right) = i$, where $\theta \in \mathcal{S}_i$. The mechanism works by looking up which set θ belongs to (i.e., $I\left(\theta\right)$), and *deterministically* releasing a parameter $\theta^*_{I(\theta)}$ that corresponds to the set. Here, θ^*_i for $i \in \mathcal{I}$ denotes another parameter. In short,

our data release mechanism has the form

$$\mathcal{M}_g(\theta, z) = \theta^*_{I(\theta)} \quad .$$

We call this class of data release mechanisms *many-to-one mechanisms*. The policy is fully determined by $\mathcal{S}_i$ and θ^*_i. Throughout the remainder of this chapter, we will demonstrate different ways of instantiating many-to-one mechanism to achieve (exactly or approximately) the lower bound in § 4.3.2.

Intuitively, many-to-one mechanisms will have a bounded distortion as long as $d\left(p_{X_\theta} \| p_{X_{\theta^*_{I(\theta)}}}\right)$ is bounded for all $\theta \in \text{Supp}(\Theta)$. At the same time, they obfuscate the secret as different data distributions within the same set are mapped to the same released parameter. It turns out this simple *deterministic* mechanism is sufficient to achieve optimal privacy-distortion tradeoffs in many cases, as opposed to the requirement for randomness in DP [3]. We will see examples in the case studies § 4.3.4.

4.3.4 Case Studies

We have results for a set of secrets (e.g., mean, quantile, fraction, standard deviation) over a set of distributions (e.g., Gaussian, uniform, exponential, geometric, binomial, Poisson, categorical). In this dissertation, we show a few typical examples, including mean of continuous distributions, quantile of exponential distributions, and fractions of categorical distributions.

Mean of Continuous Distributions

As discussed in § 4.1, the mean of the distributions can reveal sensitive information. In this section, we discuss how to protect the mean of continuous distributions. We first analyze the lower bound.

Corollary 4.3.4.1 (Privacy lower bound, secret = mean of a continuous distribution)**.** *Consider the secret function* $g(\theta) = \int_x x f_{X_\theta}(x)\, dx$*. For any* $T \in (0, 1)$*, when* $\Pi_\epsilon \leq T$*, we have* $\Delta > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot \epsilon$*.*

Indeed, we can construct a data release mechanism that achieves a tradeoff close to this bound.

Data release mechanism. We consider continuous distribution that can be parameterized with a location parameter, and the prior distribution of the location parameter is uniform and independent of other factors. More specifically, we make the following assumption.

***Assumption* 4.3.4.1.** *The distribution parameter vector θ can be written as (u, v), where $u \in \mathbb{R}$, $v \in \mathbb{R}^{q-1}$, and for any $u \neq u'$, $f_{X_{u,v}}(x) = f_{X_{u',v}}(x - u' + u)$. The prior over distribution parameters is $f_{U,V}(a, b) = f_U(a) \cdot f_V(b)$, where $f_U(a) = \frac{1}{\overline{u}-\underline{u}} \mathbb{I}(a \in [\underline{u}, \overline{u}))$.*

Examples of distributions that satisfy this assumption include the Gaussian, Laplace, and uniform distributions, as well as shifted distributions (e.g., shifted exponential, shifted log-logistic). The mechanism for this case is as follows.

***Mechanism* 4.3.4.1** (For secret = mean of a continuous distribution)**.** *The parameters of the data release mechanism are*

$$\mathcal{S}_{i,v} = \{(t, v) \,|t \in [\underline{u} + i \cdot s,\ \underline{u} + (i+1) \cdot s)\}, \tag{4.11}$$

$$\theta^*_{i,v} = (\underline{u} + (i + 0.5) \cdot s, v), \tag{4.12}$$

$$\mathcal{I} = \{(i, v) : i \in \{0, 1, \ldots, N-1\}, v \in Supp(p_V)\}, \tag{4.13}$$

where s is a hyper-parameter of the mechanism that divides $(\overline{u} - \underline{u})$ and $N = \frac{\overline{u}-\underline{u}}{s} \in \mathbb{N}$.

Fig. 4.4 shows an example when the original data distribution is Gaussian, i.e., $X_\theta \sim \mathcal{N}(u, v)$, and $u \in [\underline{\mu}, \overline{\mu})$. Intuitively, our data release mechanism "quantizes" the range of possible mean values into segments of length s. It then shifts the mean of private distribution $f_{X_{u,v}}$ to the midpoint of its corresponding segment, and releases the resulting distribution. This simple deterministic mechanism is able to achieve order-optimal privacy-distortion tradeoff in some cases, as shown below.

Proposition 4.3.4.1. *Under Assumption 4.3.4.1, Mechanism 4.3.4.1 has $\Delta = \frac{s}{2}$ and $\Pi_\epsilon \leq \frac{2\epsilon}{s}$.*

The two takeaways from this proposition are that: (1) data holder can use s to control the tradeoff between distortion and privacy, and (2) the mechanism is order-optimal. To see that, we can observe that this mechanism can achieve $\Delta = \frac{\epsilon}{T}$ for a privacy budget T. This quantity is on the same order as the lower bound in Proposition 4.3.4.1 w.r.t. ϵ and T.

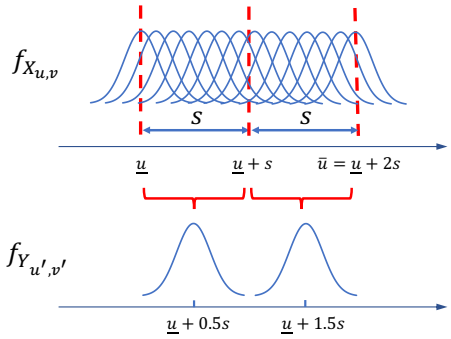


Figure 4.4: Illustration of the data release mechanism for continuous distributions when secret=mean.

Quantile of Exponential Distributions

As indicated in § 4.1, the quantiles of the distributions can reveal sensitive information. In this section, we discuss how to protect the quantiles of exponential distributions. More specifically, the distribution parameter is $\theta = \lambda$, where λ is the scale parameter. In other words, $f_{X_\lambda}(x) = \frac{1}{\lambda} e^{-x/\lambda}$.

We first analyze the lower bound.

Corollary 4.3.4.2 (Privacy lower bound, secret = α-quantile of a continuous distribution)**.** *Consider the secret function $g(\theta) = \alpha$-quantile of f_{X_θ}. For any $T \in (0,1)$, when $\Pi_\epsilon \leq T$, we have $\Delta > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot \frac{\epsilon}{-\ln(1-\alpha)}$ (i.e., $\gamma = -\frac{1}{2\ln(1-\alpha)}$).*

We can see that the γ parameter in this bound only depends on the value of α, and as γ increases, the lower bound becomes smaller.

Next, we provide data release mechanisms. Here, we assume that the parameters of the original data are drawn from a uniform distribution with lower and upper bounds. In more details, we make the following assumption.

***Assumption* 4.3.4.2.** *The prior over distribution parameter is p_λ is a uniform distribution over $\left[\underline{\lambda}, \overline{\lambda}\right)$.*

The parameters of the data release mechanism are specified below.

***Mechanism* 4.3.4.2** (For secret = quantile of a continuous distribution)**.**

$$\begin{aligned}\mathcal{S}_i &= [\underline{\lambda} + i \cdot s, \underline{\lambda} + (i+1) \cdot s) \quad , \\ \theta_i^* &= \underline{\lambda} + (i + 0.5) \cdot s \quad , \\ \mathcal{I} &= \mathbb{N},\end{aligned}$$

where $s > 0$ *is a hyper-parameter of the mechanism that divides* $\left(\overline{\lambda} - \underline{\lambda}\right)$*.*

This data release mechanism achieve the following Δ and Π_ϵ.

Proposition 4.3.4.2. *Under Assumption 4.3.4.2, Mechanism 4.3.4.2 achieves* $\Delta = \frac{1}{2}s$ *and* $\Pi_\epsilon = \frac{2\epsilon}{-\ln(1-\alpha)s}$*.*

The two takeaways are that: (1) data holder can use s to control the tradeoff between distortion and privacy, and (2) the mechanism is order-optimal. To see that, we observe this mechanism achieves $\Pi_\epsilon \cdot \Delta \leq 2\gamma\epsilon$. This is in the same order as the lower bound $\Delta > \left(\lceil \frac{1}{\Pi_\epsilon} \rceil - 1\right) \cdot 2\gamma\epsilon$.

Fraction of Categorical Distributions

As discussed in § 4.1, the fractions of the distributions can reveal sensitive information. In this section, we discuss how to protect the fractions of categorical distributions. More specifically, we assume that $\theta = (p_1, p_2, \ldots, p_C)$ s.t. $p_i \in [0, 1] \ \forall i \in [C]$ and $\sum_i p_i = 1$.

We first analyze the lower bound. Without loss of generality, we assume that we want to protect the fraction of the j-th bin, i.e. p_j.

Corollary 4.3.4.3 (Privacy lower bound, secret = fraction of a general discrete distribution)**.** *Consider the secret function* $g(\theta) = p_1$*. For any* $T \in (0, 1)$*, when* $\Pi_\epsilon \leq T$*, we have* $\Delta > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot \epsilon$*.*

In other words, the γ parameter in this bound is a constant. The lower bound purely depends on the privacy budget T and the threshold for attackers' guess ϵ.

Next, we present the data release mechanism under the following assumption.

***Assumption* 4.3.4.3.** *The prior distribution of* $(p_1, \ldots, p_C)$ *is a uniform distribution over all the probability simplex* $\{(p_1, \ldots, p_C) | p_i \in [0, 1) \ \forall i \in [C]$ *and* $\sum_i p_i = 1\}$*.*

***Mechanism* 4.3.4.3** (For secret = fraction of a general discrete distribution)**.** *The parameters of the mechanism are as follows.*

$$\begin{aligned}
\mathcal{S}_{p_1,\ldots,p_C} &= \left\{ \left(p_1 - \frac{t}{C-1}, \ldots, p_{j-1} - \frac{t}{C-1}, p_j + t, \right.\right. \\
&\qquad \left.\left. p_{j+1} - \frac{t}{C-1}, \ldots, p_C - \frac{t}{C-1} \right) \middle| t \in \left[-\frac{s}{2}, \frac{s}{2} \right) \right\} \quad , \\
\theta^*_{p_1,\ldots,p_C} &= \Big(p_1 - T, \ldots, p_{j-1} - T, p_j + (C-1)\,T, \\
&\qquad p_{j+1} - T, \ldots, p_{C+1} - T \Big) \quad ,
\end{aligned}$$

where $T = \min\{p_1, \ldots, p_{j-1}, p_{j+1}, \ldots, p_C, 0\}$*, and*

$$\begin{aligned}
\mathcal{I} = \Bigg\{ (p_1, \ldots, p_C) \Bigg| & \forall i \; p_i \in \left(-\frac{s}{2\,(C-1)}, 1 \right], \sum_i p_i = 1, \\
& p_j = (k + 0.5)\,s, \text{where } k \in \{0, 1, \ldots, C-1\} \Bigg\}.
\end{aligned}$$

Here $s > 0$ *is a hyper-parameter of the mechanism that divides 1.*

This data release mechanism achieve the following Δ and Π_ϵ.

Proposition 4.3.4.3. *Under Assumption 4.3.4.3, Mechanism 4.3.4.3 has the following* Δ *and* Π_ϵ *value/bound.*

$$\begin{aligned}
\Delta &= \frac{s}{2}, \\
\Pi_\epsilon &< \frac{2\epsilon}{s} + 1 - \left(1 - \frac{s}{C-1} \right)^{C-1}.
\end{aligned}$$

To understand this bound, we can see that when C is large, the bound for $\Pi_\epsilon \approx \frac{2\epsilon}{s} + 1 - e^{-s}$. When s is small, $\frac{2\epsilon}{s}$ dominates the above term. In those cases, s can be used to control the tradeoff between distortion and privacy. In addition, $\Pi_\epsilon \cdot \Delta \approx \epsilon$. This is in the same order as the lower bound $\Delta > \left(\lceil \frac{1}{\Pi_\epsilon} \rceil - 1 \right) \cdot \epsilon$.

Extending Data Release Mechanisms for Dataset Input/Output

The data release mechanisms discussed in previous sections assume that data holders know the *distribution parameter* of the original data. In practice, data holders often only have a dataset of samples from the data distribution and do not know the parameters of the underlying distributions. As mentioned in § 4.3.1, our data release mechanisms can be easily adapted to handle dataset input/output. In this section, we discuss how to achieve that.

The high-level idea is that the data holders can estimate the distribution parameters from the data samples, and find the corresponding $\mathcal{S}_i$ according to the estimated parameters. Then the original samples are modified as if they are sampled according to the parameter θ_i^*. As an example, we present the concrete procedure for the case where secrete is the mean of continuous distributions. For a dataset of $\mathcal{X} = \{x_1, \ldots, x_n\}$, the procedure is as follows:

1. Estimate the mean from the data samples: $\hat{\mu} = \frac{1}{n}\sum_{i\in[n]} x_i$.
2. According to Eq. (4.11), compute the index of the corresponding set $i = \lfloor \frac{\hat{\mu}-\underline{\mu}}{s} \rfloor$.
3. According to Eq. (4.12), change the mean of the data samples to $\mu_{target} = \underline{\mu}+(i+0.5)\cdot s$.
4. This can be done by sample-wise operation $x_i' = x_i - \hat{\mu} + \mu_{target}$.
5. The released dataset is $\mathcal{M}_g(\mathcal{X}, z) = \{x_1', \ldots, x_n'\}$.

Note that this mechanism applies to samples. Therefore, it can be applied either to the original data, or as an add-on to our GAN-based data sharing framework [8, 12, 6, 167, 168].

4.3.5 Experiments

In the previous sections, we demonstrated the optimality of our data release mechanisms through theoretical analysis. In this section, we focus on other *orthogonal* questions through real-world experiments: (1) how well our data release mechanisms perform when their assumptions do not hold in practice, and (2) why existing privacy frameworks are not suitable for distributional privacy.

Dataset. We use three real-world datasets to simulate each of the motivating scenarios in § 4.1. *(1) Wikipedia Web Traffic Dataset (WWT)* [160], which contains the daily page views of 145,063 Wikipedia web pages in 2015-2016. To prepare the data for our experiments, we remove web pages with empty page view records on any day (117,277 left), and compute

the mean page views for each web page across all dates. Our goal is to release the page views (a 117,277-dimensional vector) while protecting the mean of the distribution (which reveals the business scales of the company as described in § 4.1). *(2) Google Cluster Trace Dataset (GCT)* [169], which contains usage logs (e.g., CPU/memory) from an internal Google cluster with 12.5k machines in 2011. We use "platform ID" field of the dataset, which represents "microarchitecture and chipset version of the machine" [169]. Our goal is to release a distribution of platform ID while protecting the fraction of a specific platform ID (which reveals business strategy as described in § 4.1). *(3) Measuring Broadband America Dataset (MBA)* [170], which contains network statistics (including network traffic counters) collected by United States Federal Communications Commission from homes across the United States. We select the average network traffic (in GB per measurement) from AT&T clients as our data. Our goal is to release a copy of this data while hiding the 0.95-quantile (which reveals the network capability as described in § 4.1).

Baselines. We compare our mechanisms discussed in § 4.3.4 with two popular mechanisms proposed in prior work: differentially-private density estimation [149] (shortened to DP) and attribute-private Gaussian mechanism [171] (shortened to AP). For a dataset of samples $\mathcal{X} = \{x_1, ..., x_n\}$, DP works by: (1) Dividing the space into m bins: $B_1, ..., B_m$.[1] (2) Computing the histogram $C_i = \sum_{j=1}^{n} \mathbb{I}(x_j \in B_i)$. (3) Adding noise to the histograms $D_i = \max\{0, C_i + \text{Laplace}(0, \epsilon)\}$, where $\text{Laplace}\left(0, \epsilon^2\right)$ means a random noise from Laplace distribution with mean 0 and variance ϵ^2. (4) Normalizing the histogram $p_i = \frac{D_i}{\sum_{j=1}^{m} D_j}$. We can then draw y_i according to the histogram and release $\mathcal{Y} = \{y_1, ..., y_n\}$ with differential privacy guarantees. AP works by releasing $\mathcal{Y} = \left\{x_i + \mathcal{N}\left(0, \epsilon^2\right)\right\}_{i=1}^{n}$.[2]

Metrics. Our privacy and distortion metrics rely on the prior distribution of the original data $\theta \sim p_\Theta$ (though the mechanism does not). In practice (and also in these experiments), the data holder typically only has access to one dataset, which means we cannot evaluate these metrics directly. Instead, we use surrogate metrics as a way to bound our true privacy and distortion.

Surrogate privacy metric. For an original dataset $\mathcal{X} = \{x_1, ..., x_n\}$ and the released dataset $\mathcal{Y} = \{y_1, ..., y_n\}$, we define the surrogate privacy metric $\tilde{\Pi}_\epsilon$ as the error of an attacker who guesses the secret of the released dataset as the true secret: $\tilde{\Pi}_\epsilon \triangleq -|g(\mathcal{X}) - g(\mathcal{Y})|$,

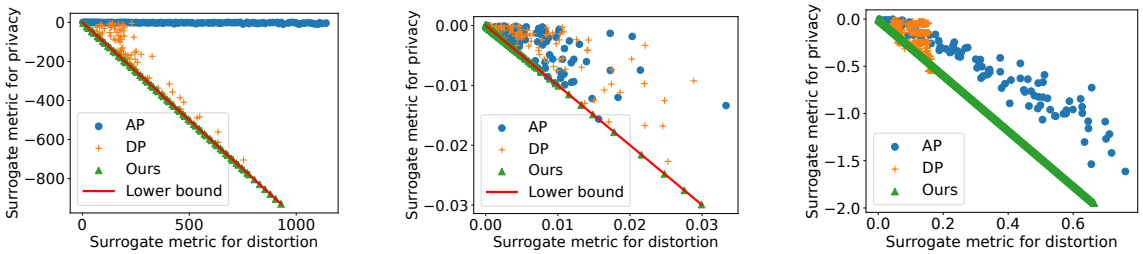
[1] In Google Cluster Trace Dataset, the bin is already pre-specified (i.e., the platform IDs), so this step is skipped.

[2] In Google Cluster Trace Dataset, the Gaussian noise $\mathcal{N}\left(0, \epsilon^2\right)$ are added to the counts of different platform IDs. We then normalize the counts and sample released platform IDs from this categorical distribution.

where $g(\mathcal{D})$ = mean of $\mathcal{D}$, fraction of a specific platform ID in $\mathcal{D}$, and 0.95-quantile of $\mathcal{D}$ in WWT, GCT, and MBA datasets respectively. Note that a minus sign is added to the definition of $\tilde{\Pi}_\epsilon$ so that a smaller value indicates stronger privacy, as in privacy metric Eq. (4.8). This simple attacker strategy is in fact a good proxy for evaluating the privacy Π_ϵ for the following reasons. (1) For our data release mechanisms for these secrets (Mechanisms 4.3.4.1 to 4.3.4.3), when the prior distribution is uniform, this strategy is actually optimal, so there is a direct mapping between $\tilde{\Pi}_\epsilon$ and Π_ϵ. (2) For AP applied on protecting mean of the data (i.e., in Wikipedia Web Traffic Dataset experiments), this strategy gives an unbiased estimator of the secret. (3) For DP and AP in other cases, this mechanism may not be an unbiased estimator of the secret, but it gives an *upper bound* on the attacker's error.

Surrogate distortion metric. We define our surrogate distortion metric as the distance between the two datasets: $\tilde{\Delta} \triangleq d(p_{\mathcal{X}} \| p_{\mathcal{Y}})$ where p_D denotes the empirical distribution of a dataset D, and d is defined as in our formulation § 4.3.1 (i.e., Wassersstein-1 distance for continuous distributions in WWT and MBA, and the TV distance for discrete distributions in GCT). This metric evaluates how much the mechanism distorts the dataset.

In fact, we can deduce a theoretical lower bound for the surrogate privacy and distortion metrics for secret = mean/fractions (shown later in Fig. 4.5) using similar techniques as the proofs of the main results.



(a) Wikipedia Web Traffic Dataset.

(b) Google Cluster Trace Dataset.

(c) Measuring Broadband America Dataset.

Figure 4.5: Privacy (lower is better) and distortion (lower is better) of AP, DP and our data release mechanisms. Each point represents one instance of the data release mechanism with one hyper-parameter. "Lower bound" is the theoretical lower bound of the achievable region. Our data release mechanisms achieve a better privacy-distortion tradeoff than AP and DP.

Results. We enumerate the hyper-parameters of each method (bin size and ϵ for DP, ϵ

for AP, and s for ours). For each method and each hyper-parameter, we compute their surrogate privacy and distortion metrics. The results are shown in Fig. 4.5 (bottom left is best); each data point represents one realization of mechanism $\mathcal{M}_g$ under a distinct hyper-parameter setting. Two takeaways are as follows.

(1) Our data release mechanisms has good privacy-distortion tradeoffs even when the assumptions do not hold. We envision that data holders can choose the data release mechanisms in the toolbox (Fig. 4.3) that matches their needs. However, in practical scenarios, the data distributions supported in the toolbox may not always exactly match real data. Our data release mechanisms for mean (i.e., Mechanism 4.3.4.1 used in WWT) and fractions (i.e., Mechanism 4.3.4.3 used in GCT) support general continuous distributions and categorical distributions, so there is no such distribution gap. Indeed, even for these surrogate metrics, our Mechanism 4.3.4.1 and Mechanism 4.3.4.3 are also optimal. This is visualized in Figs. 4.5a and 4.5b where we can see that our data release mechanisms match the theoretical lower bound of the tradeoff. However, our data release mechanisms for quantiles (i.e., Mechanism 4.3.4.2 used in Fig. 4.5c) are order-optimal only when the distributions follow the pattern of exponential distributions (§ 4.3.4). Despite the distribution mismatch, our data release mechanism still achieves a good privacy-distortion compared to DP and AP (Fig. 4.5c). More discussions are below.

(2) Our data release mechanisms achieve better privacy-distortion tradeoff than DP and AP. AP directly adds Gaussian noise to each sample, which does not change the mean of the distribution on expectation. As a result, Figure 4.5 shows that AP has a poor privacy-distortion tradeoff. DP quantizes (bins) the samples before adding noise. Quantization has a better property in terms of protecting the mean of the distribution, and therefore we see that DP has a better privacy-distortion tradeoff than AP, but still worse than ours. Note that in Fig. 4.5c, a few of the DP instances have better privacy-distortion tradeoffs than ours. This is *not* an indication that DP is fundamentally better. Instead, it is due to the randomness in DP (from the added Laplace noise), and some realizations of the specific noise in this experiment happened to lead to a better tradeoff. Another instance of the DP algorithm could lead to a poor tradeoff, so DP's achievable tradeoff points are widespread. In contrast, our data release mechanism has a better worst-case guarantee as indicated by the concentration of the tradeoff curve. In summary, these results confirm that DP and AP are not suitable for distributional privacy, and our distributional privacy framework provides better practical protections of distributional privacy.

4.3.6 Discussions

In this section, we show the initial promise of *distributional privacy* framework for defining, analyzing, and protecting distributional privacy concerns in data sharing applications. Our theoretical framework can be used to analyze the leakage of distributional information and the privacy-distortion tradeoffs of data release mechanisms (§ 4.3.1). Our data release mechanisms can be used to protect distributional information and are compatible with existing privacy-preserving data release frameworks (e.g., GANs) (§ 4.3.4). However, these results open up more research questions than they answered.

The distance metric d (Eq. (4.9)) and the secret function g (§ 4.3.1). These are the (only) two parameters that users need to provide to our distributional privacy framework. The distance metric d describes the properties of data that need to be *preserved*, while the secret function g captures what needs to be *obfuscated*. While we only discuss a limited set of secrets (§ 4.3.4) and distance metrics (§ 4.3.1), we believe that this framework is general for much broader use cases. Open questions include how to analyze the privacy-distortion tradeoff (and design mechanisms) for more secret types, multiple secrets, and other distance metrics. Our framework could also provide new insights into commonly used data sharing practices. For example, one widely used mechanism is to release normalized data [57, 15]. Although this mechanism was purely based on heuristic, it could actually be optimal (in terms of privacy-distortion tradeoff) under our framework for some specific d and g (e.g., when d is the distribution distance in the normalized space and g is the mean of the distribution). By analyzing the pairs of (d, g) for which a mechanism is optimal, we can have a better understanding of what the mechanism is good for.

The dimension and the type of the distributions. In this section, we consider a limited set of one-dimensional distributions. One important future direction is to expand this set of distributions so as to enrich the distributional privacy toolbox (Fig. 4.3) for users.

Extensions. One limitation of the current privacy metric Π_ϵ is that it depends on the prior distribution of the parameters p_Θ, which is often unknown in many applications. Motivated by maximal leakage [172], one possibility is to consider a *normalized* privacy metric:

$$\Pi'_\epsilon \triangleq \sup_{p_\Theta} \log \frac{\Pi_\epsilon}{\sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(p_\Theta\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right)},$$

where $\hat{g}\left(p_\Theta\right)$ is an attacker that knows the prior distribution but does not see the released data, and the denominator is the probability that the strongest attacker guesses the secret within tolerance ϵ. Similar to maximal leakage, we then take the log to transform the unit to nats, and then consider the worst-case leakage among all possible priors. This *normalized* Π'_ϵ has a completely different meaning to Π_ϵ: it considers how much additional "information" that the released data provides to the attacker in the worst-case scenario. It would be interesting to study the feasibility of this formulation and the associated privacy-distortion tradeoffs and mechanisms.

4.4 Chapter Summary

In this chapter, we study the two important privacy concerns: *sample-level privacy* and *distributional privacy*. Regarding *sample-level privacy*, we show that GANs without any special training mechanism have DP guarantees and are robust to membership inference attacks. However, the DP guarantee is too weak to be provide protection. To enhance GANs' sample-level privacy guarantees, we explore the use of DP-SGD, but we find that it significantly reduces the fidelity of GANs. To address this challenge, we propose to pretrain GANs on public data before training on the private data, which shows promising results. Regarding *distributional privacy*, we propose a new theoretical framework for defining, analyzing, and protecting these concerns. We analyze the fundamental achievable privacy-distortion tradeoffs and propose practical data release mechanisms that are close to the optimal bounds. We demonstrate on real-world datasets that our mechanisms achieve better privacy-distortion tradeoffs than existing privacy frameworks such as differential privacy and attribute privacy.

Broader impacts. The privacy concerns addressed in this chapter are not limited to data sharing applications but also extend to other GAN applications, such as content creation [55], as long as the training data is sensitive. Our theoretical results and proposed techniques can be applied to these applications as well.

More broadly, our *distributional privacy* framework is a general framework for analyzing distributional concerns. The proposed data release mechanisms can be used not only with GANs, but also with other data sharing frameworks. For example, they can be used to modify synthetically generated samples after they are generated (by any framework), to modify the training dataset for a generative model, or to directly modify the original data for release.

Chapter 5

Applications

In this chapter, we explore how we can use the insights from previous chapters to build data sharing tools for concrete applications. We observe that time series is common in many applications, and we use it as a case study. Applying GANs, which were originally successful in image generation, to time series data presents additional challenges in terms of fidelity. We design new network architectures and loss functions to tackle these challenges (§ 5.1). We package our time series generator, along with all the algorithmic advances discussed in this dissertation, into a modular package to facilitate the development of data sharing tools for future applications (§ 5.2). To demonstrate the utility of our tool, we present the performance of our system in time series data from the systems and networking domains as case studies (§ 5.3). Finally, we conclude this chapter in § 5.4.

5.1 Meta Architecture for Time Series Data

5.1.1 Motivation

Time series data is widely used in many domains. In medical domains, we have health record measurements, such as oxygen saturation, heart rate, and respiratory rate [173], which can be used to train diagnosis systems. In financial domains, we have bank transaction data [174], which can be used for fraud detection [175] and user behavior analysis. In web domains, we have webpage view data with metadata of URLs, which can be used to predict future views, analyze page correlations [176], or generate recommendations [177, 178]. In networking domains, we have network measurements of packet loss rate, bandwidth, and delay, with metadata such as location or device type that are useful for network management [56]. In systems domains, we have cluster usage measurements of metrics such as CPU/memory usage associated with metadata (e.g., server and job type) that can inform resource provisioning [179] and job scheduling [180]. In this chapter, we take time series data as an example and aim to design a GAN-based data sharing tools for these datasets.

5.1.2 Problem Formulation

At a high level, the time series data in many of the examples above consists of time series samples (e.g., bandwidth measurements) with high-dimensional data points and associated metadata that can be either numeric or categorical (e.g., IP address, location). We abstract them as follows: A *dataset* $\mathcal{D} = \{O^1, O^2, ..., O^n\}$ is defined as a set of *samples* O^i (e.g., the clients). Each sample $O^i = (A^i, R^i)$ contains m *metadata* $A^i = [A^i_1, A^i_2, ..., A^i_m]$. For example, metadata A^i_1 could represent client i's physical location, and A^i_2 the client's ISP. Note that we can support datasets in which multiple samples have the same set of metadata. The second component of each sample is a time series of *records* $R^i = [R^i_1, R^i_2, ..., R^i_{T^i}]$, where R^i_j means j-th measurement of i-th client. Different samples may contain a different number of measurements. The number of records for sample O^i is given by T^i. Each record $R^i_j = (t^i_j, f^i_j)$ contains a *timestamp* t^i_j, and K *measurements* $f^i_j = [f^i_{j,1}, f^i_{j,2}, ..., f^i_{j,K}]$. For example, t^i_j represents the time when the measurement f^i_j is taken, and $f^i_{j,1}$, $f^i_{j,2}$ represent the ping loss rate and traffic byte counter at this timestamp respectively. Note that the timestamps are sorted, i.e. $t^i_j < t^i_{j+1}$ $\forall 1 \leq j < T^i$. This abstraction fits many classes of data that appear in the wild. Our problem is to take any such dataset as input and learn a model that can generate a new dataset $\mathcal{D}'$ as output.

5.1.3 Challenges

The first challenge relates to *long-term temporal correlations*. As we see in Fig. 5.1, the canonical GAN does poorly in capturing temporal correlations trained on the Wikipedia Web Traffic Dataset (WWT) dataset.[1] Concurrent and prior work on using GANs for other time series data has also observed this [185, 77, 8, 186]. One approach to address this is segmenting long datasets into chunks; e.g., TimeGAN [186] chunks datasets into smaller time series each of 24 epochs, and only evaluates the model on producing new time series of this length [187]. This is not viable in our domain, as relevant properties of networking/systems data often occur over longer time scales (e.g., network measurements) (see Fig. 5.1). Second, *mode collapse* is a well-known problem in GANs where they generate only a few modes of the underlying distribution [46, 166, 103, 104]. We have discussed how to alleviate mode collapse in § 3.2. But as we will see that, the mode collapse problem here

[1] This uses: (1) a dense multilayer perceptron (MLP) which generates measurements and metadata jointly, (2) an MLP discriminator, and (3) Wasserstein loss [103, 104], consistent with prior work [181, 182, 183, 184].

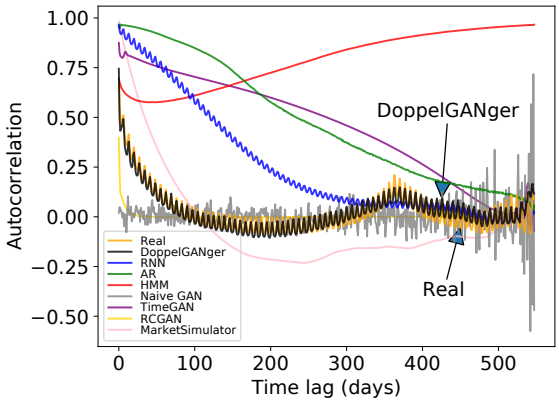


Figure 5.1: Autocorrelation of daily page views for Wikipedia Web Traffic Dataset.

is unique in time series data when there is the high variability in the range of measurement values. Finally, we need to capture *complex relations between measurements and metadata* (e.g., packet loss rate and ISP), and across different measurements (e.g., packet loss rate and byte counts). As such, state-of-the-art approaches either do not co-generate attributes with time series data or their accuracy for such tasks is unknown [8, 188, 186], and directly generating joint metadata with measurements samples does not converge (§ 5.3). Further, generating independent time series for each measurementdimension will break their correlations.

5.1.4 DoppelGANger (DG) Design

In this section, we describe how we tackle fidelity shortcomings of time series GANs. Recall that existing approaches have issues in capturing temporal effects and relations between metadata and measurements. In what follows, we present our solution starting from the canonical GAN strawman and extend it to address these challenges. Finally, we summarize the design and guidelines for users to use our workflow.

Capturing long-term effects

Recall that the canonical GAN generator architecture is a fully-connected multi-layer perceptron (MLP), which we use in our strawman solution (§ 5.1.3). As we saw, this architecture fails to capture long-term correlations (e.g., annual correlations in page views).

RNN primer and limitations. Similar to prior efforts, we posit that the main reason

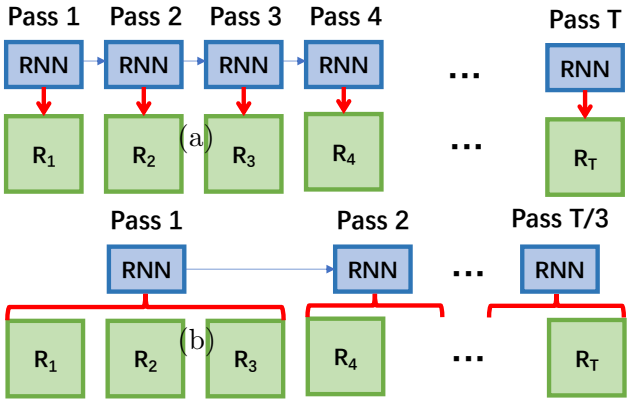


Figure 5.2: (a) The usual way of generating time series. (b) Batch generation with $S = 3$. The RNN is a single neural network, even though many units are illustrated. This *unrolled* representation conveys that the RNN is being used many times to generate samples.

is that MLPs are not well suited for time series. A better choice is to use recurrent neural networks (RNNs), which are designed to model time series and have been widely used in the GAN literature to generate time series [189, 8, 77, 188, 186]. Specifically, we use a variant of RNN called long short-term memory (LSTM) [190].

At a high level, RNNs work as follows (Fig. 5.2 (a)). Instead of generating the entire time series at once, RNNs generate one record R_j^i at a time (e.g., page views on the jth day), and then run T^i (e.g., the number of days) passes to generate the entire time series. The key difference in a RNN from traditional neural units is that RNNs have an internal state that implicitly encodes all past states of the signal. Thus, when generating R_j^i, the RNN unit can incorporate the patterns in $R_1^i, ..., R_{j-1}^i$ (e.g., all page views before the j-th day). Note that RNNs can learn correlations across the dimensions of a time series, and produce multi-dimensional outputs.

However, we empirically find that RNN generators still struggle to capture temporal correlations when the length exceeds a few hundred epochs. The reason is that for long time series, RNNs take too many passes to generate the entire sample; the more passes taken, the more temporal correlation RNNs tend to forget. Prior work copes with this problem in three ways. The first is to generate only short sequences [189, 77, 186]; long datasets are evaluated on chunks of tens of samples [186, 187]. The second approach is to train on small datasets, where rudimentary designs may be able to effectively memorize long term effects (e.g. unpublished work [191] generates time series of length 1,000, from

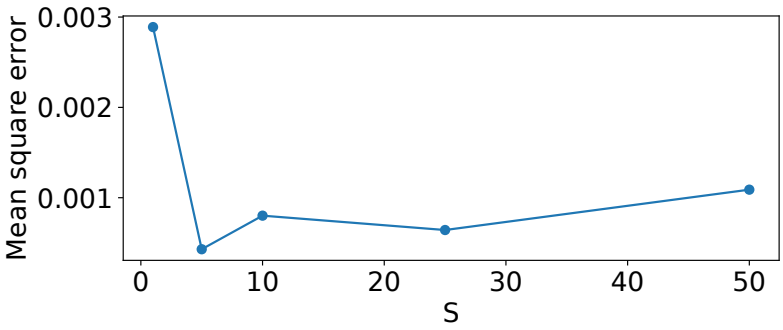


Figure 5.3: Error vs. batch parameter.

a dataset of about 100 time series). This approach leads to memorization [192], which defeats the purpose of training a model. A third approach assumes an auxiliary raw data time series as an additional input during the generation phase to help generate long time series [188]. This again defeats the purpose of synthetic data generation.

Our approach. To reduce the number of RNN passes, we propose to use a simple yet effective idea called *batch generation*. At each pass of the RNN, instead of generating one record (e.g., page views of one day), it generates S records (e.g., page views of S consecutive days), where S is a tunable parameter (Fig. 5.2 (b)).[2] This effectively reduces the total number of RNN passes by a factor of S. As S gets larger, the difficulty of synthesizing a batch of records at a single RNN pass also increases. This induces a natural tradeoff between the number of RNN passes and the single pass difficulty. For example, Fig. 5.3 shows the mean square error between the autocorrelation of our generated signals and real data on the WWT dataset. Even a small (but larger than 1) S gives substantial improvements in signal quality. In practice, we find that $S = 5$ works well for many datasets and a simple autotuning of this hyper-parameter similar to this experiment can be used in practice (§ 5.1.4).

The above workflow of using RNNs with batch generation ignores the timestamps from generation. In practice, for some datasets, the timestamps may be important in addition to timeseries samples; e.g., if derived features such as inter-arrival times of requests

[2]Our batch generation differs from two similarly-named concepts. *Minibatching* is a standard practice of computing gradients on small sets of samples rather than the full dataset for efficiency [193]. *Generating batches of sequences* in SeqGAN [77] involves generating multiple time series during GAN training to estimate the reward of a generator policy in their reinforcement learning framework. Both are orthogonal to our batch generation.

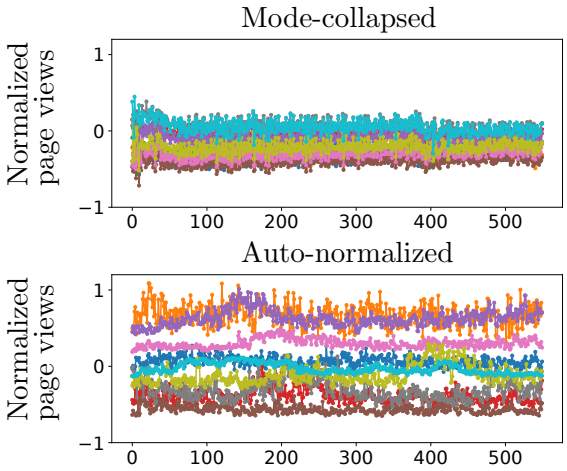


Figure 5.4: Without auto-normalization, generated samples show telltale signs of mode collapse as they have similar shapes and amplitudes.

may be important for downstream systems and networking tasks. To this end, we support two simple alternatives. First, if the raw timestamps are not important, we can assume that they are equally spaced and are the same for all samples (e.g., when the dataset is daily page views of websites). Second, if the derived temporal properties are critical, we can simply use the initial timestamp of each sample as an additional metadata (i.e., start time of the sample) and the inter-arrival times between consecutive records as an additional measurement.

Tackling Mode Collapse

Mode collapse is a well-known problem [44], where the GAN outputs homogeneous samples despite being trained on a diverse dataset. For example, suppose we train on web traffic data that includes three distinct kinds of signals, corresponding to different classes of users. A mode-collapsed GAN might learn to generate only one of those traffic types.

For instance, Fig. 5.4 (top) plots synthetic time series from a GAN trained on the WWT dataset, normalized and shifted to $[-1, 1]$. The generated samples are heavily mode-collapsed, exhibiting similar amplitudes, offsets, and shapes.[3]

Existing work and limitations. Alleviating mode collapse is an active research topic

[3]While mode collapse can happen both in measurements or in metadata, we observed substantially more mode collapse in the measurements.

in the GAN community (e.g., [46, 166, 103, 104]). We experimented with a number of state-of-the-art techniques for mitigating mode collapse [166, 104]. However, these did not resolve the problem on our datasets.

Our approach. Our intuition is that unlike images or medical data, where value ranges tend to be similar across samples, networking datasets exhibit much higher range variability. Datasets with a large range (across samples) appear to worsen mode collapse because they have a more diverse set of modes, making them harder to learn. For example, in the WWT dataset, some web pages consistently have >2000 page views per day, whereas others always have <10.

Rather than using a general solution for mode collapse, we build on this insight to develop a custom *auto-normalization* heuristic. Recall that each time series of measurements f^i (e.g., network traffic volume measurement of a client) is also assigned some metadata A^i (e.g., the connection technology of the client, cable/fiber). Suppose our dataset has two time series with different offsets: $f^1(t) = \sin(t)$ and $f^2(t) = \sin(t) + 100$ and no metadata, so $A^1 = A^2 = ()$. We have $\min(f^1) = -1$, $\max(f^1) = 2$, $\min(f^2) = 99$, $\max(f^2) = 101$. A standard normalization approach (e.g. as in [187]) would be to simply normalize this data by the *global* min and max, store them as global constants, and train on the normalized data. However, this is just scaling and shifting by a constant; from the GAN's perspective, the learning problem is the same, so mode collapse still occurs.

Instead, we normalize each time series signal *individually*, and store the min/max as "fake" metadata. Rather than training on the original (f^i, A^i) pairs, we train on $\tilde{f}^1(t) = \sin(t)$, $\tilde{A}^1 = (-1, 1)$, $\tilde{f}^2(t) = \sin(t)$, $\tilde{A}^2 = (99, 101)$.[4] Hence, the GAN learns to generate these two fake metadata defining the max/min for each time series individually, which are then used to rescale measurements to a realistic range.

Note that this approach differs from typical feature normalization in two ways: (1) it normalizes each sample individually, rather than normalizing over the entire dataset, and (2) it treats the maximum and minimum value of each time series as a random variable to be learned (and generated). In this way, all time series have the same range during generation, which alleviates the mode collapse problem. Fig. 5.4 (bottom) shows that by training distributional privacy with auto-normalization on the WWT data, we generate samples with a broad range of amplitudes, offsets, and shapes.

[4] In reality, we store $\tilde{A}^i = (\max\{f^i\} \pm \min\{f^i\})/2$ to ensure that our generated min is always less than our max.

Capturing attribute relationships

So far, we have only discussed how to generate time series. However, metadata can strongly influence the characteristics of measurements. For example, fiber users tend to use more traffic than cable users. Hence, we need a mechanism to model the *joint* distribution between measurements and metadata. As discussed in § 5.1.3, naively generating concatenated metadata A^i and measurements R^i does not learn the correlations between them well. We hypothesize that this is because jointly generating metadata and measurements using a single generator is too difficult.

Existing work and limitations. A few papers have tackled this problem, mostly in the context of generating multi-dimensional data. The dominant approach in the literature is to train a variant of GANs called conditional GANs (CGANs), which learn to produce data conditioned on a user-provided input label. For example, prior works [8, 194, 188] learn a conditional model in which the user inputs the desired metadata, and the architecture generates measurements conditioned on the attributes; generating the attributes as well is a simple extension [8]. TimeGAN claims to co-generate metadata and measurements, but it does not evaluate on any datasets that include metadata in the paper, nor does the released code handle metadata [186, 187].

Our Approach. We start by decoupling this problem into two sub-tasks: generating metadata and generating measurements *conditioned* on metadata: $P(A^i, R^i) = P(A^i) \cdot P(R^i|A^i)$, each using a dedicated generator; this is is conceptually similar to prior approaches [8, 188]. More specifically, we use a standard multi-layer perceptron (MLP) network for generating the metadata. This is a good choice, as MLPs are good at modeling (even high-dimensional) non-time-series data. For measurement generation, we use the RNN-based architecture from § 5.1.4. To preserve the hidden relationships between the metadata and the measurements, the generated metadata A^i is added as an input to the RNN at every step.

Recall from § 5.1.4 that we treat the max and min of each time series as metadata to alleviate mode collapse. Using this conditional framework, we divide the generation of max/min metadata into three steps: (1) generate "real" metadata using the MLP generator (§ 5.1.4); (2) with the generated metadata as inputs, generate the two "fake" (max/min) metadata using another MLP; (3) with the generated real and fake metadata as inputs, generate measurements using the architecture in § 5.1.4 (see Fig. 5.6).

Unfortunately, a decoupled architecture alone does not solve the problem. Empir-

ically, we find that when the average length of measurements is long (e.g., in the WWT dataset, each sample consists of 550 consecutive daily page views), the fidelity of generated data—especially the metadata—is poor. To understand why, recall that a GAN discriminator judges the fidelity of generated samples and provides feedback for the generator to improve. When the total dimension of samples (measurements + metadata) is large, judging sample fidelity is hard.

Motivated by this, we introduce an *auxiliary discriminator* which discriminates only on metadata. The losses from two discriminators are combined by a weighting parameter α: $\min_G \max_{D_1,D_2} \mathcal{L}_1(G, D_1) + \alpha\mathcal{L}_2(G, D_2)$ where $\mathcal{L}_i$, $i \in \{1, 2\}$ is the Wasserstein loss of the original and the auxiliary discriminator respectively. The generator effectively learns from this auxiliary discriminator to generate high-fidelity metadata. Further, with the help of the original discriminator, the generator can learn to generate measurements well.

Empirically, we find that this architecture improves the data fidelity significantly. Fig. 5.5 shows a histogram of the (max+min)/2 metadata distribution from distributional privacy on the WWT dataset. That is, for each time series, we extract the maximum and minimum value, and compute their average; then we compute a histogram of these averages over many time series. This distribution implicitly reflects how well distributional privacy reproduces the range of time series values in the dataset. We observe that adding the auxiliary discriminator significantly improves the fidelity of the generated distribution, particularly in the tails of the true distribution.

Putting it all together

The overall distributional privacy architecture is in Fig. 5.6, highlighting the key differences from canonical approaches. First, to capture the correlations between metadata and measurements, we use a decoupled generation of metadata and measurements using an auxiliary discriminator, and conditioning the measurements based on the metadata generated. Second, to address the mode collapse problem for the measurements, we add the fake metadata capturing the min/max values for each generated sample. Third, we use a batched RNN generator to capture the temporal correlations and synthesize long time series that are representative.

The training phase requires two primary inputs: the *data schema* (i.e., metadata and measurement dimensions, indicating whether they are categorical or numeric) and the *training data*. The only minimal tuning that data holders sharing a dataset using distri-

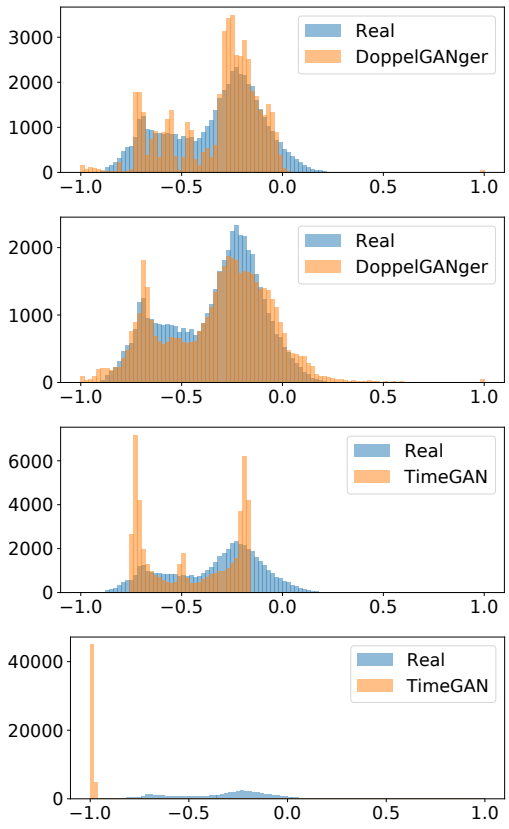


Figure 5.5: Distribution of (max+min)/2 of (a) DG without and (b) DG with the auxiliary discriminator, (c) TimeGAN, and (d) RCGAN (WWT data).

butional privacy need to be involved in is selecting the measurement batch size S (§ 5.1.4) controls the number of measurements generated at each RNN pass. Empirically, setting S so that T/S (the number of steps RNN needs to take) is around 50 gives good results, whereas prior time series GANs use $S = 1$ [8, 195, 186, 188]. Optionally, data holders can specify sensitive metadata, whose distribution can be masked or request additional privacy settings to be used. We envision data holders sharing the *generative model* with the data users. Users can then flexibly use this model and also optionally specify different metadata distribution (e.g., for amplifying rare events) if needed. That said, our workflow also accommodates a more restrictive mode of sharing, where the holder uses distributional privacy to generate synthetic data internally and then releases the generated data without

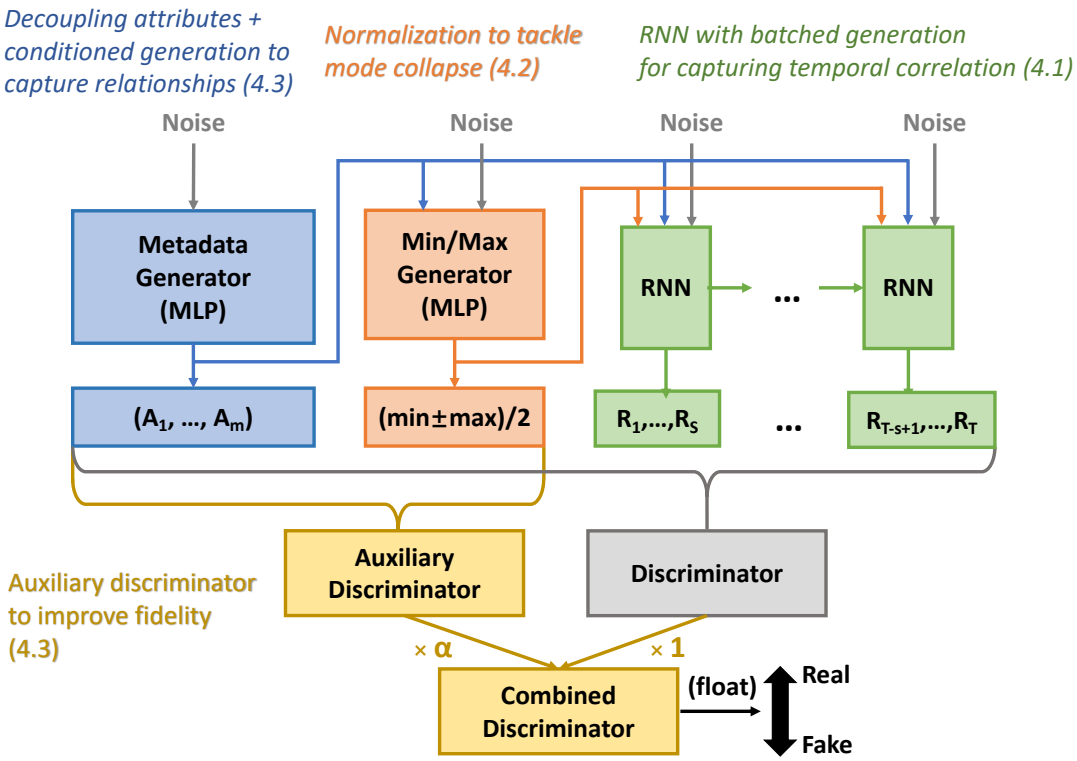


Figure 5.6: Architecture of distributional privacy highlighting key ideas and extensions to canonical GAN approaches.

sharing the model.[5]

The code and a detailed documentation (on data format, hyper-parameter setting, model training, data generation, etc.) are available at https://github.com/fjxmlzn/DoppelGANger.

5.2 Unified Library for Future Applications

To make it easier for anyone to use the proposed techniques in this dissertation in future applications, we package the algorithms (DoppelGANger for time series data in § 5.1, PacGAN for improving sample diversity in § 3.2, BSSN for stabilizing training in § 3.3) and DP-SGD training in § 4.2 into a single package. The package supports several data

[5]From a privacy perspective, model and data sharing may suffer similar information leakage risks [146], but this may be a pragmatic choice some providers can make nonetheless.

Dataset	Correlated in time & metadata	Multi-dimensional measurements	Variable-length signals
WWT [160]	x		
MBA [170]	x	x	
GCT [169]	x	x	x

Table 5.1: Challenging properties of studied datasets.

formats (e.g., CSV, XLS). Additionally, the library is designed to be modular, allowing for easy extension to new data types and the creation of new data sharing algorithms on top of it. Please refer to the documentation for more details: https://github.com/netsharecmu/NetShare.

5.3 Case Studies

In this section, we demonstrate the effectiveness of our system at preserving sample fidelity in concrete applications in systems and networking domains as case studies.

5.3.1 Setup

Datasets

These datasets are chosen to exhibit different combinations of challenges: (1) correlations within time series and metadata, (2) multi-dimensional measurements, and/or (3) variable measurement lengths.

Wikipedia Web Traffic Dataset (WWT). This dataset tracks the number of daily views of Wikipedia articles, starting from July 1st, 2015 to December 31st, 2016 [160]. In our language, each *sample* is a page view counter for one Wikipedia page, with three *metadata*: Wikipedia domain, type of access (e.g., mobile, desktop), and type of agent (e.g., spider). Each sample has one measurement: the number of daily page views.

Measuring Broadband America Dataset (MBA). This dataset was collected by United States Federal Communications Commission (FCC) [170] and consists of several measurements such as round-trip times and packet loss rates from several clients in geographically diverse homes to different servers using different protocols (e.g. DNS, HTTP, PING). Each *sample* consists of measurements from one device. Each sample has three *metadata*: Internet connection technology, ISP, and US state. A record contains UDP

ping loss rate (min. across measured servers) and total traffic (bytes sent and received), reflecting client's aggregate Internet usage.

Google Cluster Trace Dataset (GCT). This dataset [169] contains usage traces of a Google Cluster of 12.5k machines over 29 days in May 2011. We use the logs containing measurements of task resource usage, and the exit code of each task. Once the task starts, the system measures its resource usage (e.g. CPU usage, memory usage) per second, and logs aggregated statistics every 5 minutes (e.g., mean, maximum). Those resource usage values are the *measurements*. When the task ends, its end event type (e.g. FAIL, FINISH, KILL) is also logged. Each task has one end event type, which we treat as an *metadata*.

Baselines

We compare DoppelGANger with the following popular algorithms for time series data.

Hidden Markov models (HMM). While HMMs have been used for generating time series data, there is no natural way to jointly generate metadata and time series in HMMs. Hence, we infer a separate multinomial distribution for the metadata. During generation, metadata are randomly drawn from the multinomial distribution on training data, independently of the time series.

Nonlinear auto-regressive (AR). Traditional AR models can only learn to generate measurements. In order to jointly learn to generate metadata and measurements, we design the following more advanced version of AR: we learn a function f such that $R_t = f(A, R_{t-1}, R_{t-2}, ..., R_{t-p})$. To boost the accuracy of this baseline, we use a multi-layer perceptron version of f. During generation, A is randomly drawn from the multinomial distribution on training data, and the first record R_1 is drawn a Gaussian distribution learned from training data.

Recurrent neural networks (RNN). In this model, we train an RNN via teacher forcing [196] by feeding in the true time series at every time step and predicting the value of the time series at the next time step. Once trained, the RNN can be used to generate the time series by using its predicted output as the input for the next time step. A traditional RNN can only learn to generate measurements. We design an extended RNN takes metadata A as an additional input. During generation, A is randomly drawn from the multinomial distribution on training data, and the first record R_1 is drawn a Gaussian distribution learned from training data.

Naive GAN. We include the naive GAN architecture (MLP generator and discriminator)

in all our evaluations.

TimeGAN. Note that the state-of-the-art TimeGAN [187] does not jointly generate metadata and high-dimensional time series of different lengths, so several of our evaluations cannot be run on TimeGAN. However, we modified the TimeGAN implementation directly [187] to run on the WWT dataset (without metadata) and compared against it.

RCGAN [8]. RCGAN does not generate metadata, and only deals with time series of the same length, so again, several of our evaluations cannot be run on RCGAN. To make a comparison, we used the version without conditioning (called RGAN [8]) from the official implementation [197] and evaluate it on the WWT dataset (without metadata).

Market Simulator [198]. We also compare against a VAE-based approach [198] designed to generate synthetic financial market data, since its code is publicly available.

Metrics

Evaluating GAN fidelity is notoriously difficult [199, 200]; the most widely-accepted metrics are designed for labelled image data [115, 201] and cannot be applied to our datasets. Moreover, numeric metrics do not always capture the qualitative problems of generative models. We therefore evaluate DG with a combination of qualitative and quantitative microbenchmarks and downstream tasks that are tailored to each of our datasets. Our *microbenchmarks* evaluate how closely a statistic of the generated data matches the real data. E.g., the statistics could be attribute distributions or autocorrelations, and the similarity can be evaluated qualitatively or by computing an appropriate distance metric (e.g., mean square error, Jensen-Shannon divergence). Our *downstream tasks* use the synthetic data to reason about the real data, e.g., attribute prediction or algorithm comparison. In line with the recommendations of [199], these tasks can be evaluated with quantitative, task-specific metrics like prediction accuracy. Each metric is explained in more detail inline.

5.3.2 Results

Structural characterization. In line with prior recommendations [33], we explore how DG captures structural data properties like temporal correlations, metadata distributions, and metadata-measurement joint distributions.[6]

[6] Such properties are sometimes ignored in the ML literature in favor of downstream performance metrics; however, in systems and networking, we argue such microbenchmarks are important.

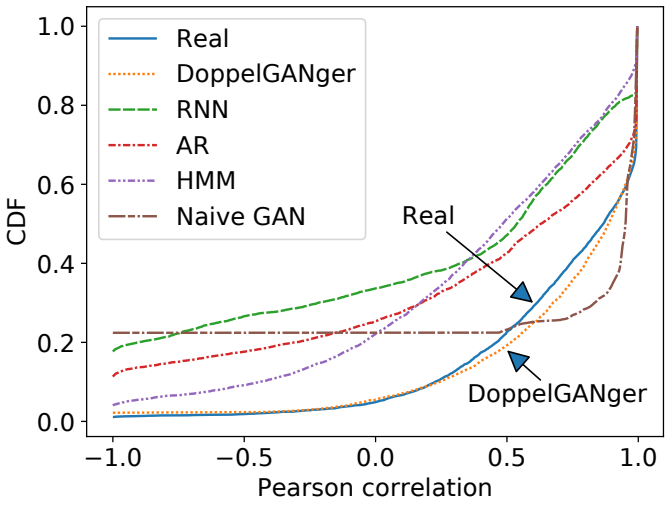


Figure 5.7: CDF of Pearson correlation between CPU rate and assigned memory usage from GCT.

Temporal correlations: To show how DG captures temporal correlations, Fig. 5.1 shows the average autocorrelation for the WWT dataset for real and synthetic datasets (discussed in §5.1.3). As mentioned before, the real data has a short-term weekly correlation and a long-term annual correlation. DG captures both, as evidenced by the periodic weekly spikes and the local peak at roughly the 1-year mark, unlike our baseline approaches. It also exhibits a 91.2% lower mean square error from the true data autocorrelation than the closest baseline (RCGAN).

The fact that DG captures these correlations is surprising, particularly since we are using an RNN generator. Typically, RNNs are able to reliably generate time series of length around 20, while the length of WWT measurements is 550. We believe this is due to a combination of adversarial training (not typically used for RNNs) and our batch generation. Empirically, eliminating either feature hurts the learned autocorrelation. TimeGAN and RCGAN, for instance, use RNNs and adversarial training but does not batch generation, though its performance may be due to other architectural differences [186, 8]. E.g., WWT is an order of magnitude longer than the time series it evaluates on [187, 197].

Another aspect of learning temporal correlations is generating time series of the right length. Fig. 5.8 shows the duration of tasks in the GCT dataset for real and synthetic datasets generated by DG and RNN. Note that TimeGAN generates time series of different

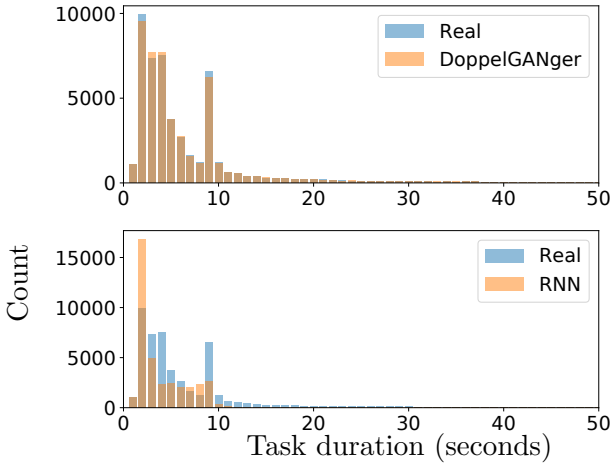


Figure 5.8: Histogram of task duration for the Google Cluster Trace Dataset. RNN-generated data misses the second mode, but DoppelGANger captures it.

lengths by first generating time series of a maximum length and then truncating according to the empirical length distribution from the training data [187]. Hence we do not compare against TimeGAN because the comparison is not meaningful; it perfectly reproduces the empirical length distribution, but not because the generator is learning to reproduce time series lengths.

DG's length distribution fits the real data well, capturing the bimodal pattern in real data, whereas RNN fails. Other baselines are even worse at capturing the length distribution. We observe this regularly; while DG captures multiple data modes, our baselines tend to capture one at best. This may be due to the naive randomness in the other baselines. RNNs and AR models incorporate too little randomness, causing them to learn simplified duration distributions; HMMs instead are too random: they maintain too little state to generate meaningful results.

Cross-measurement correlation: To evaluate correlations between the dimensions of our measurements, we computed the Pearson correlation between the CPU and memory measurements of generated samples from the GCT dataset. Figure 5.7 shows the CDF of these correlation coefficients for different time series. We observe that DG much more closely mirrors the true auto-correlation coefficient distribution than any of our baselines.

Measurement distribution: As discussed in § 5.1.4 and Fig. 5.6, DG captures the distribution of (max+min)/2 of page views in WWT dataset. As a comparison, TimeGAN and

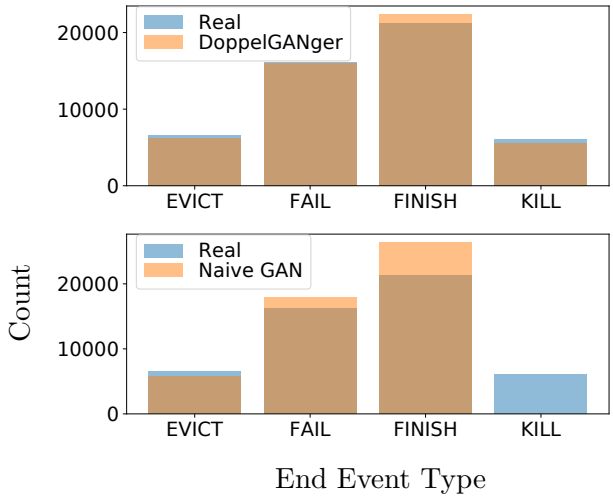


Figure 5.9: Histograms of end event types from GCT.

RCGAN have much worse fidelity. TimeGAN captures the two modes in the distribution, but fails to capture the tails. RCGAN does not learn the distribution at all. In fact, we find that RCGAN has severe mode collapse in this dataset: all the generated values are close to -1. Some possible reasons might be: (1) The maximum sequence length experimented in RCGAN is 30 [197], whereas the sequence length in WWT is 550, which is much more difficult; (2) RCGAN used different numbers of generator and discriminator updates per step in different datasets [197]. We directly take the hyper-parameters from the longest sequence length experiment in RCGAN's code [197], but other fine-tuned hyper-parameters might give better results. Note that unlike RCGAN, DG is able to achieve good results in our experiments without tuning the numbers of generator and discriminator updates.

Metadata distribution: Learning correct metadata distributions is necessary for learning measurement-metadata correlations. As mentioned in §5.3.1, for our HMM, AR, and RNN baselines, metadata are randomly drawn from the multinomial distribution on training data because there is no clear way to jointly generate metadata and measurements. Hence, they trivially learn a perfect metadata distribution. Fig. 5.9 shows that DG is also able to mimic the real distribution of end event type distribution in GCT dataset, while naive GANs miss a category entirely; this appears to be due to mode collapse, which we mitigate with our second discriminator.

Measurement-metadata correlations: Although our HMM, AR, and RNN baselines learn perfect metadata distributions, it is substantially more challenging (and important) to

	DoppelGANger	AR	RNN	HMM	Naive GAN
DSL	**0.68**	1.34	2.33	3.46	1.14
Cable	**0.74**	6.57	2.46	7.98	0.87

Table 5.2: Wasserstein-1 distance of total bandwidth distribution of DSL and cable users. Lower is better.

learn the joint metadata-measurement distribution. To illustrate this, we compute the CDF of total bandwidth for DSL and cable users in MBA dataset. Table 5.2 shows the Wasserstein-1 distance between the generated CDFs and the ground truth,[7] showing that DG is closest to the real distribution.

DG does not overfit: In the context of GANs, overfitting is equivalent to memorizing the training data, which is a common concern with GANs [192, 202]. To evaluate this, we ran an experiment inspired by the methodology of [192]: for a given generated DG sample, we find its nearest samples in the training data. We observe significant differences (both in square error and qualitatively) between the generated samples and the nearest neighbors on all datasets, suggesting that DG is not memorizing.

Resource costs: DG has two main costs: training data and training time/computation. In Fig. 5.10, we plot the mean square error (MSE) between the generated samples' autocorrelations and the real data's autocorrelations on the WWT dataset as a function of training set size. MSE is sensitive to training set size—it decreases by 60% as the training data grows by 2 orders of magnitude. However, Table 5.3 shows that DG trained on 500 data points (the size that gives DG the worst performance) still outperforms all baselines trained on 50,000 samples in autocorrelation MSE. Fig. 5.10 also illustrates variability between models; due to GAN training instability, different GAN models with the same hyperaparameters can have different fidelity metrics. Such training failures can typically be detected early in the training proccess.

With regards to training time, Table 5.4 lists the training time for DG and other baselines. All models were trained on a single NVIDIA Tesla V100 GPU with 16GB GPU memory and an Intel Xeon Gold 6148 CPU with 24GB RAM. These implementations have not been optimized for performance at all, but we find that on the WWT dataset, DG requires 17 hours on average to train, which is 3.4× slower than the fastest benchmark (Naive GAN) and 15.2× faster than the slowest benchmark (TimeGAN).

[7] Wasserstein-1 distance is the integrated absolute error between 2 CDFs.

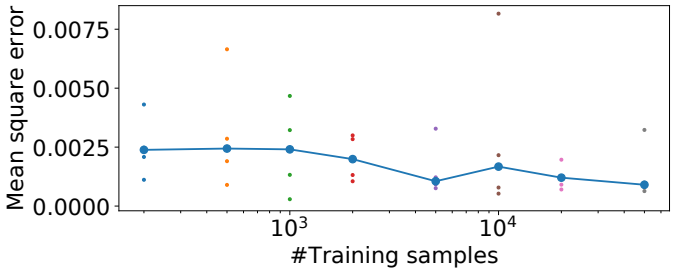


Figure 5.10: Mean square error of autocorrelation of the daily page views v.s. number of training samples for WWT dataset. For each training set size, 5 independent runs are executed and their MSE are plotted in the figure. The line connects the median MSE of the 5 independent runs.

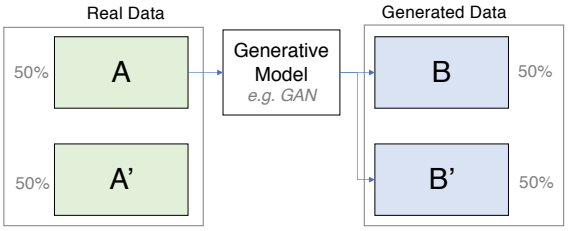


Figure 5.11: Predictive modeling setup: Using training data A, we generate samples $B \cup B'$. Subsequent experiments train downstream tasks on A or B, our training sets, and then test on A' or B'.

Predictive modeling. Given time series measurements, users may want to predict whether an event E occurs in the future, or even forecast the time series itself. For example, in GCT dataset, we could predict whether a particular job will complete successfully. In this use case, we want to show that models trained on generated data generalize to real data.

We first partition our dataset, as shown in Fig. 5.11. We split real data into two sets of equal size: a training set A and a test set A'. We then train a generative model (e.g., DG or a baseline) on training set A. We generate datasets B and B' for training and testing. Finally, we evaluate event prediction algorithms by training a predictor on A and/or B, and testing on A' and/or B'. This allows us to compare the generalization abilities of the prediction algorithms both within a class of data (real/generated), and generalization across classes (train on generated, test on real) [8].

We first predict the task end event type on GCT data (e.g., EVICT, KILL) from time

Method	MSE
RNN	0.1222
AR	0.2777
HMM	0.6030
Naive GAN	0.0190
TimeGAN	0.2384
RCGAN	0.0103
MarketSimulator	0.0324
DoppelGANger	0.0009
DoppelGANger (500 training samples)	0.0024

Table 5.3: Mean square error (MSE) of autocorrelation of the daily page views for WWT dataset (i.e. quantitative presentation of Fig. 5.1). Each model is trained with multiple independent runs, and the median MSE among the runs is presented. Except the last row, all models are trained with 50000 training samples.

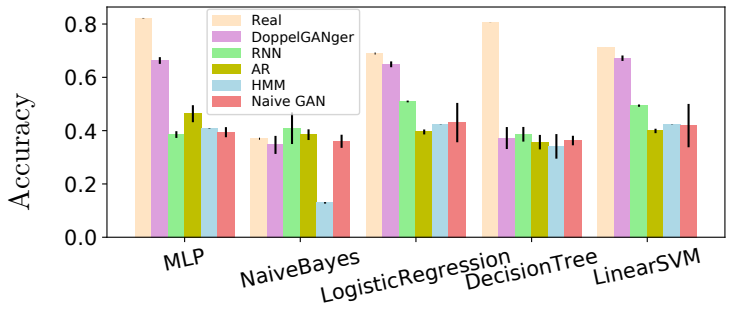


Figure 5.12: Event-type prediction accuracy on GCT.

series observations. Such a predictor may be useful for cluster resource allocators. This prediction task reflects the correlation between the time series and underlying metadata (namely, end event type). For the predictor, we trained various algorithms to demonstrate the generality of our results: multilayer perceptron (MLP), Naive Bayes, logistic regression, decision trees, and a linear SVM. Fig. 5.12 shows the test accuracy of each predictor when trained on generated data and tested on real. Real data expectedly has the highest test accuracy. However, we find that DG performs better than other baselines for all five classifiers. For instance, on the MLP predictive model, DG-generated data has 43% higher accuracy than the next-best baseline (AR), and 80% of the real data accuracy. The results on other datasets are similar.

Method	Average training time (hrs)
Naive GAN	5
Market Simulator	6
HMM	8
RNN	22
RCGAN	29
AR	93
TimeGAN	258
DoppelGANger	17

Table 5.4: Average training time (hours) of synthetic data models on the WWT dataset. All models are trained with 50000 training samples.

Algorithm comparison. We evaluate whether algorithm rankings are preserved on generated data on the GCT dataset by training different classifiers (MLP, SVM, Naive Bayes, decision tree, and logistic regression) to do end event type classification. We also evaluate this on the WWT dataset by training different regression models (MLP, linear regression, and Kernel regression) to do time series forecasting. For this use case, users have only generated data, so we want the ordering (accuracy) of algorithms on real data to be preserved when we train and test them on generated data. In other words, for each class of generated data, we train each of the predictive models on B and test on B'. This is different from Fig. 5.12, where we trained on generated data (B) and tested on real data (A'). We compare this ranking with the ground truth ranking, in which the predictive models are trained on A and tested on A'. We then compute the Spearman's rank correlation coefficient [203], which compares how the ranking in generated data is correlated with the ground-truth ranking. Table 5.5 shows that DG and AR achieve the best rank correlations. This result is misleading because AR models exhibit minimal randomness, so *all* predictors achieve the same high accuracy; the AR model achieves near-perfect rank correlation despite producing low-quality samples; this highlights the importance of considering rank correlation together with other fidelity metrics.

5.3.3 Other Case Studies

DG is being evaluated by several independent users, though DG has not yet been used to release any datasets to the best of our knowledge. A large public cloud provider (IBM) has internally validated the fidelity of DG. IBM stores time series data of resource usage

	DoppelGANger	AR	RNN	HMM	Naive GAN
GCT	**1.00**	**1.00**	**1.00**	0.01	0.90
WWT	**0.80**	**0.80**	0.20	-0.60	-0.60

Table 5.5: Rank correlation of predication algorithms on GCT and WWT dataset. Higher is better.

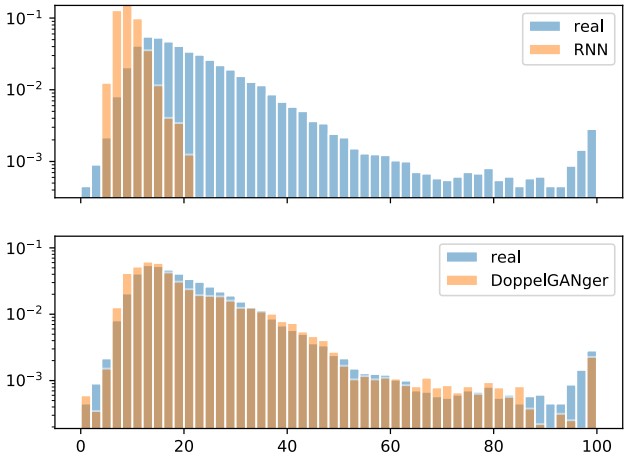


Figure 5.13: Maximum CPU usage.

measurements for different containers used in the cloud's machine learning service. They trained DG to generate resource usage measurements, with the container image name as metadata. Fig. 5.13 shows the learned distribution of containers' maximum CPU usage (in percent). We show the maximum because developers usually size containers according to their peak usage. DG captures this challenging distribution very well, even in the heavy tail.

5.4 Chapter Summary

In this chapter, we present DoppelGANger, a GAN-based data sharing tool for time series data. We show that synthetic data generated by DoppelGANger is statistically similar to the original data in terms of various metrics, such as correlation between fields, temporal correlations, and length distributions. We also show that downstream algorithms trained on synthetic data from DoppelGANger maintain good accuracy when applied to real data.

In addition, the rankings of these downstream algorithms on the synthetic data are similar to their rankings on the original data. These mean that the synthetic data from DoppelGANger has high utility for tasks such as structural characterization, predictive modeling, and algorithm evaluation. We also provide a unified library containing the algorithms proposed in this dissertation, so that they can be used in future applications.

Broader impact. Although in this chapter we primarily use time series data from networking and systems domains for experimental evaluation, the same techniques can be applied to time series data in other domains. In fact, several companies have integrated our algorithms into data sharing products and applications for data in other domains, including banking transactions and road traffic data [50, 51, 52].

Chapter 6

Conclusions and Future Works

6.1 Summary

In this dissertation, we explore how to build a *high-fidelity* and *privacy-preserving* data sharing tool using GANs. We investigate the theoretical foundations of the fidelity and privacy problems in GANs, and we propose general algorithms for improving GANs in these areas. Based on the insights, we build a data sharing tool for time series data as a case study. All the algorithmic innovations in this dissertation are included in a public library [6]. Our work has already been adopted by several companies in their data sharing products [50, 51, 52]. We hope that our efforts will lower the barrier to data sharing and help to fully realize the potential of data in our world.

Excluded work. In addition to the work discussed in this dissertation [166, 204, 205, 206, 12], I also worked related topics during my Ph.D. These include the following:

- *Fidelity foundations.* We explore how to make GANs' latent space disentangled [207] and how to improve GANs fidelity on rare classes [208], fat-tailed distributions [209], and data with noise [210].
- *Applications.* We study the applications of using GANs for security analysis in systems and networks [211, 208] and for sharing network packets and network flows [6].

6.2 Future Work

In addition to the previously-mentioned future work for each component of the dissertation (§ 3.2.4, 3.3.4, 4.2.4 and 4.3.6), we would also like to explore the following open questions.

New use cases enabled by the dissertation. Our work in this dissertation is an enabler for new technology innovations that would not be possible otherwise. For example, network vendors can use our tool to obtain internal network traffic from customers for privacy-preserving troubleshooting. Additionally, companies can utilize our tool to compress and

securely save their data as a trained GAN model. Our tool can also be used by various parties to collaboratively share data for developing and debugging models. We are actively exploring these potential use cases.

Other requirements for data sharing. In this dissertation, we focus on the two main requirements for data sharing: *high-fidelity* and *privacy-preserving*. In practice, there are many other requirements for data sharing. One direction of interest is to enable "what-if" analysis, in which practitioners can model changes in the underlying system and generate associated data. Although our DG makes some types of what-if analysis easy (e.g., slightly altering the attribute distribution), larger changes may alter the physical system model in ways that invalidate the conditional distributions learned by distributional privacy (e.g., imagine simulating a high-traffic regime with a model trained only on low-traffic-regime data). Such what-if analysis is likely to require physical system modeling or simulation, while GANs may be able to help model individual agent behavior. Another direction is to model the dependencies and correlations between different samples (i.e., x_i in § 2.2). For example, a router failure may result in high delays for all packets that go through it. GANs treat all samples independently, so they are not able to capture such correlations by themselves. To enable the learning of such correlations, a fundamentally different technique may be needed.

Other generative models. In this dissertation, we focus on the use of GANs, which were one of the most promising generative models at the early years of my Ph.D. Since then, other promising generative models and architectures have emerged. For example, diffusion models [72, 73, 74] (§ 2.2)) have outperformed GANs on sample fidelity across a wide range of datasets, while being more diverse and more stable to train. Another example is transformers [212], which have good performance on modeling long sequences of data such as text. These models provide new opportunities for building even better data sharing tools in terms of fidelity.

A new generative model tailored to data sharing. The methodology taken in the dissertation is to take an existing generative model, fix its problems, and then modify it to fit the requirements of data sharing. Since these generative models were *not* originally designed for data sharing applications, this may not be the most "efficient" way to reach our goals. A more interesting and exciting direction would be to design a new generative model scratch that is tailored data sharing applications. This would allow us to incorporate the specific requirements of data sharing (e.g., fidelity, privacy, what-if analysis, correlations between samples) into the design of the model itself.

Appendix A

Proofs from Chapter 3

In this chapter, we give the proofs for Chapter 3.

A.1 Proofs from § 3.2

In this section, we showcase how the region interpretation provides a new proof technique that is simple and tight. This transforms the measure-theoretic problem into a geometric one in a simple 2D compact plane, facilitating the proof of otherwise-challenging results.

A.1.1 Additional Theoretical Analysis

Evolution of total variation distances

In order to generalize the intuition from the above toy examples, we first analyze how the total variation evolves for the set of all pairs (P, Q) that have the same total variation distance τ when unpacked (i.e., when $m = 1$). The solutions to the following optimization problems give the desired upper and lower bounds, respectively, on total variation distance for any distribution pair in this set with a packing degree of m:

$$\begin{aligned} \min_{P,Q} \quad & d_{\mathrm{TV}}(P^m, Q^m) & \max_{P,Q} \quad & d_{\mathrm{TV}}(P^m, Q^m) \\ \text{subject to} \quad & d_{\mathrm{TV}}(P, Q) = \tau & \text{subject to} \quad & d_{\mathrm{TV}}(P, Q) = \tau \,, \end{aligned} \tag{A.1}$$

where the maximization and minimization are over all probability measures P and Q. We give the exact solution in Theorem A.1.1.1, which is illustrated pictorially in Fig. 3.6 (left).

Theorem A.1.1.1. *For all $0 \le \tau \le 1$ and a positive integer m, the solution to the maximization in Eq. (A.1) is $1-(1-\tau)^m$, and the solution to the minimization in Eq. (A.1) is*

$$L(\tau, m) \triangleq \min_{0 \le \alpha \le 1-\tau} d_{\mathrm{TV}}\Big(P_{\mathrm{inner}}(\alpha)^m, Q_{\mathrm{inner}}(\alpha, \tau)^m \Big) \,, \tag{A.2}$$

where $P_{\text{inner}}(\alpha)^m$ *and* $Q_{\text{inner}}(\alpha,\tau)^m$ *are the* m*-th order product distributions of binary random variables distributed as*

$$P_{\text{inner}}(\alpha) = \begin{bmatrix} 1-\alpha, & \alpha \end{bmatrix} , \tag{A.3}$$

$$Q_{\text{inner}}(\alpha,\tau) = \begin{bmatrix} 1-\alpha-\tau, & \alpha+\tau \end{bmatrix} . \tag{A.4}$$

Although this is a simple statement that can be proved in several different ways, we introduce in Appendix A.1.2 a novel geometric proof technique that critically relies on the proposed mode collapse region. This particular technique will allow us to generalize the proof to more complex problems involving mode collapse in Theorem 3.2.1.1, for which other techniques do not generalize. Note that the claim in Theorem A.1.1.1 has nothing to do with mode collapse. Still, the mode collapse region definition (used here purely as a proof technique) provides a novel technique that seamlessly generalizes to prove more complex statements in the following.

For any given value of τ and m, the bounds in Theorem A.1.1.1 are easy to evaluate numerically, as shown below in the left panel of Fig. 3.6. Within this achievable range, some subset of pairs (P, Q) have rapidly increasing total variation, occupying the upper part of the region (shown in red, middle panel of Fig. 3.6), and some subset of pairs (P, Q) have slowly increasing total variation, occupying the lower part as shown in blue in the right panel in Fig. 3.6. In particular, the evolution of the mode-collapse region of a pair of m-th power distributions $\mathcal{R}(P^m, Q^m)$ is fundamentally connected to the strength of mode collapse in the original pair (P, Q). This means that for a mode-collapsed pair (P, Q_1), the mth-power distribution will exhibit a different total variation distance evolution than a non-mode-collapsed pair (P, Q_2). As such, these two pairs can be distinguished by a packed discriminator. Making such a claim precise for a broad class of mode-collapsing and non-mode-collapsing generators is challenging, as it depends on the target P and the generator Q, each of which can be a complex high dimensional distribution, like natural images. The proposed region interpretation, endowed with the hypothesis testing interpretation and the data processing inequalities that come with it, is critical: it enables the abstraction of technical details and provides a simple and tight proof based on *geometric techniques* on two-dimensional regions.

Evolution of total variation distances without mode collapse

For the optimization in Eq. (3.6), it is not possible to have $d_{\mathrm{TV}}(P,Q) > (\delta-\varepsilon)/(\delta+\varepsilon)$ and $\delta+\varepsilon \leq 1$, and satisfy the mode collapse and mode augmentation constraints (see Appendix A.1.4 for a proof). Similarly, it is not possible to have $d_{\mathrm{TV}}(P,Q) > (\delta-\varepsilon)/(2-\delta-\varepsilon)$ and $\delta+\varepsilon \geq 1$, and satisfy the constraints. Hence, the feasible set is empty when $\tau > \max\{(\delta-\varepsilon)/(\delta+\varepsilon), (\delta-\varepsilon)/(2-\delta-\varepsilon)\}$. On the other hand, when $\tau \leq \delta-\varepsilon$, no pairs with total variation distance τ can have (ε,δ)-mode collapse. In this case, the optimization reduces to the simpler one in Eq. (A.1) with no mode collapse constraints. Non-trivial solution exists in the middle regime, i.e. $\delta-\varepsilon \leq \tau \leq \max\{(\delta-\varepsilon)/(\delta+\varepsilon), (\delta-\varepsilon)/(2-\delta-\varepsilon)\}$. The lower bound for this regime, given in equation Eq. (A.8), is the same as the lower bound in Theorem A.1.1.1, except it optimizes over a different range of α values. For a wide range of parameters ε, δ, and τ, those lower bounds will be the same, and even if they differ for some parameters, they differ slightly. This implies that the pairs (P,Q) with weak mode collapse will occupy the bottom part of the evolution of the total variation distances (see Fig. 3.6 right panel), and also will be penalized less under packing. Hence a generator minimizing (approximate) $d_{\mathrm{TV}}(P^m,Q^m)$ is likely to generate distributions with weak mode collapse.

Theorem A.1.1.2. *For all $0 \leq \varepsilon < \delta \leq 1$ and a positive integer m, if $0 \leq \tau < \delta-\varepsilon$, then the maximum and the minimum of Eq. (3.6) are the same as those of the optimization Eq. (A.1) provided in Theorem A.1.1.1.*

If $\delta+\varepsilon \leq 1$ and $\delta-\varepsilon \leq \tau \leq (\delta-\varepsilon)/(\delta+\varepsilon)$ then the solution to the maximization in Eq. (3.6) is

$$U_1(\epsilon,\delta,\tau,m) \triangleq \max_{\alpha+\beta\leq 1-\tau, \frac{\varepsilon\tau}{\delta-\varepsilon}\leq\alpha,\beta} d_{\mathrm{TV}}\Big(P_{\mathrm{outer1}}(\varepsilon,\delta,\alpha,\beta,\tau)^m, Q_{\mathrm{outer1}}(\varepsilon,\delta,\alpha,\beta,\tau)^m\Big), \tag{A.5}$$

where $P_{\mathrm{outer1}}(\varepsilon,\delta,\alpha,\beta,\tau)^m$ and $Q_{\mathrm{outer1}}(\varepsilon,\delta,\alpha,\beta,\tau)^m$ are the m-th order product distributions of discrete random variables distributed as

$$P_{\mathrm{outer1}}(\varepsilon,\delta,\alpha,\beta,\tau) = \left[\tfrac{\alpha(\delta-\varepsilon)-\varepsilon\tau}{\alpha-\varepsilon}, \quad \tfrac{\alpha(\alpha+\tau-\delta)}{\alpha-\varepsilon}, \quad 1-\tau-\alpha-\beta, \quad \beta, \quad 0\right], \text{ and} \tag{A.6}$$

$$Q_{\mathrm{outer1}}(\varepsilon,\delta,\alpha,\beta,\tau) = \left[0, \quad \alpha, \quad 1-\tau-\alpha-\beta, \quad \tfrac{\beta(\beta+\tau-\delta)}{\beta-\varepsilon}, \quad \tfrac{\beta(\delta-\varepsilon)-\varepsilon\tau}{\beta-\varepsilon}\right]. \tag{A.7}$$

The solution to the minimization in Eq. (3.6) is

$$L_2(\tau, m) \triangleq \min_{\frac{\varepsilon\tau}{\delta-\varepsilon} \leq \alpha \leq 1-\frac{\delta\tau}{\delta-\varepsilon}} d_{\mathrm{TV}}\Big(P_{\mathrm{inner}}(\alpha)^m, Q_{\mathrm{inner}}(\alpha, \tau)^m \Big) , \tag{A.8}$$

where $P_{\mathrm{inner}}(\alpha)$ *and* $Q_{\mathrm{inner}}(\alpha, \tau)$ *are defined as in Theorem A.1.1.1.*

If $\delta + \varepsilon > 1$ *and* $\delta - \varepsilon \leq \tau \leq (\delta - \varepsilon)/(2 - \delta - \varepsilon)$ *then the solution to the maximization in Eq. (3.6) is*

$$U_2(\epsilon, \delta, \tau, m) \triangleq \max_{\alpha+\beta \leq 1-\tau, \frac{(1-\delta)\tau}{\delta-\varepsilon} \leq \alpha, \beta} d_{\mathrm{TV}}\Big(P_{\mathrm{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau)^m, Q_{\mathrm{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau)^m \Big) , \tag{A.9}$$

where $P_{\mathrm{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau)^m$ *and* $Q_{\mathrm{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau)^m$ *are the* m*-th order product distributions of discrete random variables distributed as*

$$P_{\mathrm{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau) = \Big[\tfrac{\alpha(\delta-\varepsilon)-(1-\delta)\tau}{\alpha-(1-\delta)}, \quad \tfrac{\alpha(\alpha+\tau-(1-\varepsilon))}{\alpha-(1-\delta)}, \quad 1-\tau-\alpha-\beta, \quad \beta, \quad 0 \Big] , \text{ and} \tag{A.10}$$

$$Q_{\mathrm{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau) = \Big[0, \quad \alpha, \quad 1-\tau-\alpha-\beta, \quad \tfrac{\beta(\beta+\tau-(1-\varepsilon))}{\beta-(1-\delta)}, \quad \tfrac{\beta(\delta-\varepsilon)-(1-\delta)\tau}{\beta-(1-\delta)} \Big] . \tag{A.11}$$

The solution to the minimization in Eq. (3.6) is

$$L_3(\tau, m) \triangleq \min_{\frac{(1-\delta)\tau}{\delta-\varepsilon} \leq \alpha \leq 1-\frac{(1-\varepsilon)\tau}{\delta-\varepsilon}} d_{\mathrm{TV}}\Big(P_{\mathrm{inner}}(\alpha)^m, Q_{\mathrm{inner}}(\alpha, \tau)^m \Big) , \tag{A.12}$$

where $P_{\mathrm{inner}}(\alpha)$ *and* $Q_{\mathrm{inner}}(\alpha, \tau)$ *are defined as in Theorem A.1.1.1.*

If $\tau > \max\{(\delta - \varepsilon)/(\delta + \varepsilon), (\delta - \varepsilon)/(2 - \delta - \varepsilon)\}$*, then the optimization in Eq. (3.6) has no solution and the feasible set is an empty set.*

A proof of this theorem is provided in Appendix A.1.4, which also critically relies on the proposed mode collapse region representation of the pair (P, Q) and the celebrated result by Blackwell from [47]. The solutions in Theorem 3.2.1.2 can be numerically evaluated for any given choices of $(\varepsilon, \delta, \tau)$ as we show in Fig. A.1.

A.1.2 Proof of Theorem A.1.1.1

Note that although the original optimization Eq. (A.1) has nothing to do with mode collapse, we use the mode collapse region to represent the pairs (P, Q) to be optimized over.

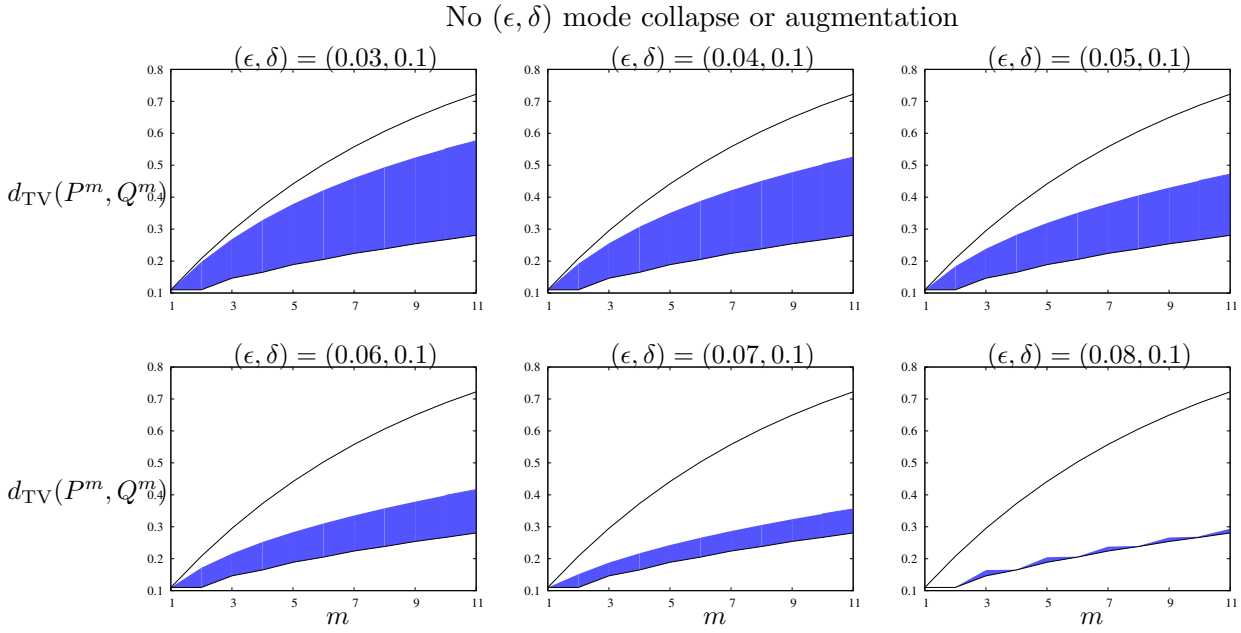


Figure A.1: The evolution of total variation distance over the packing degree m for pairs with no mode collapse/augmentation is shown as a blue band, as defined by the optimization Eq. (3.6) and computed using Theorem 3.2.1.2. For a fixed $d_{\mathrm{TV}}(P, Q) = \tau = 0.11$ and the lack of $(\varepsilon, \delta = 0.1)$-mode collapse/augmentation constraints, we show the evolution with different choices of $\varepsilon \in \{0.03, 0.04, 0.05, 0.06, 0.07, 0.08\}$. The black solid lines show the maximum/minimum total variation in the optimization Eq. (A.1) as a reference. The family of pairs (P, Q) with weaker mode collapse (i.e. larger ε in the constraint), occupies a smaller region at the bottom with smaller total variation under packing, and hence is less penalized when training the generator.

This allows us to use simple geometric techniques to enumerate over all possible pairs (P, Q) that have the same total variation distance τ.

By Remark 3.2.1.2, all pairs (P, Q) that have total variation τ must have a mode collapse region $\mathcal{R}(P, Q)$ that is tangent to the blue line in Fig. A.2. Let us denote a point where $\mathcal{R}(P, Q)$ meets the blue line by the point $(1 - \alpha - \tau, 1 - \alpha)$ in the 2D plane, parametrized by $\alpha \in [0, 1 - \tau]$. Then, for any such (P, Q), we can sandwich the region $\mathcal{R}(P, Q)$ between two regions $\mathcal{R}_{\mathrm{inner}}$ and $\mathcal{R}_{\mathrm{outer}}$:

$$\mathcal{R}_{\mathrm{inner}}(\alpha, \tau) \quad \subseteq \quad \mathcal{R}(P, Q) \quad \subseteq \quad \mathcal{R}_{\mathrm{outer}}(\tau) \,, \tag{A.13}$$

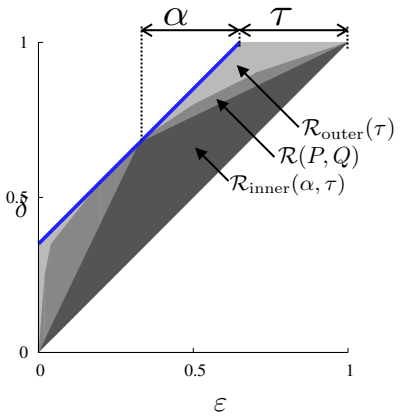


Figure A.2: For any pair (P, Q) with total variation distance τ, there exists an α such that the corresponding region $\mathcal{R}(P, Q)$ is sandwiched between $\mathcal{R}_{\text{inner}}(\alpha, \tau)$ and $\mathcal{R}_{\text{outer}}(\tau)$.

which are illustrated in Fig. A.3. Now, we wish to understand how these inner and outer regions evolve under product distributions. This endeavor is complicated by the fact that there can be infinite pairs of distributions that have the same region $\mathcal{R}(P, Q)$. However, note that if two pairs of distributions have the same region $\mathcal{R}(P, Q) = \mathcal{R}(P', Q')$, then their product distributions will also have the same region $\mathcal{R}(P^m, Q^m) = \mathcal{R}((P')^m, (Q')^m)$. As such, we can focus on the simplest, *canonical* pair of distributions, whose support set has the minimum cardinality over all pairs of distributions with region $\mathcal{R}(P, Q)$.

For a given α, we denote the pairs of canonical distributions achieving these exact inner and outer regions as in Fig. A.3: let $(P_{\text{inner}}(\alpha), Q_{\text{inner}}(\alpha, \tau))$ be as defined in Eq. (A.3) and Eq. (A.4), and let $(P_{\text{outer}}(\tau), Q_{\text{outer}}(\tau))$ be defined as below. Since the outer region has three sides (except for the universal 45-degree line), we only need alphabet size of three to find the canonical probability distributions corresponding to the outer region. By the same reasoning, the inner region requires only a binary alphabet. Precise probability mass functions on these discrete alphabets can be found easily from the shape of the regions and the equivalence to the hypothesis testing region explained in § 3.2.

By the preservation of dominance under product distributions in Remark 3.2.1.5, it follows from the dominance in Appendix A.1.2 that for any (P, Q) there exists an α such that

$$\mathcal{R}(P_{\text{inner}}(\alpha)^m, Q_{\text{inner}}(\alpha, \tau)^m) \;\subseteq\; \mathcal{R}(P^m, Q^m) \;\subseteq\; \mathcal{R}(P_{\text{outer}}(\tau)^m, Q_{\text{outer}}(\tau)^m)\,. \quad \text{(A.14)}$$

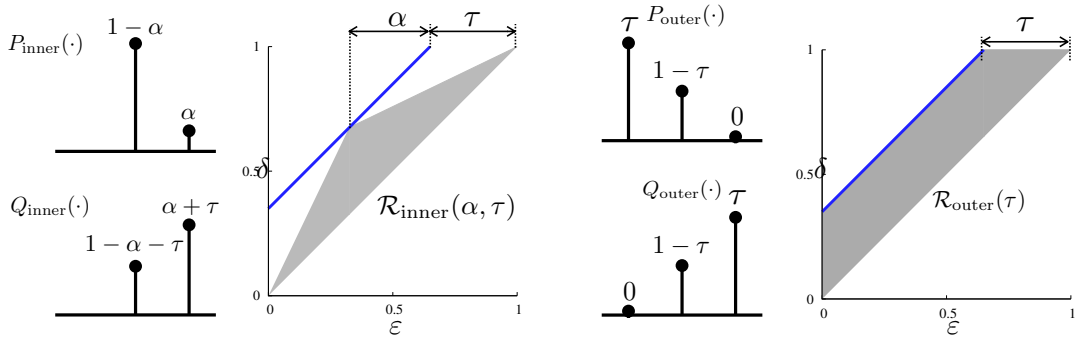


Figure A.3: Canonical pairs of distributions corresponding to $\mathcal{R}_{\text{inner}}(\alpha,\tau)$ and $\mathcal{R}_{\text{outer}}(\tau)$.

Due to the data processing inequality of mode collapse region in Remark 3.2.1.4, it follows that dominance of region implies dominance of total variation distances:

$$\min_{0\le\alpha\le 1-\tau} d_{\text{TV}}(P_{\text{inner}}(\alpha)^m, Q_{\text{inner}}(\alpha,\tau)^m) \;\le\; d_{\text{TV}}(P^m, Q^m) \;\le\; d_{\text{TV}}(P_{\text{outer}}(\tau)^m, Q_{\text{outer}}(\tau)^m) \tag{A.15}$$

The RHS and LHS of the above inequalities can be completely characterized by taking the m-th power of those canonical pairs of distributions. For the upper bound, all mass except for $(1-\tau)^m$ is nonzero only on one of the pairs, which gives $d_{\text{TV}}(P_{\text{outer}}^m, Q_{\text{outer}}^m) = 1-(1-\tau)^m$. For the lower bound, writing out the total variation gives $L(\tau,m)$ in Theorem A.1.1.1. This finishes the proof of Theorem A.1.1.1.

A.1.3 Proof of Theorem 3.2.1.1

In optimization Eq. (3.1), we consider only those pairs with (ε,δ)-mode collapse. It is simple to see that the outer bound does not change. We only need a new inner bound. Let us denote a point where $\mathcal{R}(P,Q)$ meets the blue line by the point $(1-\alpha-\tau, 1-\alpha)$ in the 2D plane, parametrized by $\alpha\in[0,1-\tau]$. We consider the case where $\alpha < 1-(\tau\delta/(\delta-\varepsilon))$ for now, and treat the case when α is larger separately, as the analyses are similar but require a different canonical pair of distributions (P,Q) for the inner bound. The additional constraint that (P,Q) has (ε,δ)-mode collapse translates into a geometric constraint that we need to consider all regions $\mathcal{R}(P,Q)$ that include the orange solid circle at point (ε,δ). Then, for any such (P,Q), we can sandwich the region $\mathcal{R}(P,Q)$ between two regions $\mathcal{R}_{\text{inner1}}$

and $\mathcal{R}_{\text{outer}}$:

$$\mathcal{R}_{\text{inner1}}(\varepsilon, \delta, \alpha, \tau) \quad \subseteq \quad \mathcal{R}(P, Q) \quad \subseteq \quad \mathcal{R}_{\text{outer}}(\tau) \ , \tag{A.16}$$

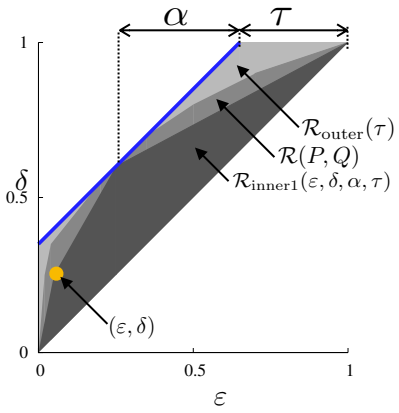


Figure A.4: For any pair (P, Q) with (ε, δ)-mode collapse, the corresponding region $\mathcal{R}(P, Q)$ is sandwiched between $\mathcal{R}_{\text{inner1}}(\varepsilon, \delta, \alpha, \tau)$ and $\mathcal{R}_{\text{outer}}(\tau)$.

Let $(P_{\text{inner1}}(\delta, \alpha), Q_{\text{inner1}}(\varepsilon, \alpha, \tau))$ defined in Eq. (3.2) and Eq. (3.3), and $(P_{\text{outer}}(\tau), Q_{\text{outer}}(\tau))$ defined in Appendix A.1.2 denote the pairs of canonical distributions achieving the inner and outer regions exactly as shown in Fig. A.5. By the preservation of dominance under

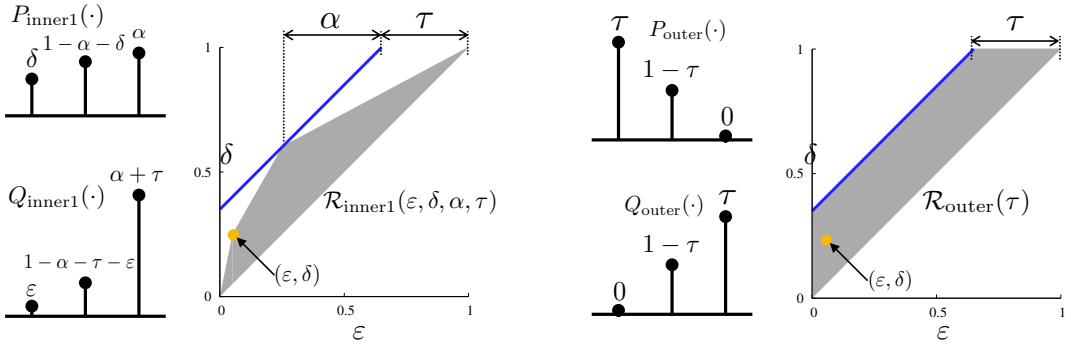


Figure A.5: Canonical pairs of distributions corresponding to $\mathcal{R}_{\text{inner}}(\varepsilon, \delta, \tau, \alpha)$ and $\mathcal{R}_{\text{outer}}(\tau)$.

product distributions in Remark 3.2.1.5, it follows from the dominance in Appendix A.1.3

that for any (P, Q) there exists an α such that

$$\mathcal{R}(P_{\text{inner1}}(\delta,\alpha)^m, Q_{\text{inner1}}(\varepsilon,\delta,\alpha,\tau)^m) \;\subseteq\; \mathcal{R}(P^m, Q^m) \;\subseteq\; \mathcal{R}(P_{\text{outer}}(\tau)^m, Q_{\text{outer}}(\tau)^m) \tag{A.17}$$

Due to the data processing inequality of mode collapse region in Remark 3.2.1.4, it follows that dominance of region implies dominance of total variation distances:

$$\min_{0\le\alpha\le 1-\frac{\tau\delta}{\delta-\varepsilon}} d_{\text{TV}}(P_{\text{inner1}}(\delta,\alpha)^m, Q_{\text{inner1}}(\varepsilon,\delta,\alpha,\tau)^m) \le\; d_{\text{TV}}(P^m, Q^m) \;\le\; d_{\text{TV}}(P_{\text{outer}}(\tau)^m, Q_{\text{outer}}(\tau)^m)\,. \tag{A.18}$$

The RHS and LHS of the above inequalities can be completely characterized by taking the m-th power of those canonical pairs of distributions. For the upper bound, all mass except for $(1-\tau)^m$ is nonzero only on one of the pairs, which gives $d_{\text{TV}}(P^m_{\text{outer}}, Q^m_{\text{outer}}) = 1-(1-\tau)^m$. For the lower bound, writing out the total variation gives $L_1(\varepsilon,\delta,\tau,m)$ in Theorem 3.2.1.1.

For $\alpha > 1-(\tau\delta/(\delta-\varepsilon))$, we need to consider a different class of canonical distributions for the inner region, shown below. The inner region $\mathcal{R}_{\text{inner2}}(\alpha,\tau)$ and corresponding canonical distributions $P_{\text{inner2}}(\alpha)$ and $Q_{\text{inner2}}(\alpha,\tau)$ defined in Eq. (3.4) and Eq. (3.5) are shown below. We take the smaller one between the total variation distance resulting from these two cases. Note that $\alpha \le 1-\tau$ by definition. This finishes the proof of Theorem 3.2.1.1.

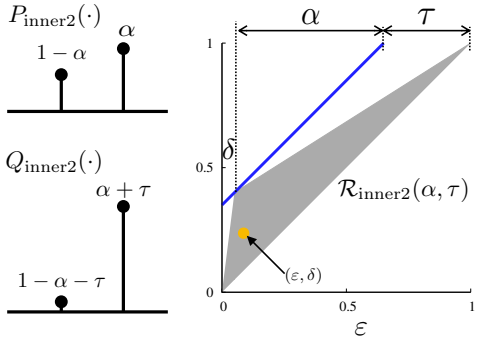


Figure A.6: When $\alpha > 1-(\tau\delta/(\delta-\varepsilon))$, this shows a canonical pair of distributions corresponding to $\mathcal{R}_{\text{inner}}(\varepsilon,\delta,\tau,\alpha)$ for the mode-collapsing scenario $H_1(\varepsilon,\delta,\tau)$.

A.1.4 Proof of Theorem 3.2.1.2

When $\tau < \delta - \varepsilon$, all pairs (P, Q) with $d_{\mathrm{TV}}(P, Q) = \tau$ cannot have (ε, δ)-mode collapse, and the optimization of Eq. (3.6) reduces to that of Eq. (A.1) without any mode collapse constraints.

When $\delta + \varepsilon \leq 1$ and $\tau > (\delta - \varepsilon)/(\delta + \varepsilon)$, no convex region $\mathcal{R}(P, Q)$ can touch the 45-degree line at τ as shown below, and the feasible set is empty. This follows from the fact that a triangle region passing through both (ε, δ) and $(1 - \delta, 1 - \varepsilon)$ will have a total variation distance of $(\delta - \varepsilon)/(\delta + \varepsilon)$. Note that no (ε, δ)-mode augmentation constraint translates into the region not including the point $(1 - \delta, 1 - \varepsilon)$. We can see easily from Fig. A.7 that any total variation beyond that will require violating either the no-mode-collapse constraint or the no-mode-augmentation constraint. Similarly, when $\delta + \varepsilon > 1$ and $\tau > (\delta - \varepsilon)/(2 - \delta - \varepsilon)$, the feasible set is also empty. These two can be unified as $\tau > \max\{(\delta - \varepsilon)/(\delta + \varepsilon), (\delta - \varepsilon)/(2 - \delta - \varepsilon)\}$.

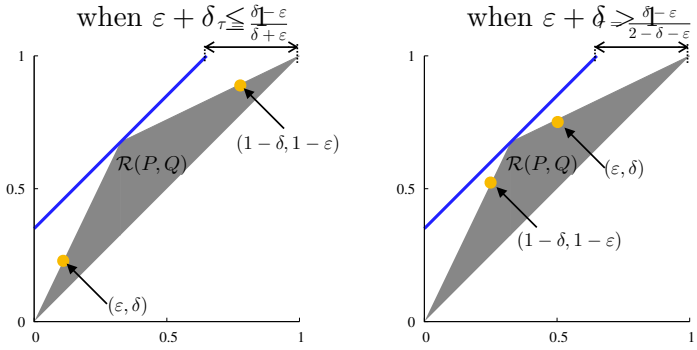


Figure A.7: When $\delta + \varepsilon \leq 1$ and $\tau = (\delta - \varepsilon)/(\delta + \varepsilon)$ (i.e. $(1 - \tau)/2 : (1 + \tau)/2 = \varepsilon : \delta$), a triangle mode collapse region that touches both points (ε, δ) and $(1 - \delta, 1 - \varepsilon)$ at two of its edges also touches the 45-degree line with a τ shift at a vertex (left). When $\delta + \varepsilon > 1$, the same happens when $\tau = (\delta - \varepsilon)/(2 - \delta - \varepsilon)$ (i.e. $(1 - \tau)/2 : (1 + \tau)/2 = (1 - \delta) : (1 - \varepsilon)$). Hence, if $\tau > \max\{(\delta - \varepsilon)/(\delta + \varepsilon), (\delta - \varepsilon)/(2 - \delta - \varepsilon)\}$, then the triangle region that does not include both orange points cannot touch the blue 45-degree line.

Suppose $\delta + \varepsilon \leq 1$, and consider the intermediate regime when $\delta - \varepsilon \leq \tau \leq (\delta - \varepsilon)/(\delta + \varepsilon)$. In optimization Eq. (3.6), we consider only those pairs with no (ε, δ)-mode collapse or (ε, δ)-mode augmentation. It is simple to see that the inner bound does not change from optimization in Eq. (A.1). Let us denote a point where $\mathcal{R}(P, Q)$ meets the blue line by the point $(1 - \alpha' - \tau, 1 - \alpha')$ in the 2D plane, parametrized by $\alpha' \in [0, 1 - \tau]$.

The $\mathcal{R}(\alpha', \tau)$ defined in Fig. A.3 works in this case also. We only need a new outer bound.

We construct an outer bound region, according to the following rule. We fit a hexagon where one edge is the 45-degree line passing through the origin, one edge is the vertical axis, one edge is the horizontal line passing through $(1, 1)$, one edge is the 45-degree line with shift τ shown in blue in Fig. A.8, and the remaining two edges include the two orange points, respectively, at (ε, δ) and $(1-\delta, 1-\varepsilon)$. For any $\mathcal{R}(P, Q)$ satisfying the constraints in Eq. (3.6), there exists at least one such hexagon that includes $\mathcal{R}(P, Q)$. We parametrize the hexagon by α and β, where $(\alpha, \tau + \alpha)$ denotes the left-most point where the hexagon meets the blue line, and $(1 - \tau - \beta, 1 - \beta)$ denotes the right-most point where the hexagon meets the blue line.

The additional constraint that (P, Q) has no (ε, δ)-mode collapse or (ε, δ)-mode augmentation translates into a geometric constraint that we need to consider all regions $\mathcal{R}(P, Q)$ that does not include the orange solid circle at point (ε, δ) and $(1-\delta, 1-\varepsilon)$. Then, for any such (P, Q), we can sandwich the region $\mathcal{R}(P, Q)$ between two regions $\mathcal{R}_{\text{inner}}$ and $\mathcal{R}_{\text{outer1}}$:

$$\mathcal{R}_{\text{inner}}(\alpha', \tau) \quad \subseteq \quad \mathcal{R}(P, Q) \quad \subseteq \quad \mathcal{R}_{\text{outrer1}}(\varepsilon, \delta, \alpha, \beta, \tau) \,, \tag{A.19}$$

where $\mathcal{R}_{\text{inner}}(\alpha, \tau)$ is defined as in Fig. A.3.

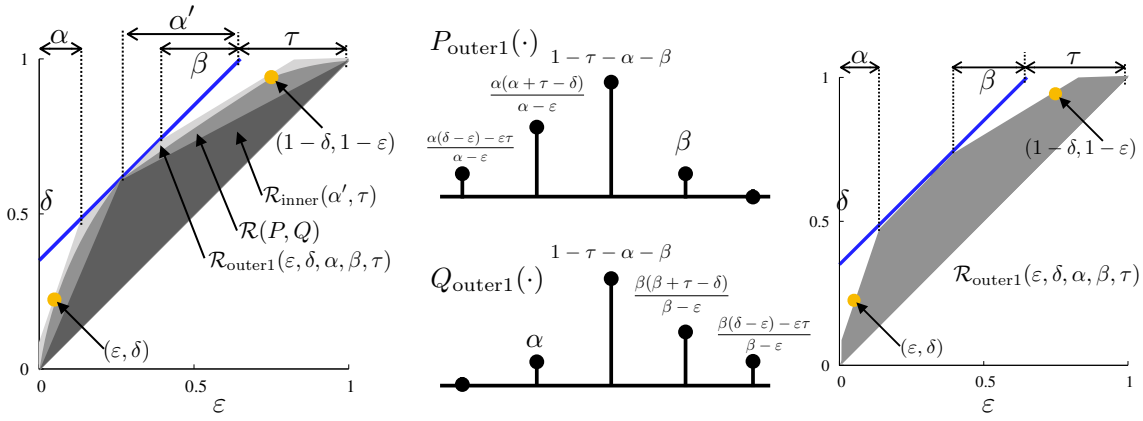


Figure A.8: For any pair (P, Q) with no (ε, δ)-mode collapse or no (ε, δ)-mode augmentation, the corresponding region $\mathcal{R}(P, Q)$ is sandwiched between $\mathcal{R}_{\text{inner}}(\alpha', \tau)$ and $\mathcal{R}_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau)$ (left). A canonical pair of distributions corresponding to $\mathcal{R}_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau)$ (middle and right).

Let $(P_{\text{inner}}(\alpha'), Q_{\text{inner}}(\alpha', \tau))$ defined in Eq. (A.3) and Eq. (A.4), and $(P_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau), Q_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta$ denote the pairs of canonical distributions achieving the inner and outer regions exactly as shown in Fig. A.8.

By the preservation of dominance under product distributions in Remark 3.2.1.5, it follows from the dominance in Appendix A.1.4 that for any (P, Q) there exist α', α, and β such that

$$\mathcal{R}(P_{\text{inner}}(\alpha')^m, Q_{\text{inner}}(\alpha', \tau)^m) \;\subseteq\; \mathcal{R}(P^m, Q^m) \;\subseteq\; \mathcal{R}(P_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau)^m, Q_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau)^m)\,. \tag{A.20}$$

Due to the data processing inequality of mode collapse region in Remark 3.2.1.4, it follows that dominance of region implies dominance of total variation distances:

$$\begin{aligned} \min_{\frac{\varepsilon\tau}{\delta-\varepsilon} \le \alpha' \le 1-\frac{\tau\delta}{\delta-\varepsilon}} d_{\text{TV}}(P_{\text{inner}}(\alpha')^m, Q_{\text{inner}}(\alpha', \tau)^m) \;&\le\; d_{\text{TV}}(P^m, Q^m) \\ &\le \max_{\alpha,\beta \ge \frac{\varepsilon\tau}{\delta-\varepsilon}, \alpha+\beta \le 1-\tau} d_{\text{TV}}(P_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau)^m, Q_{\text{outer1}}(\varepsilon, \delta, \alpha, \beta, \tau)^m)\,. \end{aligned} \tag{A.21}$$

The RHS and LHS of the above inequalities can be completely characterized by taking the m-th power of those canonical pairs of distributions, and then taking the respective minimum over α' and maximum over α and β. For the upper bound, this gives $U_1(\epsilon, \delta, \tau, m)$ in Eq. (A.5), and for the lower bound this gives $L_2(\tau, m)$ in Eq. (A.8).

Now, suppose $\delta + \varepsilon > 1$, and consider the intermediate regime when $\delta - \varepsilon \le \tau \le (\delta - \varepsilon)/(2 - \delta - \varepsilon)$. We have a different outer bound $\mathcal{R}_{\text{outer2}}(\varepsilon, \delta, \alpha, \delta, \tau)$ as the role of (ε, δ) and $(1 - \delta, 1 - \varepsilon)$ have switched. A similar analysis gives

$$d_{\text{TV}}(P^m, Q^m) \;\le \max_{\alpha,\beta \ge \frac{(1-\delta)\tau}{\delta-\varepsilon}, \alpha+\beta \le 1-\tau} d_{\text{TV}}(P_{\text{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau)^m, Q_{\text{outer2}}(\varepsilon, \delta, \alpha, \beta, \tau)^m)\,, \tag{A.22}$$

where the canonical distributions are shown in Fig. A.9 and defined in Eq. (A.10) and Eq. (A.11). This gives $U_2(\epsilon, \delta, \tau, m)$ in Eq. (A.9). For the lower bound we only need to change the range of α we minimize over, which gives $L_3(\tau, m)$ in Eq. (A.12).

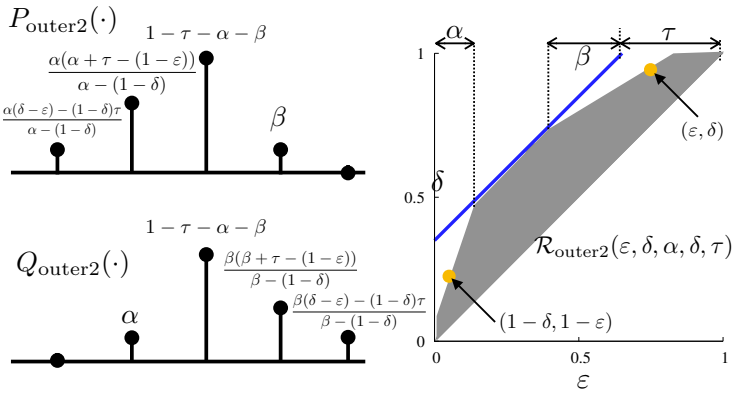


Figure A.9: A canonical pair of distributions corresponding to $\mathcal{R}_{\text{outer2}}(\varepsilon,\delta,\alpha,\beta,\tau)$.

A.2 Proofs from § 3.3

A.2.1 Additional Theoretical Analysis

Additional Analysis of Gradient

In § 3.3.1, we discuss the gradients with respect to $w_i' = \frac{w_i}{u_i^T w_i v_i}$, where u_i, v_i are the singular vectors corresponding to the largest singular values. In this section we discuss the gradients with respect the actual parameter w_i. From Eq. (12) in [43] we know

$$\nabla_{w_t} D_\eta(x) = \frac{1}{\|w_t\|_{\text{sp}}} \left(\nabla_{w_t'} D_\eta(x) - \left(\left(\nabla_{o_l^t(x)} D_\eta(x) \right)^T o_l^t(x) \right) \cdot u_t v_t^T \right)$$

From Appendix A.2.2, we know that $\left\| \nabla_{w_t'} D_\eta(x) \right\|_{\text{F}}$, $\left\| \nabla_{o_l^t(x)} D_\eta(x) \right\|$, and $\left\| o_l^t(x) \right\|$ have upper bounds. Furthermore, $\left\| u_t v_t^T \right\|_{\text{F}} = 1$. Therefore, $\left\| \nabla_{w_t'} D_\eta(x) - \left(\left(\nabla_{o_l^t(x)} D_\eta(x) \right)^T o_l^t(x) \right) \cdot u_t v_t^T \right\|_{\text{F}}$ has an upper bound. From Theorem 1.1 in [213] we know that if w_t is initialized with i.i.d random variables from uniform or Gaussian distribution, $\mathbb{E}\left(\|w_t\|_{\text{sp}} \right)$ is lower bounded away from zero at initialization. So $\|\nabla_{w_t} D_\eta(x)\|_{\text{F}}$ is upper bounded at initialization. Moreover, we observe empirically that $\|w_t\|_{\text{sp}}$ is usually increasing during training. Therefore, $\|\nabla_{w_t} D_\eta(x)\|_{\text{F}}$ is typically upper bounded during training as well.

A.2.2 Proof of Proposition 3.3.1.1

The proposition makes use of the following observation: For the discriminator defined in (3.7), the norm of gradient for w_t is upper bounded by

$$\left\|\nabla_{w_t} D_\eta(x)\right\|_{\mathrm{F}} \leq \|x\| \cdot \prod_{i=1}^{L} \|a_i\|_{\mathrm{Lip}} \cdot \prod_{i=1}^{L} \|w_i\|_{\mathrm{sp}} \Big/ \|w_t\|_{\mathrm{sp}} \quad \text{for } \forall t \in [1, L] \tag{A.23}$$

To prove this, for simplicity of notation, let $o_a^i = a_i \circ l_{w_i} \circ \ldots \circ a_1 \circ l_{w_1}$, and $o_l^i = l_{w_i} \circ a_{i-1} \circ \ldots \circ a_1 \circ l_{w_1}$.

It is straightforward to show that the norm of each internal output of discriminator is bounded by

$$\left\|o_a^t(x)\right\| \leq \|x\| \cdot \prod_{i=1}^{t} \|a_i\|_{\mathrm{Lip}} \cdot \prod_{i=1}^{t} \|w_i\|_{\mathrm{sp}} \tag{A.24}$$

and

$$\left\|o_l^t(x)\right\| \leq \|x\| \cdot \prod_{i=1}^{t-1} \|a_i\|_{\mathrm{Lip}} \cdot \prod_{i=1}^{t} \|w_i\|_{\mathrm{sp}}. \tag{A.25}$$

This holds because

$$\left\|o_a^t(x)\right\| = \left\|a_i\left(o_l^t(x)\right)\right\| \leq \|a_i\|_{\mathrm{Lip}} \cdot \left\|o_l^t(x)\right\|$$

and

$$\left\|o_l^t(x)\right\| = \left\|l_{w_i}\left(o_a^{t-1}(x)\right)\right\| \leq \|w_t\|_{\mathrm{sp}} \cdot \left\|o_a^{t-1}(x)\right\| ,$$

from which we can show the desired inequalities by induction.

Next, we observe that the norm of each internal gradient is bounded by

$$\left\|\nabla_{o_a^t(x)} D_\eta(x)\right\| \leq \prod_{i=t+1}^{L} \|a_i\|_{\mathrm{Lip}} \cdot \prod_{i=t+1}^{L} \|w_i\|_{\mathrm{sp}} \tag{A.26}$$

and

$$\left\|\nabla_{o_l^t(x)} D_\eta(x)\right\| \leq \prod_{i=t}^{L} \|a_i\|_{\mathrm{Lip}} \cdot \prod_{i=t+1}^{L} \|w_i\|_{\mathrm{sp}}. \tag{A.27}$$

This holds because

$$\left\|\nabla_{o_a^t(x)} D_\eta(x)\right\| = \left\|w_{t+1}^T \nabla_{o_l^{t+1}(x)} D_\eta(x)\right\| \leq \|w_{t+1}\|_{\text{sp}} \left\|\nabla_{a_l^{t+1}(x)} D_\eta(x)\right\|$$

and

$$\left\|\nabla_{o_l^t(x)} D_\eta(x)\right\| = \left\|\langle \nabla_{o_a^t(x)} D_\eta(x), \left[a_t'(x)|_{x=o_l^t(x)}\right]\rangle\right\| \leq \|a_t\|_{\text{Lip}} \left\|\nabla_{o_a^t(x)} D_\eta(x)\right\| ,$$

from which we can show inequalities Eqs. (A.26) and (A.27) by induction.

Now we have that

$$\begin{aligned}
\|\nabla_{w_t} D_\eta(x)\|_{\text{F}} &= \left\|\nabla_{o_l^t(x)} D_\eta(x) \cdot \left(o_a^{t-1}(x)\right)^{\text{T}}\right\|_{\text{F}} \\
&= \left\|\nabla_{o_l^t(x)} D_\eta(x)\right\| \cdot \left\|o_a^{t-1}(x)\right\| \\
&\leq \prod_{i=t}^{L} \|a_i\|_{\text{Lip}} \cdot \prod_{i=t+1}^{L} \|w_i\|_{\text{sp}} \cdot \|x\| \cdot \prod_{i=1}^{t-1} \|a_i\|_{\text{Lip}} \cdot \prod_{i=1}^{t-1} \|w_i\|_{\text{sp}} \\
&= \|x\| \cdot \prod_{i=1}^{L} \|a_i\|_{\text{Lip}} \cdot \prod_{i=1}^{L} \|w_i\|_{\text{sp}} \Big/ \|w_t\|_{\text{sp}}
\end{aligned}$$

where we use Eqs. (A.24) to (A.27) at the inequality. The upper bound of gradient's Frobenius norm for spectrally-normalized discriminators follows directly.

A.2.3 Proof of Proposition 3.3.1.2

As $l_w(x)$ is a linear transformation, we have $l_{cw}(x) = c \cdot l_w(x)$, and $l_w(cx) = c \cdot l_w(x)$. Moreover, since ReLU and leaky ReLU is linear in $\mathbb{R}_{>0}$ and $\mathbb{R}_{<0}$ region, we have $a_i(cx) = c \cdot a_i(x)$. Therefore, we have

$$\begin{aligned}
D_\eta'(x) &= \left(a_L \circ l_{c_L \cdot w_L} \circ a_{L-1} \circ l_{c_{L-1} \cdot w_{L-1}} \circ \ldots \circ a_1 \circ l_{c_1 \cdot w_1}\right)(x) \\
&= \prod_{i=1}^{L} c_i \cdot \left(a_L \circ l_{w_L} \circ a_{L-1} \circ l_{w_{L-1}} \circ \ldots \circ a_1 \circ l_{w_1}\right)(x) \\
&= D_\eta(x)
\end{aligned}$$

A.2.4 Proof of Theorem 3.3.1.1

For any discriminator $D_\eta = a_L \circ l_{w_L} \circ a_{L-1} \circ l_{w_{L-1}} \circ \ldots \circ a_1 \circ l_{w_1}$, consider $\eta' = \left\{ w'_t \triangleq c_t w_t \right\}_{t=1}^{L}$ with the constraint $\prod_{i=1}^{L} c_i = 1$ and $c_i \in \mathbb{R}^+$. Let $Q = \left\| \nabla_{w'_i} D_{\eta'}(x) \right\|_{\mathrm{F}} \|w'_i\|_{\mathrm{sp}}$. We have

$$\begin{aligned}
\left\| \nabla_{\eta'} D_{\eta'}(x) \right\|_{\mathrm{F}} &= \sqrt{\sum_{i=1}^{L} \left\| \nabla_{w'_i} D_{\eta'}(x) \right\|_{\mathrm{F}}^2} \\
&= \sqrt{\sum_{i=1}^{L} \frac{Q^2}{c_i^2 \|w_i\|_{\mathrm{sp}}^2}} \\
&\geq \sqrt{L \left(\prod_{i=1}^{L} \frac{Q^2}{c_i^2 \|w_i\|_{\mathrm{sp}}^2} \right)^{1/L}} \\
&= \sqrt{L} \cdot Q^{1/L} \cdot \left(\prod_{i=1}^{L} \|w_i\|_{\mathrm{sp}} \right)^{-1/L}
\end{aligned}$$

and the equality is achieved iff $c_i^2 \|w_i\|_{\mathrm{sp}}^2 = c_j^2 \|w_j\|_{\mathrm{sp}}^2$, $\forall i, j \in [1, L]$ according to AM-GM inequality. When $c_i^2 \|w_i\|_{\mathrm{sp}}^2 = c_j^2 \|w_j\|_{\mathrm{sp}}^2$, $\forall i, j \in [1, L]$, we have $c_t = \prod_{i=1}^{L} \|w_i\|_{\mathrm{sp}}^{1/L} \Big/ \|w_t\|_{\mathrm{sp}}$.

A.2.5 Proof of Theorem 3.3.1.2

Main Proof

Since a_{ij} are symmetric random variables, we know $\mathbb{E}\left(\frac{a_{ij}}{\|A\|_{\mathrm{sp}}} \right) = 0$. Further, by symmetry, we have that for any $(i, j) \neq (h, \ell)$, $\mathbb{E}\left(\frac{a_{ij}^2}{\|A\|_{\mathrm{sp}}^2} \right) = \mathbb{E}\left(\frac{a_{h\ell}^2}{\|A\|_{\mathrm{sp}}^2} \right)$. Therefore, we have

$$\mathrm{Var}\left(\frac{a_{ij}}{\|A\|_{\mathrm{sp}}} \right) = \mathbb{E}\left(\frac{a_{ij}^2}{\|A\|_{\mathrm{sp}}^2} \right) = \frac{1}{mn} \cdot \mathbb{E}\left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}{\|A\|_{\mathrm{sp}}^2} \right) = \frac{1}{mn} \cdot \mathbb{E}\left(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|_{\mathrm{sp}}^2} \right)$$

Our approach will be to upper and lower bound the quantity $\frac{1}{mn} \cdot \mathbb{E}\left(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|_{\mathrm{sp}}^2} \right)$.

Upper bound Assume the singular values of A are $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min\{m,n\}}$. We have

$$\frac{1}{mn} \cdot \mathbb{E}\left(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|_{\mathrm{sp}}^2}\right) = \frac{1}{mn} \cdot \mathbb{E}\left(\frac{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2}{\sigma_1^2}\right) \leq \frac{\min\{m,n\}}{mn} = \frac{1}{\max\{m,n\}} \quad ,$$

which gives the desired upper bound.

Lower bound Now for the lower bound, if a_{ij} are drawn from zero-mean Gaussian distribution and $\max\{m,n\} \geq 3$, we have

$$\begin{aligned} & \frac{1}{mn} \cdot \mathbb{E}\left(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|_{\mathrm{sp}}^2}\right) && \text{(A.28)} \\ = & \frac{1}{mn} \cdot \mathbb{E}\left(\frac{1}{\|A\|_{\mathrm{sp}}^2 / \|A\|_{\mathrm{F}}^2}\right) \\ \geq & \frac{1}{mn} \cdot \frac{1}{\mathbb{E}\left(\left\|\frac{A}{\|A\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2\right)} \\ = & \frac{1}{mn} \cdot \frac{1}{\mathbb{E}\left(\|B\|_{\mathrm{sp}}^2\right)} && \text{(A.29)} \end{aligned}$$

where $B \in R^{m \times n}$ is uniformly sampled from the sphere of $m \times n$-dimension unit ball. We use the following lemma to lower bound (A.29).

Lemma A.2.5.1 (Theorem 1.1 in [213]). *Assume $A \in R^{m \times n}$ is uniformly sampled from the sphere of $m \times n$-dimension unit ball. When $\max\{m,n\} \geq 3$, we have*

$$\mathbb{E}\left(\|A\|_{sp}^2\right) \leq K^2 \left(\mathbb{E}\left(\max_{1 \leq i \leq m} \|a_{i\bullet}\|^2\right) + \mathbb{E}\left(\max_{1 \leq j \leq n} \|a_{\bullet j}\|^2\right)\right) \quad ,$$

where K is a constant which does not depend on m, n. Here $a_{i\bullet}$ denotes the i-th row of A, and $a_{\bullet j}$ denotes the j-th column of A.[1]

[1]Note that the original theorem in [213] requires that the entries of A be i.i.d. symmetric random variables, whereas in our case the entries are not i.i.d., as we require $\|A\|_{\mathrm{F}} = 1$. However, the i.i.d. assumption in their proof is only used to ensure that A, $S_{\sigma^{(1)},\epsilon^{(1)}}(A)$, and $S_{\sigma^{(2)},\epsilon^{(2)}}(A)$ have the same distribution, where $\sigma^{(t)}$ for $t = 0, 1$ are vectors of independent random permutations; $\epsilon^{(t)}$ for $t = 0, 1$ are matrices of i.i.d. random variables with equal probability of being ± 1; and $S_{\sigma^{(1)},\epsilon^{(1)}}(A) = \left(\epsilon_{ij}^{(1)} \cdot a_{i,\sigma_i^{(1)}(j)}\right)_{i,j}$ and

We thus have that

$$\frac{1}{mn} \cdot \frac{1}{\mathbb{E}\left(\|B\|_{\text{sp}}^2\right)} \geq \frac{1}{mn} \cdot \frac{1}{K^2\left(\mathbb{E}\left(\max_{1\leq i\leq m}\|b_{i\bullet}\|^2\right)+\mathbb{E}\left(\max_{1\leq j\leq n}\|b_{\bullet j}\|^2\right)\right)}.$$

Hence, we need to upper bound $\mathbb{E}\left(\max_{1\leq i\leq m}\|b_{i\bullet}\|^2\right)$ and $\mathbb{E}\left(\max_{1\leq j\leq n}\|b_{\bullet j}\|^2\right)$. Let $z \in \mathbb{R}^m$ be a vector uniformly sampled from the sphere of m-dimension unit ball. Observe that $z \overset{d}{=} [\|b_{1\bullet}\|, ..., \|b_{m\bullet}\|]$. The following lemma upper bounds the square of the infinity norm of this vector.

Lemma A.2.5.2. *Assume $z = [z_1, z_2, ..., z_n]$ is uniformly sampled from the sphere of n-dimension unit ball, where $n \geq 2$. Then we have*

$$\mathbb{E}\left(\max_{1\leq i\leq n} z_i^2\right) \leq \frac{4\log(n)}{n-1}.$$

(Proof in Appendix A.2.5)

Hence, when $m, n \geq 2$, we have

$$\mathbb{E}\left(\max_{1\leq i\leq m}\|b_{i\bullet}\|^2\right) \leq \frac{4\log(m)}{m-1}$$

Similarly, we have

$$\mathbb{E}\left(\max_{1\leq j\leq n}\|b_{\bullet j}\|^2\right) \leq \frac{4\log(n)}{n-1}$$

$S_{\sigma^{(2)},\epsilon^{(2)}}(A) = \left(\epsilon_{ij}^{(2)} \cdot a_{\sigma_j^{(2)}(i),j}\right)_{i,j}$. Our matrix A satisfies this requirement, and therefore the same theorem holds.

Therefore,

$$\begin{aligned}
&\operatorname{Var}\left(\frac{a_{ij}}{\|A\|_{\mathrm{sp}}}\right) \\
&\geq \frac{1}{mn} \cdot \frac{1}{K^2\left(\frac{4\log(m)}{m-1}+\frac{4\log(n)}{n-1}\right)} \\
&\geq \frac{1}{8K^2} \cdot \frac{1}{n\log(m)+m\log(n)} \\
&\geq \frac{1}{16K^2} \cdot \frac{1}{\max\{m,n\}\log(\min\{m,n\})}
\end{aligned}$$

which gives the result.

Proof of Lemma A.2.5.2

Proof.

$$\begin{aligned}
&\mathbb{E}\left(\max_{1\leq i\leq n} z_i^2\right) \\
&= \int_0^1 \mathbb{P}\left(\max_{1\leq i\leq n} z_i^2 \geq \delta\right) d\delta \\
&\leq \int_0^1 \min\left\{1, n\cdot\mathbb{P}\left(z_1^2\geq\delta\right)\right\} d\delta \qquad \text{(A.30)}
\end{aligned}$$

where (A.30) follows from the union bound. Next, we use the following lemma to upper bound $\mathbb{P}\left(z_1^2 \geq \delta\right)$.

Lemma A.2.5.3. *Assume $z = [z_1, z_2, ..., z_n]$ is uniformly sampled from the sphere of n-dimension unit ball, where $n \geq 2$. Then for $\frac{1}{n} \leq \delta < 1$ and $\forall i \in [1, n]$, we have*

$$\mathbb{P}\left(z_i^2 \geq \delta\right) \leq e^{-\frac{n-1}{2}\cdot\delta+1}.$$

(Proof in Appendix A.2.5). This in turn gives

$$\int_0^1 \min\left\{1, n\cdot\mathbb{P}\left(z_1^2\geq\delta\right)\right\}d\delta \leq \int_0^{\min\left\{1,\frac{2\log(n)+2}{n-1}\right\}} 1\cdot d\delta + \int_{\min\left\{1,\frac{2\log(n)+2}{n-1}\right\}}^1 n\cdot e^{-\frac{n-1}{2}\cdot\delta+1}\cdot d\delta \tag{A.31}$$

$$\leq \begin{cases} 1 & (n\leq 6) \\ \frac{2\log(n)+2}{n-1} - \frac{2n}{n-1}e^{-\frac{n-3}{2}} + \frac{2}{n-1} & (n\geq 7) \end{cases}$$

$$\leq \frac{4\log(n)}{n-1}$$

where Eq. (A.31) follows from Lemma A.2.5.3.

□

Proof of Lemma A.2.5.3

Due to the symmetry of z_i, we only need to prove the inequality for $i=1$ case. Let $x=[x_1,...,x_n]\sim\mathcal{N}(\mathbf{0},I_n)$, where I_n is the identity matrix in n dimension. We know that $\frac{x_1^2}{\sum_{i=1}^n x_i^2}\overset{d}{=}z_1^2$. Therefore, we have

$$\mathbb{P}\left(z_1^2\geq\delta\right)=\mathbb{P}\left(\frac{x_1^2}{\sum_{j=1}^n x_j^2}\geq\delta\right)=\mathbb{P}\left(\frac{x_1^2}{\left(\sum_{i=2}^n x_i^2\right)/(n-1)}\geq\frac{(n-1)\,\delta}{1-\delta}\right)\quad.$$

Note that x_1^2 and $\sum_{i=2}^n x_i^2$ are two independent chi-squared random variables, therefore, we know that $\frac{x_1^2}{\left(\sum_{i=2}^n x_i^2\right)/(n-1)}\sim F(1,n-1)$, where F denotes the central F-distribution. Therefore,

$$\begin{aligned}\mathbb{P}\left(\frac{x_1^2}{\left(\sum_{i=2}^n x_i^2\right)/(n-1)}\geq\frac{(n-1)\,\delta}{1-\delta}\right) &= 1-I_\delta\left(\frac{1}{2},\frac{n-1}{2}\right)\\ &= I_{1-\delta}\left(\frac{n-1}{2},\frac{1}{2}\right)\\ &= \frac{B_{1-\delta}\left(\frac{n-1}{2},\frac{1}{2}\right)}{B\left(\frac{n-1}{2},\frac{1}{2}\right)}\quad,\end{aligned} \tag{A.32}$$

where $I_x(a,b)$ is the regularized incomplete beta function, $B_x(a,b)$ is the incomplete beta function, and $B(a,b)$ is beta function.

For the ease of computation, we take the log of Eq. (A.32). The numerator gives

$$\begin{aligned}
&\log\left(B_{1-\delta}\left(\frac{n-1}{2},\frac{1}{2}\right)\right)\\
&=\log\left(\frac{(1-\delta)^{(n-1)/2}}{(n-1)/2}{}_2F_1\left(\frac{n-1}{2},\frac{1}{2};\frac{n+1}{2};1-\delta\right)\right)\\
&=\frac{n-1}{2}\log\left(1-\delta\right)-\log(n-1)+\log\left({}_2F_1\left(\frac{n-1}{2},\frac{1}{2};\frac{n+1}{2};1-\delta\right)\right)+\log(2)\ ,
\end{aligned}\tag{A.33}$$

where ${}_2F_1\left(\cdot\right)$ is the hypergeometric function. Let $\left(q\right)_i=\begin{cases}1 & (i=0)\\ q(q+1)\ldots(q+i-1) & (i>0)\end{cases}$, we have

$$\begin{aligned}
&{}_2F_1\left(\frac{n-1}{2},\frac{1}{2};\frac{n+1}{2};1-\delta\right)\\
&=\sum_{i=0}^{\infty}\frac{\left(\frac{n-1}{2}\right)_i\left(\frac{1}{2}\right)_i\left(1-\delta\right)^i}{\left(\frac{n+1}{2}\right)_i\cdot i!}\\
&\leq\sum_{i=0}^{\infty}\frac{\left(\frac{1}{2}\right)_i\left(1-\delta\right)^i}{\cdot i!}\\
&=\delta^{-\frac{1}{2}}
\end{aligned}\tag{A.34}$$

Substituting it into Eq. (A.33) gives

$$\log\left(B_{1-\delta}\left(\frac{n-1}{2},\frac{1}{2}\right)\right)\leq\frac{n-1}{2}\log\left(1-\delta\right)-\log\left(n-1\right)-\frac{1}{2}\log\left(\delta\right)+\log(2)\ .\tag{A.35}$$

The log of the denominator of (A.32) is

$$\begin{aligned}
&\log\left(B\left(\frac{n-1}{2},\frac{1}{2}\right)\right)\\
&=\log\left(\frac{\Gamma\left(\frac{n-1}{2}\right)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)}\right)\\
&\geq\log\left(\sqrt{\pi}\cdot\left(\frac{n+1}{2}\right)^{-\frac{1}{2}}\right)\\
&=-\frac{1}{2}\log(n+1)+\frac{1}{2}\log(2)+\frac{1}{2}\log(\pi)\ .
\end{aligned}\tag{A.36}$$

where Γ denotes the Gamma function and we use the Gautschi's inequality: $\frac{\Gamma(x+1)}{\Gamma\left(x+\frac{1}{2}\right)} < (x+1)^{\frac{1}{2}}$ for positive real number x.

Combining Eq. (A.32), Eq. (A.35), and Eq. (A.36) we get

$$
\begin{aligned}
&\log\left(\mathbb{P}\left(\frac{x_1^2}{\left(\sum_{i=2}^n x_i^2\right)/(n-1)} \geq \frac{(n-1)\,\delta}{1-\delta}\right)\right) \\
&\leq \frac{n-1}{2}\log\left(1-\delta\right) - \log\left(n-1\right) + \frac{1}{2}\log(n+1) - \frac{1}{2}\log\left(\delta\right) + \frac{1}{2}\log(2/\pi) \\
&\leq \frac{n-1}{2}\log\left(1-\delta\right) - \frac{1}{2}\log(n-1) - \frac{1}{2}\log(\delta) + \frac{1}{2}\log(6/\pi) \\
&\leq \frac{n-1}{2}\log\left(1-\delta\right) - \frac{1}{2}\log\left(\frac{n-1}{n}\right) + \frac{1}{2}\log(6/\pi) \\
&\leq \frac{n-1}{2}\log\left(1-\delta\right) + \frac{1}{2}\log\frac{12}{\pi} \\
&\leq -\frac{n-1}{2}\cdot\delta + 1
\end{aligned}
$$

Therefore, we have

$$
\mathbb{P}\left(z_1^2 \geq \delta\right) \leq e^{-\frac{n-1}{2}\cdot\delta+1}
$$

A.2.6 Proof of Theorem 3.3.2.1

Let $s_w = c_{in}c_{out}k_wk_h$. Since w_{ij} are symmetric random variables, we know $\mathbb{E}\left(\frac{w_{ij}}{\sigma_w}\right) = 0$. Therefore, we have

$$
\mathrm{Var}\left(\frac{w_{ij}}{\sigma_w}\right) = \mathbb{E}\left(\frac{w_{ij}^2}{\sigma_w^2}\right) = \frac{1}{s_w}\cdot\mathbb{E}\left(\frac{\sum_{i=1}^m\sum_{j=1}^n w_{ij}^2}{\sigma_w^2}\right) = \frac{1}{s_w}\cdot\mathbb{E}\left(\frac{\|w\|_{\mathrm{F}}^2}{\sigma_w^2}\right)
$$

Note that

$$
\begin{aligned}
\frac{1}{s_w}\cdot\mathbb{E}\left(\frac{\|w\|_{\mathrm{F}}^2}{\sigma_w^2}\right) \in \Bigg[&\frac{2}{s_w}\cdot\mathbb{E}\left(\frac{\|w\|_{\mathrm{F}}^2}{\left\|w^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\mathrm{sp}}^2 + \left\|w^{c_{in}\times(c_{out}k_wk_h)}\right\|_{\mathrm{sp}}^2}\right), \\
&\frac{4}{s_w}\cdot\mathbb{E}\left(\frac{\|w\|_{\mathrm{F}}^2}{\left\|w^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\mathrm{sp}}^2 + \left\|w^{c_{in}\times(c_{out}k_wk_h)}\right\|_{\mathrm{sp}}^2}\right)\Bigg] .
\end{aligned}
$$

Assume the singular values of $w^{c_{out}\times(c_{in}k_wk_h)}$ are $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{c_{out}}$, and the

singular values of $w^{c_{in}\times(c_{out}k_wk_h)}$ are $\sigma'_1 \geq \sigma'_2 \geq \ldots \geq \sigma'_{c_{in}}$. We have

$$\begin{aligned}
&\frac{4}{s_w}\cdot\mathbb{E}\left(\frac{\|w\|_{\mathrm{F}}^2}{\left\|w^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\mathrm{sp}}^2+\left\|w^{c_{in}\times(c_{out}k_wk_h)}\right\|_{\mathrm{sp}}^2}\right)\\
=&\frac{4}{s_w}\cdot\mathbb{E}\left(\frac{1}{2}\cdot\frac{\sum_{i=1}^{c_{out}}\sigma_i^2}{\sigma_1^2}+\frac{1}{2}\cdot\frac{\sum_{i=1}^{c_{in}}\sigma_i'^2}{\sigma_1'^2}\right)\leq\frac{2\left(c_{out}+c_{in}\right)}{s_w}=\frac{2}{c_{in}k_wk_h+c_{out}k_wk_h}\quad,
\end{aligned}$$

which gives the desired upper bound.

As for the lower bound, observe that

$$\begin{aligned}
&\frac{2}{s_w}\cdot\mathbb{E}\left(\frac{\|w\|_{\mathrm{F}}^2}{\left\|w^{c_{out}\times(c_{in}k_wk_h)}\right\|_{\mathrm{sp}}^2+\left\|w^{c_{in}\times(c_{out}k_wk_h)}\right\|_{\mathrm{sp}}^2}\right)\\
=&\frac{2}{s_w}\cdot\mathbb{E}\left(\frac{1}{\left\|\frac{w^{c_{out}\times(c_{in}k_wk_h)}}{\|w\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2+\left\|\frac{w^{c_{in}\times(c_{out}k_wk_h)}}{\|w\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2}\right)\\
\geq&\frac{2}{s_w}\cdot\frac{1}{\mathbb{E}\left(\left\|\frac{w^{c_{out}\times(c_{in}k_wk_h)}}{\|w\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2\right)+\mathbb{E}\left(\left\|\frac{w^{c_{in}\times(c_{out}k_wk_h)}}{\|w\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2\right)}
\end{aligned}$$

Then we can follow the same approach in Appendix A.2.5 for bounding $\mathbb{E}\left(\left\|\frac{w^{c_{out}\times(c_{in}k_wk_h)}}{\|w\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2\right)$ and $\mathbb{E}\left(\left\|\frac{w^{c_{in}\times(c_{out}k_wk_h)}}{\|w\|_{\mathrm{F}}}\right\|_{\mathrm{sp}}^2\right)$, which gives the desired lower bound.

Appendix B

Proofs from Chapter 4

In this chapter, we give the proofs for Chapter 4.

B.1 Proofs from § 4.2

B.1.1 Proof of Theorem 4.2.1.1

Assume that the two neighboring datasets are D_0 and D_1, whose empirical distributions are $\hat{\mu}_m^0$ and $\hat{\mu}_m^1$, and the trained generator distributions from Algorithm 4.2.1 are g_0 and g_1 respectively. The proof has two parts. First, we upper bound the distance between g_0 and g_1 by building on prior generalization results. Then, we use this distance to prove a differential privacy guarantee.

Lemma B.1.1.1 (Error of neural-network discriminators, Corollary 3.3 in [155])**.** *Let μ be the real distribution, and $\hat{\mu}_k$ be the empirical distribution of μ on k i.i.d training samples. Define the trained generator from the optimization algorithm as g and assume the optimization error is bounded by τ_{opt}, i.e., $d_{\mathcal{D}}(\hat{\mu}_k \| g) - \inf_{\nu \in \mathcal{G}} d_{\mathcal{D}}(\hat{\mu}_k \| \nu) \leq \tau_{\text{opt}}$. Under assumptions (A2) and (A3), then with probability at least $1 - \xi$ w.r.t. the randomness of training samples, we have*

$$d_{\mathcal{D}}(\mu \| g) \leq \frac{1}{2}\tau_{k,\xi} = \underbrace{\inf_{\nu \in \mathcal{G}} d_{\mathcal{D}}(\mu \| \nu)}_{\text{approximation error}} + \underbrace{\tau_{\text{opt}}}_{\text{optimization error}} + \underbrace{\frac{C_\xi}{\sqrt{k}}}_{\text{generalization error}},$$

where C_ξ is defined in Eq. (4.3).

From this lemma, we know that with high probability, $d_{\mathcal{D}}(\hat{\mu}_m^i \| g_i)$ is small. The next lemma states that $d_{\mathcal{D}}(\hat{\mu}_m^0 \| \hat{\mu}_m^1)$ is also small.

Lemma B.1.1.2. *Under the assumption (A2), for any two neighboring datasets D_0, D_1 with m samples, we have*

$$d_{\mathcal{D}}(\hat{\mu}_m^0 \| \hat{\mu}_m^1) \leq \frac{2\Delta}{m},$$

.

(Proof in Appendix B.1.6.) Next, we use these results to argue that $d_{\mathcal{D}}(g_0 \| g_1)$ is small with high probability.

Lemma B.1.1.3. *Assume we have two training sets D_0 and D_1, and the trained generator distributions using D_0 and D_1 with Algorithm 4.2.1 are g_0 and g_1, respectively. Under the assumption of Lemma B.1.1.1 and Lemma B.1.1.2, we have that with probability at least $1 - 2\xi$,*

$$d_{\mathcal{D}}(g_0 \| g_1) \leq \tau_{k,\xi} + \frac{2\Delta}{m}.$$

(Proof in Appendix B.1.7.) We next use this bound on the integral probability metric to bound Kullback-Leibler (KL) divergence between g_0 and g_1 with the following lemma.

Lemma B.1.1.4. *Given a generator set $\mathcal{G}$ and a discriminator set $\mathcal{D}$ which satisfy assumption (A1), then we have $\forall \nu_1, \nu_2 \in \mathcal{G}$*

$$d_{\mathrm{KL}}(\nu_1 \| \nu_2) + d_{\mathrm{KL}}(\nu_2 \| \nu_1) \leq \Gamma_{\mathcal{D},\mathcal{G}} d_{\mathcal{D}}(\nu_1 \| \nu_2)$$

where $\Gamma_{\mathcal{D},\mathcal{G}}$ is defined in Eq. (4.1) and $d_{\mathrm{KL}}(\cdot\|\cdot)$ is the Kullback–Leibler divergence.

This follows directly from Proposition 2.9 in [155], which states the following. Denote μ's and ν's density functions as ρ_μ and ρ_ν respectively. If $\log(\rho_\mu/\rho_\nu) \in span\mathcal{D}$, then we have

$$d_{\mathrm{KL}}(\mu \| \nu) + d_{\mathrm{KL}}(\nu \| \mu) \leq \|\log(\rho_\mu/\rho_\nu)\|_{\mathcal{D},1} d_{\mathcal{D}}(\mu \| \nu).$$

Note that $\Gamma_{\mathcal{D},\mathcal{G}} = 1$ when generators in $\mathcal{G}$ are invertible neural networks with l layers and discriminator set $\mathcal{D}$ is $(l+2)$-layer neural networks, according to Lemma 4.1 in [150].

Following Lemma B.1.1.4 and Lemma B.1.1.3, immediately we have that with probability at least $1 - 2\xi$,

$$d_{\mathrm{KL}}(g_0 \| g_1) + d_{\mathrm{KL}}(g_1 \| g_0) \leq \Gamma_{\mathcal{D},\mathcal{G}} \left(\tau_{k,\xi} + \frac{2\Delta}{m} \right)$$

and

$$d_{\mathrm{KL}}\left(g_0^n \| g_1^n\right) + d_{\mathrm{KL}}\left(g_1^n \| g_0^n\right) \leq n \cdot \Gamma_{\mathcal{D},\mathcal{G}} \left(\tau_{k,\xi} + \frac{2\Delta}{m}\right)$$

Then, we connect KL divergence with differential privacy:

Lemma B.1.1.5. *If a mechanism M satisfies that for any two neighboring databases D_0 and D_1, $d_{\mathrm{KL}}(p\|q) + d_{\mathrm{KL}}(q\|p) \leq s$, where p, q are the probability measure of $M(D_0)$ and $M(D_1)$ respectively, then M satisfies $(\epsilon, \frac{s}{\epsilon(1-e^{-\epsilon})})$-probabilistic-differential-privacy for all $\epsilon > 0$.*

(Proof in Appendix B.1.8.)

Note that probabilistic differential privacy implies differential privacy (§ 2.2). From the above lemmas, we know that the mechanism in Algorithm 4.2.1 under assumptions (A1)-(A3) satisfies (ϵ, δ)-differential-privacy for any $\epsilon > 0$ and $\delta \geq \frac{n\ \Gamma_{\mathcal{D},\mathcal{G}}}{\epsilon(1-e^{-\epsilon})}\ \left(\frac{2\Delta}{m} + \tau_{k,\xi}\right)$, with probability at least $1 - 2\xi$ over the randomness in Line 1. The final step is to move the low probability of failure (2ξ) into the δ term.

B.1.2 Proof of Proposition 4.2.1.1

The proof has two parts. First, we lower bound the distance between g_0 and g_1 by building on prior generalization results. Then, we use this distance to prove the the lower bound of δ for (ϵ, δ)-differential-privacy.

Because $\Delta' = \sup_{f\in\mathcal{D}} \sup_{x,y\in\mathcal{X}} |f(x) - f(y)|$, we know that there exists $f' \in \mathcal{D}$ and $x_1, y_1 \in \mathcal{X}$ such that $|f'(x_1) - f'(y_1)| \geq \frac{\Delta'}{2}$. Let's construct two databases: $D_0 = \{x_1, ..., x_m\}$ and $D_1 = \{y_1, x_2, ..., x_m\}$ where $x_2, ..., x_m$ are arbitrary samples from $\mathcal{X}$. Then we have

$$d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| \hat{\mu}_m^1\right) = \frac{1}{m} \sup_{f\in\mathcal{D}} \{f(x_1) - f(y_1)\} \geq \frac{\Delta'}{2m}$$

where the proof of the first equality is in Appendix B.1.6.

On the other hand, from Lemma B.1.1.1, we know that with probability at least $1-\xi$, $d_{\mathcal{D}}\left(\hat{\mu}_m^i \| g_i\right) \leq \frac{1}{2}\tau_{k,\xi}$ for $i = 0, 1$. Combining the above, we have the following lemma.

Lemma B.1.2.1. *Assume we have two training sets D_0 and D_1, and the trained generator using D_0 and D_1 with Algorithm 4.2.1 are g_0 and g_1 respectively. Under the assumption*

of Lemma B.1.1.1 and Lemma B.1.1.2, we have that with probability at least $1 - 2\xi$,

$$d_{\mathcal{D}}\left(g_0 \| g_1\right) \geq \frac{\Delta'}{2m} - \tau_{k,\xi}$$

(Proof in Appendix B.1.9.) Note that this lower bound is nonnegative only for $m < {}^{\Delta'}/_{2\tau_{k,\xi}}$.

Now we connect integral probability metric to total variation (TV) distance. Because the discriminators are bounded (A2), we have

$$d_{\mathcal{D}}\left(g_0 \| g_1\right) \leq 2\Delta d_{\mathrm{TV}}\left(g_0 \| g_1\right)$$

where $d_{\mathrm{TV}}\left(g_0 \| g_1\right)$ is the TV distance between g_0 and g_1. From the above, we know that

$$d_{\mathrm{TV}}\left(g_0 \| g_1\right) \geq \frac{\Delta'}{4m\Delta} - \frac{\tau_{k,\xi}}{2\Delta}$$

Finally, we connect TV distance to differential privacy with the following lemma.

Lemma B.1.2.2. *If a mechanism satisfies* (ϵ, δ)*-differential-privacy, then for any two neighboring databases* D_0 *and* D_1*, we have*

$$d_{\mathrm{TV}}\left(p \| q\right) \leq \frac{e^{\epsilon} + 2\delta - 1}{e^{\epsilon} + 1}$$

where p, q *are the probability measure of* $M(D_0)$ *and* $M(D_1)$ *respectively.*

Therefore, we have

$$\delta \geq \frac{\left(e^{\epsilon} + 1\right)\Delta'}{4m\Delta} - \frac{\left(e^{\epsilon} + 1\right)\tau_{k,\xi}}{2\Delta} + 1 - e^{\epsilon}.$$

B.1.3 Proof of Proposition 4.2.1.2

For any x, we have

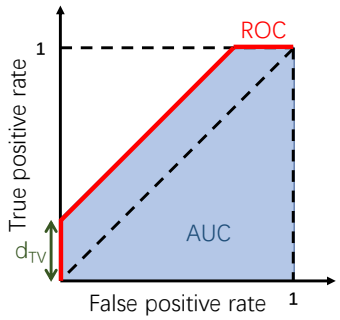


Figure B.1: The upper bound of ROC curves.

$$\begin{aligned}
\rho^{\theta}_{\text{train}}(x) &= \frac{\mathbb{P}(x \in D, \text{parameter is } \theta)}{\mathbb{P}(\text{parameter is } \theta)} \\
&= \frac{\rho_{p_X}(x) \int_{x_2,\ldots,x_m} \prod_{i=2}^{m} \rho_{p_X}(x_i)\, \mathbb{P}(\theta|x, x_2, \ldots, x_m)\, dx_2 \ldots dx_m}{\int_{x_1,\ldots,x_m} \prod_{i=1}^{m} \rho_{p_X}(x_i) \mathbb{P}(\theta|x_1, \ldots, x_m)\, dx_1 \ldots dx_m} \\
&= \rho_{q_\theta}(x)
\end{aligned}$$

B.1.4 Proof of Proposition 4.2.1.3

It is known from the hypothesis testing literature [214] that, for any attack policy, the difference between the true positive rate (TP) and false positive rate (FP) is upper bounded by the total variation (TV) distance $d_{\text{TV}}(q_\theta \| p_X)$:

$$\text{TP} \leq \text{FP} + \min\{d_{\text{TV}}(q_\theta \| p_X), 1 - \text{FP}\} \ . \tag{B.1}$$

Note that total variation distance and ROC curve has a very simple geometric relationship, as noted in [166] (Remark 7). That is, the total variation distance between q_θ and p_X is the intersection between the vertical axis and the tangent line to the upper boundary of the ROC curve that has slope 1 (e.g., see Fig. B.1). This immediately implies that $f(x)$ is an upper bound for all possible ROC curves.

To show tightness, we can construct a g' and μ' as shown in Fig. B.2, such that $d_{\text{TV}}(\mu' \| g') = r$ and they achieve the ROC curve in Fig. B.1.

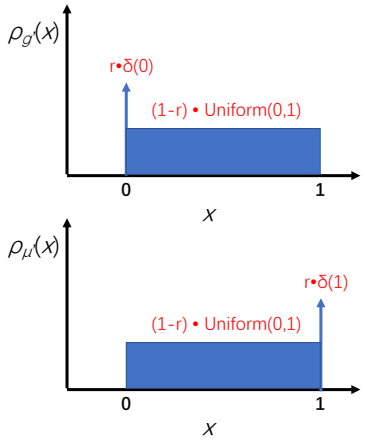


Figure B.2: The pair of distributions that achieve the ROC upper bound.

B.1.5 Proof of Theorem 4.2.1.2

We begin by showing that, under the assumptions in Lemma B.1.1.1 and assuming that $\forall \nu \in \mathcal{G}, \log\left(\mathbb{P}_{p_X}/\mathbb{P}_\nu\right) \in span\mathcal{F}$, we have that with probability at least $1-\delta$ w.r.t. the randomness of training samples,

$$d_{\mathrm{TV}}\left(q_\theta \| p_X\right) \leq \epsilon_{\mathrm{TV}}\left(m, \delta\right) \triangleq \frac{\sqrt{\Xi_{\mathcal{F},\mathcal{G},p_X} \cdot \tau_{m,\delta}}}{2\sqrt{2}} \,. \tag{B.2}$$

To show this, note that Lemma B.1.1.1 gives an upper bound on the integral probability metric between the real and generated distribution. We first connect this distance to the KL divergence with the following lemma.

Lemma B.1.5.1. *Denote the real distribution as μ. Given a generator set $\mathcal{G}$ and a discriminator set $\mathcal{F}$ which satisfy $\forall \nu \in \mathcal{G}, \log\left(\mathbb{P}_\mu/\mathbb{P}_\nu\right) \in span\mathcal{F}$, then we have $\forall q_\theta \in \mathcal{G}$*

$$d_{\mathrm{KL}}\left(q_\theta \| \mu\right) + d_{\mathrm{KL}}\left(\mu \| q_\theta\right) \leq \Xi_{\mathcal{F},\mathcal{G},\mu} d_{\mathcal{D}}\left(\mu \| q_\theta\right)$$

where $\Xi_{\mathcal{F},\mathcal{G},\mu} \triangleq \sup_{\nu \in \mathcal{G}} \left\|\log\left(\mathbb{P}_\mu/\mathbb{P}_\nu\right)\right\|_{\mathcal{D},1}$.

Similar to Lemma B.1.1.4, this lemma relies on Proposition 2.9 in [155]. Furthermore, we use Pinsker's inequality [215] to upper bound the TV distance by the KL distance. Pinsker's inequality says that $d_{\mathrm{TV}}\left(a \| b\right) \leq \sqrt{\frac{1}{2} d_{\mathrm{KL}}\left(a \| b\right)}$ for any two distributions a, b.

Therefore, we have

$$\Xi_{\mathcal{F},\mathcal{G},p_X} d_{\mathcal{D}}\left(p_X \| q_\theta\right) \geq 4 \cdot d_{\mathrm{TV}}\left(q_\theta \| p_X\right)^2$$

Combing this equation with Lemma B.1.1.1 we get the desired inequality.

We can use Eq. (B.2) to upper bound Eq. (B.1) as

$$\mathrm{TP} \leq \mathrm{FP} + \min\left\{\epsilon_{\mathrm{TV}}\left(m, \delta\right), 1 - \mathrm{FP}\right\} . \tag{B.3}$$

Combining Eq. (B.3) with Proposition 4.2.1.3 gives the result.

B.1.6 Proof of Lemma B.1.1.2

Assume that the samples of D_i datasets are $x_1^i, ..., x_m^i$. Without loss of generality, we assume that $x_i^0 = x_i^1$ for $1 \leq i \leq m-1$. Then we have

$$\begin{aligned}
d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| \hat{\mu}_m^1\right) &= \sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x \sim \hat{\mu}_m^0}\left[f(x)\right] - \mathbb{E}_{x \sim \hat{\mu}_m^1}\left[f(x)\right]\right\} \\
&= \sup_{f \in \mathcal{F}} \left\{\frac{1}{m}\sum_{i=1}^m f(x_i^0) - \frac{1}{m}\sum_{i=1}^m f(x_i^1)\right\} \\
&= \frac{1}{m} \sup_{f \in \mathcal{F}} \left\{f(x_m^0) - f(x_m^1)\right\} \\
&\leq \frac{2\Delta}{m}
\end{aligned}$$

B.1.7 Proof of Lemma B.1.1.3

From Lemma B.1.1.1, we know that

$$\begin{aligned}
\mathbb{P}\left[d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| g_0\right) > \frac{1}{2}\tau_{k,\xi}\right] \leq \xi \\
\mathbb{P}\left[d_{\mathcal{D}}\left(\hat{\mu}_m^1 \| g_1\right) > \frac{1}{2}\tau_{k,\xi}\right] \leq \xi
\end{aligned}$$

Therefore,

$$\mathbb{P}\left[d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| g_0\right) \leq \frac{1}{2}\tau_{k,\xi} \wedge d_{\mathcal{D}}\left(\hat{\mu}_m^1 \| g_1\right) \leq \frac{1}{2}\tau_{k,\xi}\right] \geq 1 - 2\xi.$$

With probability at least $1-2\xi$, we have

$$
\begin{aligned}
d_{\mathcal{D}}\left(g_0 \| g_1\right) =& \sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim g_0}[f(x)]-\mathbb{E}_{x \sim g_1}[f(x)]\right\} \\
=& \sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim g_0}[f(x)]-\mathbb{E}_{x \sim \hat{\mu}_m^0}[f(x)]+\mathbb{E}_{x \sim \hat{\mu}_m^0}[f(x)]-\mathbb{E}_{x \sim \hat{\mu}_m^1}[f(x)]+\mathbb{E}_{x \sim \hat{\mu}_m^1}[f(x)]-\mathbb{E}_{x \sim g_1}[f(x)]\right\} \\
\leq & \sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim g_0}[f(x)]-\mathbb{E}_{x \sim \hat{\mu}_m^0}[f(x)]\right\}+\sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim \hat{\mu}_m^0}[f(x)]-\mathbb{E}_{x \sim \hat{\mu}_m^1}[f(x)]\right\} \\
& +\sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim \hat{\mu}_m^1}[f(x)]-\mathbb{E}_{x \sim g_1}[f(x)]\right\} \\
=& \sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim \hat{\mu}_m^0}[f(x)]-\mathbb{E}_{x \sim g_0}[f(x)]\right\}+\sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim \hat{\mu}_m^0}[f(x)]-\mathbb{E}_{x \sim \hat{\mu}_m^1}[f(x)]\right\} \\
& +\sup_{f \in \mathcal{F}}\left\{\mathbb{E}_{x \sim \hat{\mu}_m^1}[f(x)]-\mathbb{E}_{x \sim g_1}[f(x)]\right\} \\
& (\mathcal{F} \text{ is even}) \\
=& d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| g_0\right)+d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| \hat{\mu}_m^1\right)+d_{\mathcal{D}}\left(\hat{\mu}_m^1 \| g_1\right) \\
\leq & \tau_{k, \xi}+\frac{2 \Delta}{m}
\end{aligned}
$$

B.1.8 Proof of Lemma B.1.1.5

Define ρ_p, ρ_q as the probability (density) functions of p and q respectively. Assume set $S_0=\{x: \log \rho_p(x)-\log \rho_q(x) \geq \epsilon\}$, then $\forall x \in S_0$, we have $\rho_p(x) \geq \rho_q(x) e^\epsilon$, and

$$
\begin{aligned}
s \geq & d_{\mathrm{KL}}(p \| q)+d_{\mathrm{KL}}(q \| p) \\
=& \int_x\left(\rho_p(x)-\rho_q(x)\right)\left(\log \rho_p(x)-\log \rho_q(x)\right) \\
\geq & \int_S\left(\rho_p(x)-\rho_q(x)\right)\left(\log \rho_p(x)-\log \rho_q(x)\right) \\
& (\text{because } \left(\rho_p(x)-\rho_q(x)\right)\left(\log \rho_p(x)-\log \rho_q(x)\right) \geq 0 \quad \forall x) \\
\geq & \int_S \rho_p(x)\left(1-e^{-\epsilon}\right) \epsilon
\end{aligned}
$$

i.e. $\mathbb{P}\left[M(D_0) \in S_0\right] \leq \frac{s}{\epsilon(1-e^{-\epsilon})}$. For any set S, we have

$$\begin{aligned}\mathbb{P}\left[M(D_0) \in S \setminus S_0\right] =& \int_{S \setminus S_0} \rho_p(x) dx \\ \leq& \int_{S \setminus S_0} \rho_q(x) e^{\epsilon} dx \\ =& e^{\epsilon} \mathbb{P}\left[M(D_1) \in S \setminus S_0\right]\end{aligned}$$

B.1.9 Proof of Lemma B.1.2.1

Recall that in Appendix B.1.9 we get

$$\mathbb{P}\left[d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| g_0\right) \leq \frac{1}{2}\tau_{k,\xi} \wedge d_{\mathcal{D}}\left(\hat{\mu}_m^1 \| g_1\right) \leq \frac{1}{2}\tau_{k,\xi}\right] \geq 1 - 2\xi.$$

With probability at least $1 - 2\xi$, we have

$$\begin{aligned}d_{\mathcal{D}}\left(g_0 \| g_1\right) =& \sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x \sim g_0}\left[f(x)\right] - \mathbb{E}_{x \sim g_1}\left[f(x)\right]\right\} \\ =& \sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x \sim g_0}\left[f(x)\right] - \mathbb{E}_{x \sim \hat{\mu}_m^0}\left[f(x)\right] + \mathbb{E}_{x \sim \hat{\mu}_m^0}\left[f(x)\right] - \mathbb{E}_{x \sim \hat{\mu}_m^1}\left[f(x)\right] + \mathbb{E}_{x \sim \hat{\mu}_m^1}\left[f(x)\right] - \mathbb{E}_{x \sim g_1}\left[f(x)\right]\right\} \\ \geq& -\sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x \sim g_0}\left[f(x)\right] - \mathbb{E}_{x \sim \hat{\mu}_m^0}\left[f(x)\right]\right\} + \sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x \sim \hat{\mu}_m^0}\left[f(x)\right] - \mathbb{E}_{x \sim \hat{\mu}_m^1}\left[f(x)\right]\right\} \\ & - \sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x \sim \hat{\mu}_m^1}\left[f(x)\right] - \mathbb{E}_{x \sim g_1}\left[f(x)\right]\right\} \\ =& -d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| g_0\right) + d_{\mathcal{D}}\left(\hat{\mu}_m^0 \| \hat{\mu}_m^1\right) - d_{\mathcal{D}}\left(\hat{\mu}_m^1 \| g_1\right) \\ \geq& \frac{\Delta'}{2m} - \tau_{k,\xi}\end{aligned}$$

B.1.10 Proof of Lemma B.1.2.2

Assume that $S = \{x \in X | p(x) > q(x)\}$ and $T = X \setminus S = \{x \in X | p(x) <= q(x)\}$. Let $a_1 = \int_{x \in S} p(x) dx$, $b_1 = \int_{x \in S} q(x) dx$, $a_2 = \int_{x \in T} p(x) dx$, and $b_2 = \int_{x \in T} q(x) dx$. Because of the the differential privacy guarantee, we have

$$\begin{aligned}a_1 - \delta \leq e^{\epsilon} b_1 \\ b_2 - \delta \leq e^{\epsilon} a_2\end{aligned}$$

Note that $a_1 + a_2 = 1$, $b_1 + b_2 = 1$. Therefore, we have

$$b_1 + a_2 \geq \frac{2 - 2\delta}{1 + e^{\epsilon}}$$

and

$$d_{\mathrm{TV}}\left(p \| q\right) = \frac{a_1 + b_2 - b_1 - a_2}{2} \leq \frac{e^{\epsilon} + 2\delta - 1}{e^{\epsilon} + 1}.$$

B.2 Proofs from § 4.3

B.2.1 Proof of Theorem 4.3.2.1

$$\begin{aligned}
T &\geq \Pi_{\epsilon} \\
&= \sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right) \\
&= \sup_{\hat{g}} \mathbb{E}\left(\mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right)\right) \\
&= \mathbb{E}\left(\sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right)\right)
\end{aligned}$$

Therefore, there exists θ' s.t. $\sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right) \leq T$.

Let

$$L_{\theta'} \triangleq \inf_{\theta \in \mathrm{Supp}(p_{\Theta}), z : \mathcal{M}_g(\theta, z) = \theta'} g\left(\theta\right) \ ,$$

$$R_{\theta'} \triangleq \sup_{\theta \in \mathrm{Supp}(p_{\Theta}), z : \mathcal{M}_g(\theta, z) = \theta'} g\left(\theta\right) \ .$$

We can define a sequence of attackers such that $\hat{g}_i\left(\theta'\right) = L_{\theta'} + (i + 0.5) \cdot 2\epsilon$ for $i \in \{0, 1, \ldots, N-1\}$ and $L_{\theta'} + 2N\epsilon \geq R_{\theta'} > L_{\theta'} + 2(N-1)\epsilon$ (Fig. B.3). From the above, we have

$$T \cdot N \geq \sum_i \mathbb{P}\left(\hat{g}_i\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right) \geq 1,$$

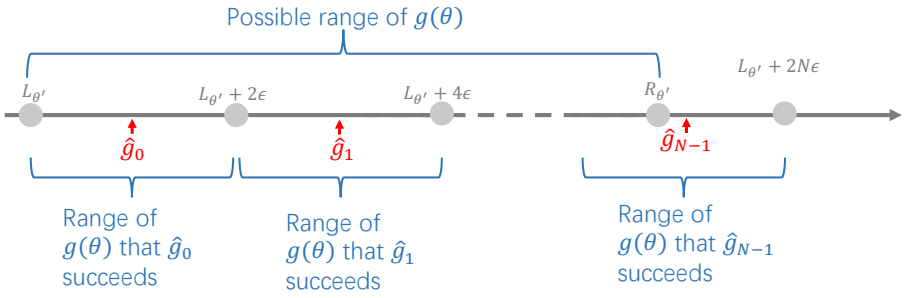


Figure B.3: The construction of attackers for proof of Theorem 4.3.2.1. The 2ϵ ranges of $\hat{g}_0, ..., \hat{g}_{N-1}$ jointly cover the entire range of possible secret $[L_{\theta'}, R_{\theta'}]$. The probability of guessing the secret correclty for any attacker is $\leq T$. Therefore, $R_{\theta'} - L_{\theta'} > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot 2\epsilon$ (Eq. (B.4)).

Therefore, we have $N \geq \lceil \frac{1}{T} \rceil$, and

$$R_{\theta'} - L_{\theta'} > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot 2\epsilon \quad . \tag{B.4}$$

Then we have

$$\begin{aligned}
\Delta &\geq \sup_{\theta \in \text{Supp}(p_\Theta), z \in \text{Supp}(p_Z): \mathcal{M}_g(\theta, z) = \theta'} d\left(p_{X_\theta} \| p_{X_{\theta'}}\right) \\
&\geq \sup_{\theta_i \in \text{Supp}(p_\Theta), z_i: \mathcal{M}_g(\theta_i, z_i) = \theta'} D\left(X_{\theta_1}, X_{\theta_2}\right) \qquad \text{(B.5)} \\
&> \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot 2\gamma\epsilon. \qquad \text{(B.6)}
\end{aligned}$$

where Eq. (B.6) utilizes $R_{\theta'} - L_{\theta'} > \left(\lceil \frac{1}{T} \rceil - 1\right) \cdot 2\epsilon$ and the definition of γ, and Eq. (B.5) comes from triangle inequality.

B.2.2 Proof of Corollary 4.3.4.1

For any $X_{\theta_1}, X_{\theta_2}$, we have

$$\begin{aligned} D\left(X_{\theta_1}, X_{\theta_2}\right) &= \frac{1}{2} d_{\text{Wasserstein-}\alpha}\left(p_{X_{\theta_1}} \| p_{X_{\theta_2}}\right) \\ &\geq \frac{1}{2}\left|g\left(\theta_1\right) - g\left(\theta_2\right)\right| \qquad \text{(B.7)} \\ &= \frac{1}{2} R\left(X_{\theta_1}, X_{\theta_2}\right). \end{aligned}$$

where Eq. (B.7) comes from Jensen inequality. Therefore, we have $\gamma = \inf_{\theta_1,\theta_2 \in \text{Supp}(p_\Theta)} \frac{D\left(X_{\theta_1}, X_{\theta_2}\right)}{R\left(X_{\theta_1}, X_{\theta_2}\right)} \geq \frac{1}{2}$. The result then follows from Theorem 4.3.2.1.

B.2.3 Proof of Proposition 4.3.4.1

For any released parameter $\theta' = (u', v')$, there exists $i \in \{0, ..., N-1\}$ such that $u' = \underline{u} + (i + 0.5) \cdot s$. We have

$$\begin{aligned} &\sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right) \\ &\quad = \sup_{\hat{g}} \int_{\underline{u}+i\cdot s}^{\underline{u}+(i+1)\cdot s} f_{U|U'}\left(u|u'\right) \cdot \int_{u-\epsilon}^{u+\epsilon} f_{\hat{g}(u',v')}\left(h\right) \, \mathrm{d}h \, \mathrm{d}u \\ &\quad = \sup_{\hat{g}} \int_{\underline{u}+i\cdot s-\epsilon}^{\underline{u}+(i+1)\cdot s+\epsilon} f_{\hat{g}(u',v')}(h) \cdot \int_{\hat{g}\left(f_{X_{u',v'}}\right)-\epsilon}^{\hat{g}\left(f_{X_{u',v'}}\right)+\epsilon} f_{U|U'}\left(u|u'\right) \, \mathrm{d}u \, \mathrm{d}h \\ &\quad \leq \sup_{\hat{g}} \int_{\underline{u}+i\cdot s-\epsilon}^{\underline{u}+(i+1)\cdot s+\epsilon} \frac{2\epsilon}{s} \cdot f_{\hat{g}(u',v')}(h) \, \mathrm{d}h \\ &\quad \leq \frac{2\epsilon}{s}. \end{aligned}$$

Therefore, we have

$$\begin{aligned}\Pi_\epsilon &= \sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right) \\ &= \sup_{\hat{g}} \mathbb{E}\left(\mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right)\right) \\ &= \mathbb{E}\left(\sup_{\hat{g}} \mathbb{P}\left(\hat{g}\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right] \middle| \theta'\right)\right) \\ &\leq \frac{2\epsilon}{s}.\end{aligned}$$

For the distortion, we can easily get that $\Delta = \frac{s}{2}$.

B.2.4 Proof of Corollary 4.3.4.2

Let $X_{\lambda_1}, X_{\lambda_2}$ be two exponential random variables. We have

$$\frac{D\left(X_{\lambda_1}, X_{\lambda_2}\right)}{R\left(X_{\lambda_1}, X_{\lambda_2}\right)} = \frac{\frac{1}{2}\left(\lambda_1 - \lambda_2\right)}{-\ln\left(1-\alpha\right)\left(\lambda_1 - \lambda_2\right)} = -\frac{1}{2\ln\left(1-\alpha\right)}. \tag{B.8}$$

Therefore we can get that

$$\gamma = -\frac{1}{2\ln\left(1-\alpha\right)}.$$

B.2.5 Proof of Proposition 4.3.4.2

Both Δ and Π_ϵ are obvious from Eq. (B.8).

B.2.6 Proof of Corollary 4.3.4.3

Let $X_{p_1^1,p_2^1,\ldots,p_C^1}$ and $X_{p_1^2,p_2^2,\ldots,p_C^2}$ be two categorical random variables. We have

$$\begin{aligned} & D\left(X_{p_1^1,p_2^1,\ldots,p_C^1}, X_{p_1^2,p_2^2,\ldots,p_C^2}\right) \\ = & \frac{1}{2} d_{\mathrm{TV}}\left(p_{X_{p_1^1,p_2^1,\ldots,p_C^1}} \| p_{X_{p_1^2,p_2^2,\ldots,p_C^2}}\right) \\ \geq & \frac{1}{2}\left|p_j^1 - p_j^2\right|, \\ & R\left(X_{p_1^1,p_2^1,\ldots,p_C^1}, X_{p_1^2,p_2^2,\ldots,p_C^2}\right) \\ = & \left|p_j^1 - p_j^2\right|. \end{aligned} \tag{B.9}$$

Therefore, we can get that

$$\gamma \geq \frac{1}{2}.$$

B.2.7 Proof of Proposition 4.3.4.3

It is straightforward to get the formula for Δ from Eq. (B.9). Here we focus on the proof for Π_ϵ.

We separate the space of possible data parameters into two regions: $S_1 = \left\{(p_1, \ldots, p_C) | p_i \in \left[\frac{s}{2(C-1)}, 1 - \right.\right.$ and $S_2 = \{(p_1, \ldots, p_C) | p_i \in [0, 1) \ \forall i \in [C] \text{ and } \sum_i p_i = 1\} \setminus S_1$. The high-level idea of our proof is as follows. Note that for any parameter $\theta \in S_1$, there exists a $\mathcal{S}_{p_1,\ldots,p_C}$ s.t. $\theta \in \mathcal{S}_{p_1,\ldots,p_C}$ and $\mathcal{S}_{p_1,\ldots,p_C} \subset S_1$. Therefore, we can bound the attack success rate if $\theta \in S_1$. At the same time, the probability of $\theta \in S_2$ is bounded. Therefore, we can bound the overall attacker's success rate (i.e., Π_ϵ). More specifically, let the optimal attacker be $\hat{g}^*$. We have

$$\begin{aligned} \Pi_\epsilon &= \mathbb{P}\left(\hat{g}^*\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right) \\ &= \int_{\theta \in S_1} p(\theta) \mathbb{P}\left(\hat{g}^*\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right) d\theta \\ &\quad + \int_{\theta \in S_2} p(\theta) \mathbb{P}\left(\hat{g}^*\left(\theta'\right) \in \left[g\left(\theta\right) - \epsilon, g\left(\theta\right) + \epsilon\right]\right) d\theta \\ &< \frac{2\epsilon}{s} + \left(1 - \left(1 - \frac{s}{C-1}\right)^{C-1}\right). \end{aligned}$$

Bibliography

[1] 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*, pp. 2672–2680, 2014. xii, 1, 3, 12, 13, 35, 38, 54

[2] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. xii, 3, 30

[3] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy.," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014. xiv, 5, 50, 64, 68

[4] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Obfuscatory obscanturism: making workload traces of commercially-sensitive systems safe to release," in *2012 IEEE Network Operations and Management Symposium*, pp. 1279–1286, IEEE, 2012. xiv, 10, 64

[5] A. Srivastava, L. Valkov, C. Russell, M. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," *arXiv preprint arXiv:1705.07761*, 2017. xvii, 29, 30, 31, 32

[6] Y. Yin, Z. Lin, M. Jin, G. Fanti, and V. Sekar, "Practical gan-based synthetic ip header trace generation using netshare," in *Proceedings of the ACM SIGCOMM 2022 Conference*, pp. 458–472, 2022. 1, 6, 66, 73, 102

[7] "The caida ucsd anonymized internet traces." https://www.caida.org/catalog/datasets/passive_dataset. Accessed: 2022-01-30. 1, 61

[8] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017. 1, 73, 80, 81, 82, 86, 88, 92, 93, 97

[9] L. R. Warren, J. Clarke, S. Arora, and A. Darzi, "Improving data sharing between acute hospitals in england: an overview of health record system distribution and

retrospective observational analysis of inter-hospital transitions of care," *BMJ open*, vol. 9, no. 12, p. e031637, 2019. 1

[10] E. Chaibub Neto, A. Pratap, T. M. Perumal, M. Tummalacherla, P. Snyder, B. M. Bot, A. D. Trister, S. H. Friend, L. Mangravite, and L. Omberg, "Detecting the impact of subject characteristics on machine learning-based diagnostic applications," *NPJ digital medicine*, vol. 2, no. 1, pp. 1–6, 2019. 1

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009. 1, 8

[12] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Using gans for sharing networked time series data: Challenges, initial promise, and open questions," in *Proceedings of the ACM Internet Measurement Conference*, pp. 464–483, 2020. 1, 9, 52, 60, 66, 73, 102

[13] E. Union, "General data protection regulation." https://gdpr-info.eu/. 2

[14] O. f. C. R. (OCR), "Hipaa home," Dec 2022. 2

[15] J. Wilkes, "Google cluster-usage traces v3," technical report, Google Inc., Mountain View, CA, USA, Apr. 2020. Posted at https://github.com/google/cluster-data/blob/master/ClusterData2019.md. 2, 8, 10, 77

[16] A. Narayanan and V. Shmatikov, "How to break anonymity of the netflix prize dataset," *arXiv preprint cs/0610105*, 2006. 2, 8, 10

[17] J. Sommers, V. Yegneswaran, and P. Barford, "A framework for malicious workload generation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 82–87, 2004. 2, 10

[18] J. Sommers, R. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. Duffield, "Efficient network-wide flow record generation," in *2011 Proceedings IEEE INFOCOM*, pp. 2363–2371, IEEE, 2011. 2, 10, 11

[19] T. Issariyakul and E. Hossain, "Introduction to network simulator 2 (ns2)," in *Introduction to network simulator NS2*, pp. 1–18, Springer, 2009. 2, 10

[20] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 208–221, 2014. 2, 10

[21] S. Di and F. Cappello, "Gloudsim: Google trace based cloud simulator with virtual machines," *Software: Practice and Experience*, vol. 45, no. 11, pp. 1571–1590, 2015. 2, 10

[22] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes, "Workload modeling for resource usage analysis and simulation in cloud computing," *Computers & Electrical Engineering*, vol. 47, pp. 69–81, 2015. 2, 10

[23] L. Sliwko and V. Getov, "Agocs—accurate google cloud simulator framework," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pp. 550–558, IEEE, 2016. 2, 10

[24] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*, pp. 143–154, 2010. 2, 11

[25] V. Tarasov, E. Zadok, and S. Shepler, "Filebench: A flexible framework for file system benchmarking," *USENIX; login*, vol. 41, no. 1, pp. 6–12, 2016. 2, 11

[26] "Fio's documentation." 2, 11

[27] J. Sommers and P. Barford, "Self-configuring network traffic generation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 68–81, ACM, 2004. 2, 11

[28] M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith, "Tmix: a tool for generating realistic tcp application workloads in ns-2," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 3, pp. 65–76, 2006. 2, 11

[29] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu, "Burse: A bursty and self-similar workload generator for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 668–680, 2014. 2, 11

[30] T. Li and J. Liu, "Cluster-based spatiotemporal background traffic generation for network simulation," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 25, no. 1, pp. 1–25, 2014. 2, 11

[31] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pp. 87–92, IEEE, 2010. 2, 11

[32] B. Melamed, J. R. Hill, and D. Goldsman, "The tes methodology: Modeling empirical stationary time series," in *Proceedings of the 24th conference on Winter simulation*, pp. 135–144, ACM, 1992. 2, 11

[33] B. Melamed and J. R. Hill, "Applications of the tes modeling methodology," in *Proceedings of 1993 Winter Simulation Conference-(WSC'93)*, pp. 1330–1338, IEEE, 1993. 2, 11, 92

[34] B. Melamed, "An overview of tes processes and modeling methodology," in *Performance Evaluation of Computer and Communication Systems*, pp. 359–393, Springer, 1993. 2, 11

[35] B. Melamed and D. E. Pendarakis, "Modeling full-length vbr video using markov-renewal-modulated tes models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 5, pp. 600–611, 1998. 2, 11

[36] K. V. Vishwanath and A. Vahdat, "Swing: Realistic and responsive network traffic generation," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 3, pp. 712–725, 2009. 2, 11

[37] Y. Denneulin, E. Romagnoli, and D. Trystram, "A synthetic workload generator for cluster computing," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, p. 243, IEEE, 2004. 2, 11

[38] S. Antonatos, K. G. Anagnostakis, and E. P. Markatos, "Generating realistic workloads for network intrusion detection systems," in *Proceedings of the 4th international workshop on Software and performance*, pp. 207–215, 2004. 2, 11

[39] D.-C. Juan, L. Li, H.-K. Peng, D. Marculescu, and C. Faloutsos, "Beyond poisson: Modeling inter-arrival time of requests in a datacenter," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 198–209, Springer, 2014. 2, 11

[40] S. Di, D. Kondo, and F. Cappello, "Characterizing and modeling cloud applications/jobs on a google data center," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 139–160, 2014. 2, 11

[41] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017. 3, 12, 34

[42] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019. 3

[43] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018. 4, 5, 34, 35, 36, 37, 38, 40, 41, 44, 46, 48, 49, 54, 116

[44] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016. 4, 5, 14, 16, 17, 84

[45] 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*, 2018. 4, 5

[46] A. Srivastava, L. Valkov, 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. 4, 80, 85

[47] D. Blackwell, "Equivalent comparisons of experiments," *The Annals of Mathematical Statistics*, vol. 24, no. 2, pp. 265–272, 1953. 5, 19, 22, 24, 107

[48] F. Farnia, J. M. Zhang, and D. Tse, "Generalizable adversarial training via spectral normalization," *arXiv preprint arXiv:1811.07457*, 2018. 5, 34, 35, 37, 38, 40, 41, 47, 48

[49] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, ACM, 2016. 6

[50] H. Company, "Hazy builds on new technique to generate sequential and time-series synthetic data." https://hazy.com/blog/2020/07/09/how-to-generate-sequential-data, 2022. 6, 101, 102

[51] B. Software, "Synthesizing series of transactions with a Generative Adversarial Network." https://blog.boogiesoftware.com/2020/02/synthesizing-series-of-transactions.html, 2022. 6, 101, 102

[52] Hazy, "Generate Synthetic Time-series Data with Open-source Tools." https://www.kdnuggets.com/2022/06/generate-synthetic-timeseries-data-opensource-tools.html, 2022. 6, 101, 102

[53] H. Ling, K. Kreis, D. Li, S. W. Kim, A. Torralba, and S. Fidler, "Editgan: High-precision semantic image editing," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16331–16345, 2021. 6

[54] K. M. Lewis, S. Varadharajan, and I. Kemelmacher-Shlizerman, "Tryongan: Body-aware try-on via layered interpolation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–10, 2021. 6

[55] X. Huang, A. Mallya, T.-C. Wang, and M.-Y. Liu, "Multimodal conditional image synthesis with product-of-experts gans," in *European Conference on Computer Vision*, pp. 91–109, Springer, 2022. 6, 78

[56] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "Cfa: A practical prediction system for video qoe optimization.," in *NSDI*, pp. 137–150, 2016. 8, 79

[57] A. Manousis, H. Shah, H. Milner, Y. Li, H. Zhang, and V. Sekar, "The shape of view: an alert system for video viewership anomalies," in *Proceedings of the 21st ACM Internet Measurement Conference*, pp. 245–260, 2021. 8, 52, 77

[58] Nuance, "Nuance." https://www.nuance.com/index.html, 2022. 8

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012. 8

[60] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015. 8

[61] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 153–167, 2017. 8

[62] S. Luo, H. Xu, C. Lu, K. Ye, G. Xu, L. Zhang, Y. Ding, J. He, and C. Xu, "Characterizing microservice dependency and performance: Alibaba trace analysis," in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 412–426, 2021. 8

[63] A. Jajoo, Y. C. Hu, X. Lin, and N. Deng, "A case for task sampling based learning for cluster job scheduling," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 19–33, 2022. 8

[64] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi, "Preserving both privacy and utility in network trace anonymization," pp. 459–474, 10 2018. 10

[65] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017. 11

[66] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: A survey revisited," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, pp. 1–43, 2016. 11

[67] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966. 11

[68] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*, pp. 599–619, Springer, 2012. 11

[69] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014. 12

[70] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *International Conference on Machine Learning*, pp. 2722–2730, PMLR, 2019. 12

[71] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013. 12, 58

[72] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020. 12, 103

[73] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, vol. 32, 2019. 12, 103

[74] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*, pp. 2256–2265, PMLR, 2015. 12, 103

[75] 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*, pp. 2672–2680, 2014. 13, 18, 29

[76] 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. 13

[77] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient.," in *AAAI*, pp. 2852–2858, 2017. 13, 80, 82, 83

[78] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances in Neural Information Processing Systems 29*, pp. 613–621, 2016. 13

[79] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, and Z. Wang, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016. 13

[80] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016. 13

[81] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013. 13

[82] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *arXiv preprint arXiv:1807.03039*, 2018. 14

[83] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," *arXiv preprint arXiv:1712.09196*, 2017. 14

[84] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998. 16

[85] 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*, pp. 2234–2242, 2016. 16, 29, 32

[86] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016. 16

[87] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016. 17, 29, 30, 32

[88] V. Nagarajan and J. Z. Kolter, "Gradient descent gan optimization is locally stable," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5591–5600, 2017. 17

[89] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," *IEEE Transactions on Information Theory*, vol. 63, pp. 4037–4049, June 2017. 24

[90] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised representation learning by sorting sequences," in *Proceedings of the IEEE international conference on computer vision*, pp. 667–676, 2017. 29

[91] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998. 29, 30

[92] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015. 29, 32

[93] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016. 29, 32

[94] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016. 29

[95] S. Arora and Y. Zhang, "Do gans actually learn the distribution? an empirical study," *arXiv preprint arXiv:1706.08224*, 2017. 32

[96] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016. 34

[97] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018. 34

[98] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016. 34

[99] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, pp. 3391–3401, 2017. 34

[100] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015. 34, 35, 41

[101] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018. 34, 36, 48

[102] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017. 34, 36, 37, 40

[103] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017. 34, 35, 40, 66, 80, 85

[104] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, pp. 5767–5777, 2017. 34, 35, 36, 38, 40, 46, 80, 85

[105] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, "Improving the improved training of wasserstein gans: A consistency term and its dual effect," *arXiv preprint arXiv:1803.01541*, 2018. 34, 35

[106] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural photo editing with introspective adversarial networks," *arXiv preprint arXiv:1609.07093*, 2016. 34, 40, 46, 48

[107] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in neural information processing systems*, pp. 901–909, 2016. 34, 40, 46, 48

[108] H. Gouk, E. Frank, B. Pfahringer, and M. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *arXiv preprint arXiv:1804.04368*, 2018. 34, 35, 36, 48

[109] Z. Lin, K. K. Thekumparampil, G. Fanti, and S. Oh, "Infogan-cr: Disentangling generative adversarial networks with contrastive regularizers," *ICML*, 2020. 34, 36

[110] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018. 34, 36, 48

[111] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018. 34, 36

[112] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4471–4480, 2019. 34, 36

[113] T. Miyato and M. Koyama, "cgans with projection discriminator," *arXiv preprint arXiv:1802.05637*, 2018. 34, 36

[114] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," *arXiv preprint arXiv:1804.01523*, 2018. 34, 36

[115] 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*, pp. 2234–2242, 2016. 34, 92

[116] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018. 35, 37

[117] H. Jiang, Z. Chen, M. Chen, F. Liu, D. Wang, and T. Zhao, "On computation and generalization of generative adversarial networks under spectrum control," in *International Conference on Learning Representations*, 2019. 35

[118] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017. 35

[119] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks," *arXiv preprint arXiv:1707.09564*, 2017. 35

[120] L. Wang, J. Huang, K. Huang, Z. Hu, G. Wang, and Q. Gu, "Improving neural language generation with spectrum control," in *International Conference on Learning Representations*, 2019. 35

[121] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," *arXiv preprint arXiv:1705.10941*, 2017. 35, 48

[122] A. Odena, J. Buckman, C. Olsson, T. Brown, C. Olah, C. Raffel, and I. Goodfellow, "Is generator conditioning causally related to gan performance?," in *International Conference on Machine Learning*, pp. 3849–3858, 2018. 35

[123] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994. 36

[124] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR, abs/1211.5063*, vol. 2, p. 417, 2012. 36

[125] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, 2013. 36

[126] J. Bernstein, A. Vahdat, Y. Yue, and M.-Y. Liu, "On the distance between two neural networks and the stability of learning," *arXiv preprint arXiv:2002.03432*, 2020. 36

[127] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Efficient BackProp*, pp. 9–50. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. 40, 43

[128] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 6541–6550, 2018. 41

[129] S. Singla and S. Feizi, "Fantastic four: Differentiable and efficient bounds on singular values of convolution layers," in *International Conference on Learning Representations*, 2021. 42

[130] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010. 43

[131] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015. 45

[132] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. 46

[133] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016. 46

[134] ajbrock, "Neural photo editor." https://github.com/ajbrock/Neural-Photo-Editor, 2017. 48

[135] J. Pennington, S. S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice," *arXiv preprint arXiv:1711.04735*, 2017. 48

[136] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013. 48

[137] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*, pp. 1–19, Springer, 2008. 50

[138] S. Meiser, "Approximate and probabilistic differential privacy definitions.," *IACR Cryptology ePrint Archive*, vol. 2018, p. 277, 2018. 51

[139] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *International Conference on Machine Learning*, pp. 5558–5567, 2019. 51, 57, 60, 62

[140] Z. Li and Y. Zhang, "Label-leaks: Membership inference attack with label," *arXiv preprint arXiv:2007.15528*, 2020. 51

[141] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018. 51

[142] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, IEEE, 2019. 51

[143] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018. 51

[144] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, IEEE, 2017. 51

[145] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282, IEEE, 2018. 51

[146] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019. 51, 57, 58, 89

[147] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "Gan-leaks: A taxonomy of membership inference attacks against gans," *arXiv preprint arXiv:1909.03935*, 2019. 51, 57, 58, 60

[148] B. Hilprecht, M. Härterich, and D. Bernau, "Monte carlo and reconstruction membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 232–249, 2019. 51, 58

[149] L. Wasserman and S. Zhou, "A statistical framework for differential privacy," *Journal of the American Statistical Association*, vol. 105, no. 489, pp. 375–389, 2010. 52, 74

[150] Y. Bai, T. Ma, and A. Risteski, "Approximability of discriminators implies diversity in gans," *arXiv preprint arXiv:1806.10586*, 2018. 53, 128

[151] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016. 53

[152] 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, PMLR, 2019. 53

[153] Z. Lin, V. Sekar, and G. Fanti, "Why spectral normalization stabilizes gans: Analysis and improvements," *arXiv preprint arXiv:2009.02773*, 2020. 54

[154] A. Müller, "Integral probability metrics and their generating classes of functions," *Advances in Applied Probability*, vol. 29, no. 2, pp. 429–443, 1997. 54

[155] P. Zhang, Q. Liu, D. Zhou, T. Xu, and X. He, "On the discrimination-generalization tradeoff in gans," *arXiv preprint arXiv:1711.02771*, 2017. 54, 62, 127, 128, 132

[156] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in neural information processing systems*, pp. 271–279, 2016. 58

[157] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016. 58

[158] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 58

[159] "Omie: The online mutual information estimator," 2017.

[160] Google, "Web traffic time series forecasting," 2018. https://www.kaggle.com/c/web-traffic-time-series-forecasting. 60, 73, 90

[161] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 267–280, ACM, 2010. 61

[162] R. Bassily, A. Cheu, S. Moran, A. Nikolov, J. Ullman, and S. Wu, "Private query release assisted by public data," in *International Conference on Machine Learning*, pp. 695–703, PMLR, 2020. 61

[163] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, *et al.*, "Differentially private fine-tuning of language models," *arXiv preprint arXiv:2110.06500*, 2021. 61

[164] T. Liu, G. Vietri, T. Steinke, J. Ullman, and Z. S. Wu, "Leveraging public data for practical private query release," *arXiv preprint arXiv:2102.08598*, 2021. 61

[165] A. Kurakin, S. Chien, S. Song, R. Geambasu, A. Terzis, and A. Thakurta, "Toward training at imagenet scale with differential privacy," 2022. 61

[166] 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. 1505–1514, 2018. 66, 80, 85, 102, 131

[167] J. Jordon, J. Yoon, and M. Van Der Schaar, "Pate-gan: Generating synthetic data with differential privacy guarantees," in *International conference on learning representations*, 2018. 73

[168] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019. 73

[169] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011. 74, 90, 91

[170] F. C. Commission, "Raw data - measuring broadband america - seventh report," 2018. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-seventh. 74, 90

[171] W. Zhang, O. Ohrimenko, and R. Cummings, "Attribute privacy: Framework and mechanisms," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 757–766, 2022. 74

[172] I. Issa, A. B. Wagner, and S. Kamath, "An operational approach to information leakage," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1625–1657, 2019. 77

[173] L. for Computational Physiology, "eicu data." 79

[174] A. PATNE, "Bank transaction data." https://www.kaggle.com/datasets/apoorvwatsky/bank-transaction-data, 2018. 79

[175] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical science*, vol. 17, no. 3, pp. 235–255, 2002. 79

[176] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, "Web usage mining: Discovery and applications of usage patterns from web data," *Acm Sigkdd Explorations Newsletter*, vol. 1, no. 2, pp. 12–23, 2000. 79

[177] T. T. S. Nguyen, H. Y. Lu, and J. Lu, "Web-page recommendation based on web usage and domain knowledge," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2574–2587, 2013. 79

[178] R. Forsati and M. R. Meybodi, "Effective page recommendation algorithms based on distributed learning automata and weighted association rules," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1316–1330, 2010. 79

[179] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE transactions on services Computing*, vol. 5, no. 2, pp. 164–177, 2011. 79

[180] M. Maqableh, H. Karajeh, *et al.*, "Job scheduling for cloud computing using neural networks," *Communications and Network*, vol. 6, no. 03, p. 191, 2014. 79

[181] J. T. Guibas, T. S. Virdi, and P. S. Li, "Synthetic medical images from dual generative adversarial networks," *arXiv preprint arXiv:1709.01872*, 2017. 80

[182] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," *arXiv preprint arXiv:1703.06490*, 2017. 80

[183] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using gan for improved liver lesion classification," in *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pp. 289–293, IEEE, 2018. 80

[184] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "Gan-based synthetic brain mr image generation," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 734–738, IEEE, 2018. 80

[185] W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: better text generation via filling in the_," *arXiv preprint arXiv:1801.07736*, 2018. 80

[186] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, pp. 5509–5519, 2019. 80, 81, 82, 86, 88, 93

[187] J. Yoon, "TimeGAN code repository." https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/02edab3b2b6d635470fa80184bbfd03b8bf8082d/alg/timegan/. 80, 82, 85, 86, 92, 93, 94

[188] E. L. Zec, H. Arnelid, and N. Mohammadiha, "Recurrent conditional gans for time series sensor modelling," in *Time Series Workshop at International Conference on Machine Learning,(Long Beach, California)*, 2019. 81, 82, 83, 86, 88

[189] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016. 82

[190] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 82

[191] De Meer, Fernando, "Generating financial series with generative adversarial networks," 2019. 82

[192] S. Arora and Y. Zhang, "Do gans actually learn the distribution? an empirical study," *arXiv preprint arXiv:1706.08224*, 2017. 83, 96

[193] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012. http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf. 83

[194] R. Fu, J. Chen, S. Zeng, Y. Zhuang, and A. Sudjianto, "Time series simulation by conditional generative adversarial net," *arXiv preprint arXiv:1904.11419*, 2019. 86

[195] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, and C. S. Greene, "Privacy-preserving generative deep neural networks support clinical data sharing," *BioRxiv*, p. 159756, 2017. 88

[196] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989. 91

[197] C. Esteban, S. L. Hyland, and G. Rätsch, "RCGAN code repository." https://github.com/ratschlab/RGAN. 92, 93, 95

[198] H. Buehler, B. Horvath, T. Lyons, I. Perez Arribas, and B. Wood, "A data-driven market simulator for small data environments," *Available at SSRN 3632431*, 2020. 92

[199] 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. 92

[200] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger, "An empirical study on evaluation metrics of generative adversarial networks," *arXiv preprint arXiv:1806.07755*, 2018. 92

[201] 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. 92

[202] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2642–2651, JMLR. org, 2017. 96

[203] C. Spearman, "The proof and measurement of association between two things," *American journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904. 99

[204] Z. Lin, V. Sekar, and G. Fanti, "Why spectral normalization stabilizes gans: Analysis and improvements," *Advances in neural information processing systems*, vol. 34, pp. 9625–9638, 2021. 102

[205] Z. Lin, V. Sekar, and G. Fanti, "On the privacy properties of gan-generated samples," in *International Conference on Artificial Intelligence and Statistics*, pp. 1522–1530, PMLR, 2021. 102

[206] Z. Lin, S. Wang, V. Sekar, and G. Fanti, "Distributional privacy for data sharing," in *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*. 102

[207] Z. Lin, K. Thekumparampil, G. Fanti, and S. Oh, "Infogan-cr and modelcentrality: Self-supervised model training and selection for disentangling gans," in *international conference on machine learning*, pp. 6127–6139, PMLR, 2020. 102

[208] Z. Lin, H. Liang, G. Fanti, V. Sekar, R. A. Sharma, E. Soltanaghaei, A. Rowe, H. Namkung, Z. Liu, D. Kim, *et al.*, "Raregan: Generating samples for rare classes," *arXiv preprint arXiv:2203.10674*, 2022. 102

[209] T. Huster, J. Cohen, Z. Lin, K. Chan, C. Kamhoua, N. O. Leslie, C.-Y. J. Chiang, and V. Sekar, "Pareto gan: Extending the representational power of gans to heavy-tailed distributions," in *International Conference on Machine Learning*, pp. 4523–4532, PMLR, 2021. 102

[210] K. K. Thekumparampil, A. Khetan, Z. Lin, and S. Oh, "Robustness of conditional gans to noisy labels," in *Advances in Neural Information Processing Systems*, pp. 10271–10282, 2018. 102

[211] Z. Lin, S.-J. Moon, C. M. Zarate, R. Mulagalapalli, S. Kulandaivel, G. Fanti, and V. Sekar, "Towards oblivious network analysis using generative adversarial networks," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pp. 43–51, 2019. 102

[212] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. 103

[213] Y. Seginer, "The expected norm of random matrices," *Combinatorics, Probability and Computing*, vol. 9, no. 2, pp. 149–166, 2000. 116, 120

[214] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *International conference on machine learning*, pp. 1376–1385, PMLR, 2015. 131

[215] A. B. Tsybakov, *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008. 132