UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2023-130
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 12/12/2023 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Hailong HU
Born on 15 April 1994 in Sichuan, (China)

PRIVACY ATTACKS AND PROTECTION IN GENERATIVE MODELS

Dissertation defence committee

Dr Jun PANG, dissertation supervisor
Professor, Université du Luxembourg

Dr Mathias HUMBERT
Professor, University of Lausanne

Dr Gabriele LENZINI, Chairman
Professor, Université du Luxembourg

Dr Yang ZHANG, Vice Chairman
Professor, CISPA Helmholtz Center for Information Security

This dissertation was funded in whole by the Luxembourg National Research Fund (FNR)
Grant reference 13550291

Abstract

Privacy Attacks and Protection in Generative Models

by Hailong Hu

Recent years have witnessed the tremendous success of generative models in data synthesis. Typically, a well-trained model itself and its training set constitute key assets for model owners, which allows technology companies to gain a leading position in the global market. However, privacy is a key consideration in deploying state-of-the-art generative models in practice. On the one hand, the exposure of model privacy can lead to the compromise of the intellectual property rights of legitimate model owners, which consequently affects the market share of companies. On the other hand, the disclosure of training data, especially when it includes personal information, constitutes a direct infringement of data privacy, which severely leads to legal sanctions for companies. Indeed, the advent of emerging generative models critically necessitates novel privacy analysis and protection techniques to ensure the confidentiality of cutting-edge models and their training data. To solve these challenges, this dissertation investigates several new privacy attacks and protection methods for generative models from the perspective of model privacy and data privacy. In addition, this dissertation also explores a new mode that leverages existing pre-trained generative models to study the security vulnerabilities of discriminative models, which provides a fresh angle to apply generative models to the risk analysis of discriminative models.

This dissertation is organized into three parts. In the first part, i.e. model privacy in generative models, I develop new model extraction attacks to steal generative adversarial networks (GANs). The evaluations show that preventing model extraction attacks against GANs is difficult but protecting GANs through verifying the ownership can be a deterrence against malicious adversaries. Thus, I further propose an ownership protection method to safeguard GANs, which can effectively recognize these stolen models constructed from physical stealing and model extraction. In the second part, i.e. data privacy in generative models, I develop two types of membership inference attacks against diffusion models, and the proposed loss-based method reveals the relationship between membership inference risks and the generative mechanism of diffusion models. I also investigate property inference risks in diffusion models and propose the first property aware sampling method to mitigate this attack, which bears the benefits of being plug-in and model-agnostic. In the third part, i.e. applications of generative models, I propose a new type of out-of-distribution (OOD) attack by leveraging off-the-shelf pre-trained GANs, which demonstrates that GANs can be utilized to directly construct samples to fool classification models and evade OOD detection. Taken together, this dissertation primarily provides new privacy attacks and protection methods for generative models and can contribute to a deeper and more comprehensive understanding of the privacy of generative artificial intelligence.

Acknowledgements

The Ph.D. journey at the University of Luxembourg is a precious chapter in my life. I could not make it here without the support and guidance from many people.

First and foremost, I am incredibly thankful to my supervisor Dr. Jun Pang for his invaluable guidance throughout my Ph.D. study. He always supported me to explore and develop my research interests. Jun also provided me with strong support and encouragement whenever I met difficulties and felt bewildered. He consistently prioritized my career success, and his advice and help were crucial for my job search.

I am genuinely thankful to Dr. Yang Zhang for his help over the course of my Ph.D. study. Yang's insights and enthusiasm on trustworthy machine learning give me a brand new and inspiring perspective on the privacy in generative models that I work on. These have also inspired and shaped my research directions significantly.

I am sincerely thankful to Dr. Michail Papadakis for being a member in my dissertation supervisory committee during my Ph.D. study.

I am truly thankful to Dr. Gabriele Lenzini and Dr. Mathias Humbert for joining my Ph.D. defense committee.

I deeply thank Dr. Sjouke Mauw for his support during my time as a teaching assistant. He heads the SaToss group with an optimistic and friendly atmosphere. My life in Luxembourg has been greatly enriched by colleagues in the SaToss group.

I profoundly thank all members of the SaToss group. Despite working in a somewhat different field, I was still able to learn a lot from each of them.

I would like to thank the Luxembourg National Research Fund for supporting my dissertation research.

Last but most importantly, I would express my deepest thanks and love to my parents. Their unconditional love and unwavering support fortify my confidence and courage to pursue my goals.

Hailong Hu
Belval Campus, Luxembourg

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial intelligence (AI) technology profoundly powers numerous facets of modern society, ranging from agriculture to manufacturing, transportation to finance, and entertainment to healthcare [LBH15]. At the same time, AI systems entail a number of potential security and privacy risks, such as erroneous decision-making in autonomous driving, exposure of personal health information in AI-assisted medical diagnosis, or being used for malicious purposes. In particular, recent generative AI systems, such as ChatGPT which can generate impressively coherent and readable text, further bring about people's concerns in relation to privacy.

1.1 Generative Models

"*What I cannot create, I do not understand.*"

— Richard Feynman, 1988

Over the past few years, generative AI has undoubtedly made incredible achievements in a variety of domains, such as image synthesis, natural language processing, and audio generation. For instance, diffusion-based generative AI systems, like DALL-E2 [RDN+22] or Stable Diffusion [RBL+22], can synthesize various styles of images that are aesthetically pleasing and visually stunning. Autoregressive-based generative AI systems, such as GPT-3 [BMR+20], excel in generating natural language texts that are lexically diverse and semantically coherent, making it exceedingly challenging to distinguish them from human-authored content. Variational autoencoder-based generative AI systems, like Jukebox [DJP+20], exhibit promising potential in generating music with singing in the raw audio domain. Behind the success of generative AI, one of the key catalysts is the breakthrough of generative models which synergistically combine generative modeling methods and deep neural networks.

Generative models generally include energy-based models [AHS85, LCH+06], autoregressive models [BB99, GGML15], normalizing flows [RM15, DSDB16], variational autoencoders [KW14], generative adversarial networks (GANs) [GPAM+14], and diffusion models [SDWMG15]. The generation performance of generative models can vary, depending on the characteristics of different application domains. For example, text generation in the natural language processing domain mainly relies

on autoregressive models, which operate sequentially predicting subsequent tokens conditioned on prior information [BMR+20]. In contrast, image synthesis in the computer vision domain predominantly utilizes GANs and diffusion models, where they are much more versed in capturing intricate patterns and dependencies within the image data [YZS+22].

GANs, first introduced by Goodfellow et al. [GPAM+14], learn to generate novel samples via an adversarial learning way. A typical GAN comprises two neural networks: a generator and a discriminator. In a GAN, the generator learns to produce data samples that closely resemble these samples from the training set, thereby fooling the discriminator. Conversely, the discriminator, usually designed as a binary classifier, aims to differentiate between samples from the generator and from the training set. The adversarial learning process, i.e. a min-max game, is conducted between the generator and the discriminator until Nash equilibrium is reached. Drawing from the foundational idea of the seminal GAN, researchers have proposed various methods to further improve the generation performance of GANs, such as designing new neural networks [RMC16, KALL18], using novel latent spaces [KLA19, KLA+20], constructing new loss functions [SGZ+16, MKKY18].

Diffusion models, initially proposed by Sohl-Dickstein et al. [SDWMG15], learn to generate novel samples via a way of successively adding the noise and denoising. Specifically, a diffusion model first perturbs training samples to Gaussian noise samples by gradually adding different levels of noise into training samples. Then, it learns to reverse this process to synthesize the noise-free samples from noise samples via step by step removing the noise. Denoising diffusion probabilistic models [HJA20] significantly enhance the generation performance of the original diffusion model and enable it to synthesize high-quality images by parameterization techniques and a new objective function. Additionally, researchers successfully improve the performance of a diffusion model by effectively utilizing the score function of the log-likelihood of the data [SSDK+21]. Unlike GANs which are known for their high sampling speeds but often present challenges in terms of training stability, diffusion models can exhibit robust training stability but typically suffer from a notably slower sampling process. Despite these divergent yet complementary characteristics, GANs and diffusion models are widely appreciated for their excellent performance in image synthesis.

1.2 Privacy in Machine Learning

With the burgeoning prosperity of AI and the profound integration into numerous aspects of our daily lives, people have increasingly drawn attention to privacy and security threats on AI-based applications [PMSW18]. Indeed, research on adversarial example attacks has demonstrated that an adversarial 3D-printed object can cause an autonomous driving system to fail in detecting and crash into it, which exerts direct and formidable threats on security-critical applications [CWX+21]. Recent studies on membership inference attacks have illustrated that various machine

learning (ML) models, including classification models, language models, and generative models, can inadvertently memorize certain training samples, which results in privacy concerns for privacy-sensitive applications [SSSS17, CTW+21, CHN+23].

In general, based on the adversary's goal, we can classify attacks against a machine learning model into two categories: integrity attacks and privacy attacks. Integrity attacks concern how an adversary manipulates an ML model to compromise the model's performance. They can occur in the training phase, such as poisoning attacks [BNS+06] and backdoor attacks [GLDGG19], or in the inference phase, such as evasion attacks [BCM+13, SZS+14], which are usually related to the security or robustness properties of an ML model. In contrast, privacy attacks usually steal or reveal the information of a well-trained ML model including the model itself and the training set.

Privacy attacks targeting to steal the information of an ML model itself include stealing the functions or parameters of an ML model (also called model extraction attacks) [TZJ+16, CJM20], and stealing the hyperparameters of an ML model [WG18, OAFS18]. These types of attacks that infringe the model privacy usually compromise the intellectual property of a model owner. In addition, privacy attacks targeting to steal the information of the training set of an ML model include membership inference attacks [SSSS17], property inference attacks [AMS+15], and data extraction attacks [FJR15]. These types of attacks that infringe the data privacy usually lead to the information leakage of a private training set. For instance, membership inference attacks aim to infer whether a given sample is in the model's training set. These attacks allow adversaries to reveal that one person has this disease by inferring an ML model associated with a disease. As another example, property inference attacks focus on inferring some global information about the model's training set, which is not shared by model owners. These attacks enable adversaries to reveal the proportion of the training dataset from a certain class, such as the fraction of females or minorities on an ML model used for loan scoring.

1.3 Dissertation Motivation

Advanced ML technologies enable enterprises to acquire an advantageous position in the global market. Generally, a well-trained state-of-the-art ML model and its training set are valuable assets of model owners or providers. On the one hand, a well-trained ML model is typically thought the intellectual property of enterprises and should be confidential [KNL+20]. On the other hand, the model's training data associated with personal information should be secret. In particular, an increasing number of legislation, such as the General Data Protection Regulation (GDPR) [PotEU16] and the California Consumer Privacy Act (CCPA) [LoC18], have required that enterprises should assess the privacy risks of techniques when they are applied to personal data. Privacy breaches in models or training data can result in considerable financial losses and legal risks for businesses. Therefore, in this dissertation, we delve into privacy risks centering around models and their training data, classifying these risks into two types: model privacy and data privacy.

Model privacy in generative models. We illustrate our motivation with regard to model privacy in generative models from two aspects.

The first motivation is that adversaries might bear significantly low costs to obtain an ML model that enjoys a comparable level of performance to the victim model (also referred to as the target model in this work). In practice, attaining a cutting-edge performance level for an ML model is not straightforward. It generally necessitates engaging in the complicated and exhausting process of data collection and purification, possessing expert-level knowledge in model architecture and algorithm design, performing elaborate hyperparameter tuning, and having access to extensive computing resources. Therefore, a high-quality well-trained ML model is a comprehensive undertaking that entails substantial financial investment and the commitment of significant human resources, and it is considered the intellectual property of model owners. Moreover, the escalating number of enterprises has deployed Machine Learning as a Service (MLaaS) where customers are allowed to use their state-of-the-art ML models by subscription payments, such as Amazon AWS, Google Cloud, and Microsoft Azure. This emerging attack surface further incentivizes adversaries to compromise the model privacy and intellectual property of model owners, such as launching model extraction attacks to illicitly obtain high-value ML models.

The second motivation is that model extraction attacks and defenses in generative models are less explored. Early work on attacking model privacy focuses on stealing simple classification models [TZJ+16], such as logistic regression, decision tree, support vector machine and multilayer perceptrons, and complex classification models on deep convolution neural networks [OSF19, JCB+20] and classification models on natural language processing tasks [KTP+20]. There are several works that steal hyperparameters or parameters of a classification model [WG18, CJM20]. However, generative models, as an equally important type of ML model, do not receive much attention from the privacy research community. In a nutshell, both motivations drive us to study the feasibility of model extraction attacks in generative models and strive to propose potential defense measures to safeguard the ownership of generative models.

Data privacy in generative models. We illustrate our motivation with regard to data privacy in generative models from two aspects.

The first motivation is that data privacy attacks might cause severe privacy risks, i.e. considerable leakage of sensitive information in the training set via attacking an ML model. This threat is not allowed by existing data protection regulations when the training set involves personal data, such as human faces or medical records. For example, GDPR [PotEU16], a regulation in European Union law on data protection, requires that it is mandatory to conduct thorough assessments of potential privacy threats associated with artificial intelligence technologies that involve sensitive data. Therefore, successful data privacy attacks can cause enterprises to incur substantial financial loss and potential legal sanctions. In other words, it is paramount for an ML model to systematically study data privacy attacks under various adversarial scenarios. Beyond that, these attacks, in a synergistic manner, can serve as powerful

catalysts, fostering significant advancements in protection methodologies.

The second motivation is that the latest generative models necessitate new data privacy assessments and protection methods. Prior methods on membership inference attacks mainly focus on classification models [SSSS17], early generative models, such as variational autoencoders (VAEs) [HHB19] and GANs [HMDDC19], which are not applicable to the latest generative models — diffusion models. On the one hand, it is due to the difference between classification models and generative models. Take the image domain as an example, classification models output the prediction results of images, such as class labels or confidence scores of being certain class, while generative models directly generate images that are high-dimensional data. On the other hand, this is because the latest diffusion models [HJA20, SSDK+21] have different generative mechanisms from VAEs [KW14] and GANs [GPAM+14]. For instance, unlike GANs which have one loss value in the training phase, diffusion models have lots of loss values because diffusion models introduce a time dimension and each diffusion time step corresponds to a loss value. This provides more potential signals to adversaries and it requires a new study to unearth new insights about privacy risks in diffusion models. In addition, different sampling mechanisms between diffusion models and early generative models, such as GANs, require a new defense method to protect the privacy of diffusion models. To sum up, both motivations drive us to investigate data privacy attacks and propose protection methods on the latest diffusion models.

Applications of generative models. In addition to studying privacy risks related to generative models themselves, we further unlock their potential and leverage generative models as a tool to investigate evasion attacks in discriminative models. Our motivation is that generated samples from generative models might be utilized by adversaries to fool classification models, even if classification models are protected by an out-of-distribution (OOD) detector. On the one side, there are many well-trained generative models which can generate images with both a greater stylistic diversity and a substantially larger quantity of images than those from datasets collected by humans. Thus, generated data from generative models can be utilized to maliciously attack classification models. On the other side, there are many research works on studying the OOD generalization of classification models. However, these works evaluate the OOD generalization of classification models via other OOD datasets, which might largely overestimate the OOD generalization. This is because OOD datasets used in the current literature are often limited to several common datasets [YZLL21, CLW+21, FLL+22]. For instance, the OOD performance of a classification model trained on CIFAR-10 is evaluated by the SVHN dataset [NWC+11] and LSUN dataset [YSZ+15]. Therefore, leveraging various pre-trained and publicly available generative models, we aim to develop attack techniques to fool a classification model and evade OOD detection. This new angle of attack aims to provide novel insights into the generalization of classification models.

1.4 Dissertation Contributions

This dissertation makes five technical contributions over three thematic parts. These range from model privacy and data privacy to evasion attacks, addressing the motivation illustrated previously.

1.4.1 Model Privacy

We analyze model privacy via the lens of model extraction attacks which aim to steal the function of victim models. By investigating model extraction attacks against GANs, we show that adversaries can steal a model that has quite similar performance (in terms of *fidelity*) as the victim model with only about 50K queries/generated samples. By investigating ownership protection methods for GANs, we provide evidence that existing ownership protection methods fail to defend against our model extraction attacks and our defense can provide effective ownership verification on both physical stealing and model extraction scenarios.

• **Model extraction in GANs.** First, we taxonomize the space of model extraction attacks on GANs and define the fidelity extraction and accuracy extraction, respectively [HP21b]. The corresponding model extraction methods are proposed and achieve excellent performance, which is also the first systematic study on model extraction attacks in GANs. We perform one case study to illustrate the impact of model extraction attacks against GANs on a large-scale scenario. We propose new effective defense measures to mitigate model extraction attacks against GANs by a trade-off between attack performance and model utility.

• **Ownership protection in GANs.** We show that prior works about watermark-based and fingerprint-based methods cannot provide ownership protection under model extraction attacks [HP23b]. We propose a protection method GAN-Guards from a new angle: detecting ownership infringement by utilizing the common characteristics of a target model and stolen models. Our method achieves a new state-of-the-art performance on both physical stealing attacks and emergent model extraction attacks.

1.4.2 Data Privacy

We analyze data privacy from two different perspectives - one that seeks to infer the individual information of a training set, and one that seeks to infer the global information of a training set. By evaluating membership inference attacks against the latest diffusion models, we reveal the relationship between membership inference risks and the generative mechanism of diffusion models: different diffusion steps of a diffusion model have significantly different privacy risks, and there exist high-risk regions which lead to leakage of training samples from the perspective of diffusion steps. By evaluating the performance of the property inference defense, we demonstrate that property inference risks of diffusion models can be effectively

mitigated by a property aware sampling method that gracefully boasts plug-in and model-agnostic advantages.

• **Membership inference of diffusion models.** We propose two types of attacks to infer membership of diffusion models [HP23a]. We show that from the dimension of diffusion steps, there are high-risk regions that can cause the leakage of training samples, and the likelihood values of samples from a diffusion model are a strong clue to infer training samples. We evaluate our attacks on one classical defense - diffusion models trained with differentially-private stochastic gradient descent (DP-SGD), finding that it mitigates our attacks at a substantial compromise to the quality of synthetic samples. Our code in this work is available at `https://github.com/HailongHuPri/MIDM`.

• **Property inference of diffusion models.** We perform the first study of property inference attacks against diffusion models under the most practical attack scenario, showing that diffusion models and their samplers are vulnerable to property inference attacks [HP23c]. We conduct one case study to demonstrate the privacy risks of property inference of diffusion models in practice. We propose the first model-agnostic and plug-in defense — PriSampler to mitigate property inference attacks against diffusion models, illustrating that our method achieves state-of-the-art performance in both model utility and defense performance.

1.4.3 Applications

We analyze evasion attack risks of classification models by the utilization of GAN models. By investigating OOD attacks against classification models, we demonstrate that well-trained GANs can be utilized to directly construct samples to fool classification models and evade OOD detection.

• **OOD attacks via GANs.** We formulate the problem as: leveraging any pre-trained GANs, an adversary aims to fool a classification model and make the model misclassify a sample from GANs as a pre-specified target class [HP22]. We propose a novel and unified OOD targeted attack framework for white-box and black-box scenarios, which is the first work to study out-of-distribution attacks through GANs. Our framework casts this problem as an optimization problem and a family of attack methods are developed. We thoroughly evaluate our attack methods on various victims trained on various datasets and expand our method to attack classification models with classical and state-of-the-art defense measures.

1.5 Dissertation Structure

This dissertation is comprised of three thematic parts.

Part I investigates the privacy attacks and protection of generative adversarial networks from the perspective of model privacy.

- In Chapter 3, we systematically study the feasibility of model extraction attacks against GANs. We propose two types of model extraction attacks, conduct one case study in practice and present several mitigating techniques. This chapter was previously published as [HP21b].
- In Chapter 4, we present one defense method against model extraction attacks by verifying the ownership of GANs, demonstrating its effectiveness by comparison and a number of analysis experiments, such as performance with respect to the number of generations of model extraction attacks, the number of generated samples, different datasets, different target models and adaptive attacks. This chapter is based on [HP23b].

Part II focuses on the privacy attacks and protection of diffusion models from the perspective of data privacy.

- In Chapter 5, we investigate membership inference risks of diffusion models. We propose two attack methods, namely loss-based and likelihood-based attacks, illustrating their effectiveness by attacking several state-of-the-art diffusion models trained on privacy-sensitive datasets and diffusion models trained with differential privacy. This chapter is built upon [HP23a].
- In Chapter 6, we study property inference risks of diffusion models. We conduct property inference attacks against different types of samplers and diffusion models under the most practical attack scenario, and present one case study about attacking off-the-shelf pre-trained diffusion models, and propose a new model-agnostic plug-in method PriSampler to mitigate the property inference of diffusion models. This chapter is based on [HP23c].

Part III is about the application of generative models in analyzing the robustness of classification models.

- In Chapter 7, we propose a novel OOD framework to fool a classification model by leveraging any pre-trained GANs. We develop a family of attack methods under this framework and show that our methods can achieve competitive performance by comparing with several state-of-the-art adversarial example attacks and evading several widely-used and the latest defenses. This chapter is based on [HP22].

We conclude and discuss future avenues in Chapter 8.

Chapter 2

Preliminaries

In this chapter, we start with the introduction of generative models, and illustrate two predominant families of generative models in computer vision: generative adversarial networks and diffusion models. Then, we describe the background knowledge about threat models from the viewpoint of adversaries.

2.1 Generative Models

Problem formulation. Given a training set with a size of N, i.e. $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$, one of the most commonly used assumptions in the machine learning research community is that these data samples in the dataset $\mathcal{D}$ are independent and identically distributed and are from an underlying (unknown) data distribution $p_{data}(x)$ [GBC16]. The goal of generative models is to model and estimate the unknown data distribution $p_{data}(x)$ based on a given training set $\mathcal{D}$. To achieve this, we first denote a generative model $p_\theta(x)$, where $\theta \in \Theta$ is model parameters and Θ is the set of parameter values. Then, we aim to find the optimal parameters $\theta^* \in \Theta$ such that

$$p_{\theta^*}(x) \approx p_{data}(x). \tag{2.1}$$

Once the optimal generative model $p_{\theta^*}(x)$ is obtained, we can synthesize numerous new data via sampling from $p_{\theta^*}(x)$. Furthermore, synthesized samples and generative models themselves can be also applied to a variety of applications, such as data augmentation to cope with the scarcity of data [KAH+22], image editing via the well-trained generative models [XZY+22]. A generative model is also known as a statistical model or a probabilistic generative model. In the machine learning community, a generative model is also called a deep generative model, because current generative models generally utilize deep neural networks to learn high-dimensional probability distribution and achieve state-of-the-art generation performance.

Training of generative models. To get the optimal parameter θ^* of a generative model, we can minimize the distance between $p_{data}(x)$ and $p_\theta(x)$ by the following formula:

$$\theta^* = \arg\min_\theta D(p_{data}||p_\theta), \tag{2.2}$$

where $D(\cdot||\cdot)$ is a distance metric. Common distance metrics include f-divergences [Csi64, AS66], Kullback-Leibler (KL) divergence [KL51] and integral probability

metrics [Mül97]. Take the KL divergence as an example, its standard formula for $p_{data}(x)$ and $p_\theta(x)$ is:

$$\begin{aligned} D_{KL}(p_{data}||p_\theta) &= \int p_{data}(x) \log \frac{p_{data}(x)}{p_\theta(x)} \mathrm{d}x \\ &= \mathbb{E}_{p_{data}(x)} \left[\log(\frac{p_{data}(x)}{p_\theta(x)}) \right]. \end{aligned} \tag{2.3}$$

Furthermore, given a training set $\mathcal{D}$ containing N samples, we can approximate Equation 2.3 by the empirical mean:

$$\begin{aligned} D_{KL}(p_{data}||p_\theta) &\approx \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{p_{data}(x_i)}{p_\theta(x_i)} \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} \log p_{data}(x_i) - \frac{1}{N} \sum_{i=1}^{N} p_\theta(x_i). \end{aligned} \tag{2.4}$$

In the course of model training, the first item of Equation 2.4, i.e. $\frac{1}{N} \sum_{i=1}^{N} \log p_{data}(x_i)$, is a constant because it is independent of the model parameter θ. In other words, we can drop this item in the training process. As a result, combining Equation 2.2 and Equation 2.4, a generative model can be trained by:

$$\begin{aligned} \theta^* &= \arg\min_\theta \left(-\frac{1}{N} \sum_{i=1}^{N} p_\theta(x_i) \right) \\ &= \arg\max_\theta \left(\frac{1}{N} \sum_{i=1}^{N} p_\theta(x_i) \right). \end{aligned} \tag{2.5}$$

Equation 2.5 is a standard method to learn a probabilistic generative model, which is also the *maximum likelihood estimation* approach in the statistics community.

Challenges of generative modeling. When optimizing Equation 2.5 and ensuring the validity of probability distributions, there are two natural constraints for a generative model $p_\theta(x)$ [Son22]:

(1) Non-negativity: $\forall x : p_\theta(x) \geq 0$.

(2) Normalization: $\int p_\theta(x) \mathrm{d}x = 1$.

Non-negativity guarantees that each probability is non-negative, while normalization says that the sum/integration of the values of the probability distribution is equal to 1. Generally, normalization is much more challenging than non-negativity. This is because normalization involves intractable computing problems for high-dimensional data x, while non-negativity can be conveniently implemented by exponential functions, such as $\exp(p(x))$. Therefore, this also inspires numerous talented researchers to propose various methods for generative modeling.

Various families of generative models. In general, there are six families of generative models: energy-based models [AHS85, LCH+06], autoregressive models [BB99, GGML15], normalizing flows [RM15, DSDB16], variational autoencoders [KW14],

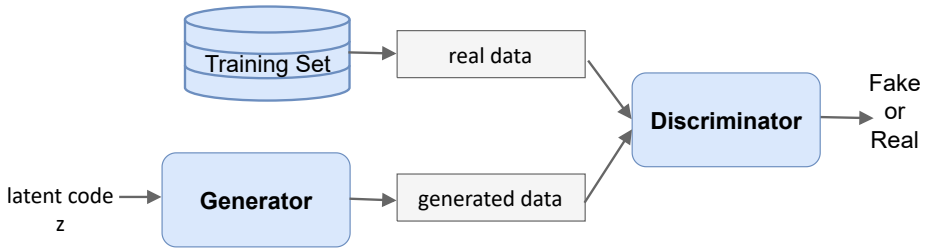


Figure 2.1: The mechanism of a GAN.

generative adversarial networks [GPAM+14] and diffusion models [SDWMG15]. In computer vision, early generative models cannot generate high-quality and high-resolution images, due to either the restriction of model architectures, such as autoregressive models, normalizing flows, and variational autoencoders, or the heavy computation for high-dimensional data, such as energy-based models. As a consequence, generative adversarial networks and diffusion models stand out in the image generation domain due to their unique characteristics. In the following section, we introduce two types of predominant generative models in image generation.

2.1.1 Generative Adversarial Networks

Generative adversarial networks (GANs) completely avoid the normalization challenge by only focusing on modeling the data generation process. Therefore, GANs are also the first generative models which can generate high-quality, high-resolution, and realistic images.

The basic mechanism of a GAN

GANs are a class of generative models where they adversarially learn the unknown distribution p_{data} on the training data set $\mathcal{D}$. As shown in Figure 2.1, a GAN generally consists of two components: a generator G and a discriminator D, both of which are structured as deep neural networks. G is responsible for generating fake data $x_g = G(z)$, where the latent code z is sampled from a prior distribution p_z, such as Gaussian distribution or uniform distribution. In contrast, D takes the role of a binary classifier which differentiates real-like samples x_g from real samples $x_r \in \mathcal{D}$ as accurately as possible. In the course of training, G and D compete with each other, resulting in a min-max (two-player adversarial) game. In the course of the deployment, only G is utilized to produce new synthetic data while D is usually discarded.

The seminal GAN [GPAM+14] utilizes multilayer perceptrons as the architectures of the generator G and the discriminator D. It is trained through optimizing the following loss function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z}\left[1 - \log D(G(z))\right]. \tag{2.6}$$

A GAN is an implicit generative model, and the learned distribution of the GAN, i.e. p_g, can be implicitly defined from p_z and $G(z)$. If the generator G and the discriminator D converge and reach global equilibrium, then $p_{data}(x) = p_g(x)$. For

a fixed G, the optimal discriminator D^* can be obtained by:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2.7}$$

Once D^* can be exactly calculated, Goodfellow et al. [GPAM+14] further show that the loss function, i.e., Equation 2.6, can be reformulated as:

$$\begin{aligned} \max_D V(D,G) &= D_{KL}(p_{data}||\frac{1}{2}(p_{data}+p_g)) + D_{KL}(p_g||\frac{1}{2}(p_{data}+p_g)) - \log 4 \\ &= 2D_{JS}(p_{data}||p_g) - \log 4. \end{aligned} \tag{2.8}$$

$D_{JS}(\cdot||\cdot)$ represents the Jensen-Shannon (JS) divergence, which is a specific instance of f-divergence. Therefore, recalling Equation 2.2, the objective of the seminal GAN in essence is also to minimize the distributional distance between p_{data} and p_g, where the distance is defined by the JS divergence.

Various GANs

To further improve the performance of GANs, numerous methods have been proposed from the perspective of loss functions or architectures of GANs. In the following, we introduce several representative GAN models.

• **WGAN.** The seminal GAN has instability problems during the training process. One of the reasons is that the JS divergence fails to accurately reflect the distance when p_{data} and p_g do not overlap. However, in practice, there is a high probability for p_{data} and p_g to overlap [AB17]. To address these problems, the Wasserstein GAN (WGAN) [ACB17] proposes to utilize the Wasserstein distance to measure the distance between distributions. Intuitively, the Wasserstein distance is the cost of the optimal transport that is required to transform p_{data} into p_g. Thus, WGAN contributes to a more stable training process and enhances the GAN's performance.

• **WGAN-GP.** Gulrajani et al. find that the use of weight clipping in WGAN to enforce a Lipschitz constraint on the discriminator can lead to undesired behaviour [GAA+17]. Therefore, they propose WGAN with gradient penalty (WGAN-GP) penalize the norm of the gradient of the discriminator with respect to its input. As a result, the performance of a GAN can be further improved.

• **SNGAN.** Spectral normalization GAN (SNGAN) [MKKY18] proposes to stabilize GAN training by a weight normalization technique. It is used on the discriminator to adjust the weights of all the layers to 1. SNGAN ensures that the discriminator is KLipchitz continuous. This enables SNGAN to present better training behavior and obtain a better quality of generated samples.

• **DCGAN.** Unlike the seminal GAN that uses multilayer perceptrons as the architectures of a GAN [GPAM+14], the deep convolutional GAN (DCGAN) [RMC16] proposes to introduce a deconvolutional neural network into the generator of a GAN. The spatial up-sampling ability of the deconvolution operation enables a GAN to generate higher resolution images. Consequently, the deconvolution operation becomes a main operation in subsequent GAN architectures.

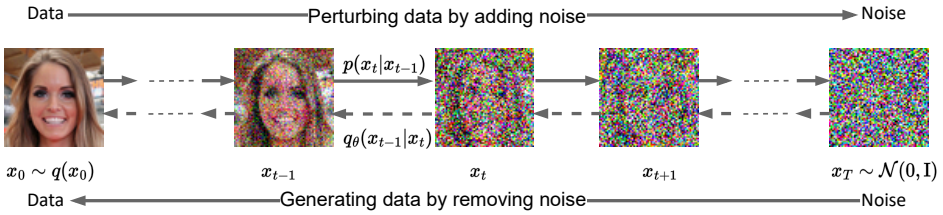


Figure 2.2: The mechanism of a diffusion model.

• **PGGAN.** In the pursuit of enhancing the performance of a GAN from the architectural perspective, the Progressive Growing GAN (PGGAN) [KALL18] proposes a method that incorporates progressive growing neural networks in the GAN framework. This enables PGGAN to generate as high as 1024×1024 resolution of realistic images. To be specific, PGGAN is trained from 4×4 pixel images and gradually increases the size of training images and neural networks. As a consequence, the large size of generated images can be obtained.

• **StyleGAN.** On the basis of PGGAN, StyleGAN [KLA19] further improves the performance of a GAN by introducing a style controlled mechanism into the generator of a GAN. In the style controlled mechanism, an 8-layer multilayer perceptron is first employed to transform input latent codes z into intermediate latent codes w. Subsequently, these intermediate latent codes are transformed by learned affine transformations, and the resulting codes, i.e. styles, are added to each convolution layer of the generator to control the styles of generated images. To enhance stochastic variation, Gaussian noise is also added after each convolution layer. As a result, the style-based generator can synthesize diverse styles and high-resolution images.

2.1.2 Diffusion Models

Unlike GAN models which directly model the data generation process but have challenges in training stability, diffusion models learn a data distribution by gradually adding the noise and denoising.

The basic mechanism of a diffusion model

A diffusion model is a generative model, and it aims to learn the distribution p_{data} of a training set and generate new unseen data samples. In general, as shown in Figure 2.2, a diffusion model consists of two processes: a forward process and a reverse process. In the forward process, it adds different levels of noise $0 = \sigma_0 < \sigma_1 <, ..., < \sigma_T = \sigma_{max}$ into training data, in order to transform a training data's distribution into a Gaussian distribution within T time steps. In the reverse process, it randomly samples a noise image from the Gaussian distribution and gradually denoises it into an image.

Various diffusion models

In the following, we introduce three fundamental types of diffusion models.

• **DDPM.** The denoising diffusion probabilistic model (DDPM) is proposed by Ho et al. [HJA20]. In the forward process, a sample at the t time step is perturbed by: $x_t \leftarrow \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \mathrm{I})$, $x_0 \sim p_{data}$, and $\alpha_t \in [0, 1]$ is a variance

schedule to control the magnitude of noise in each time step. $\alpha_0 = 1$ means that an image at $t = 0$ time step is not perturbed and $\alpha_T = 0$ indicates that the perturbed image at $t = T$ time step becomes pure Gaussian noise. In the reverse process, a noise image from $\mathcal{N}(0, \mathrm{I})$ is step by step denoised and eventually recovers a noise-free image, and during the process a neural network $\epsilon_\theta(x_t, t)$ is trained to predict noise by minimizing the following loss:

$$L(\theta) = \mathbb{E}_{t\sim[1,T], x\sim p_{data}, \varepsilon\sim\mathcal{N}(0,\mathrm{I})}[||\varepsilon - \varepsilon_\theta(\sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\varepsilon, t)||^2]. \tag{2.9}$$

• **SMLD.** The score matching with Langevin dynamics (SMLD) is proposed by Song et al. [SE19]. In the forward process, a perturbed sample at the t time step is obtained by: $x_t \leftarrow x_0 + \sigma_t\epsilon$, where σ_t is the noise schedule to control the magnitude of noise. In the reverse process, a neural network $s_\theta(x_t, \sigma_i)$ is trained to predict *score*. The *score* refers to the gradient of the log probability density with respect to data, i.e. $\nabla_x log\ p(x)$. SMLD minimizes the following loss:

$$L_\theta = \mathbb{E}_{t\sim[1,T], x\sim p_{data}, x_t\sim q(x_t|x)}[\lambda(\sigma_t)||s_\theta(x_t, \sigma_t) - \nabla_{x_t}\log q(x_t|x)||^2], \tag{2.10}$$

where $\lambda(\sigma_t)$ is a coefficient function and $\nabla_{x_t}\log q(x_t|x) = -\frac{x_t - x}{\sigma_t^2}$.

• **SSDE.** The score-based stochastic differential equation (SSDE) proposed by Song et al. [SSDK+21] presents a general and unified framework for generative modeling. The process of a diffusion model is described as a stochastic differential equation. Specifically, SSDE defines the forward process as:

$$\mathrm{d}x = \mathrm{f}(x, t)\mathrm{d}t + g(t)\mathrm{dw}, \tag{2.11}$$

where $\mathrm{f}(x, t)$, $g(t)$ and dw are the drift coefficient, the diffusion coefficient and a standard Wiener process, respectively. In the reverse process, it can be expressed by a reverse-time SDE: $\mathrm{d}x = [\mathrm{f}(x, t) - g(t)^2\nabla_x \log q_t(x)]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathrm{w}}$, where $\bar{\mathrm{w}}$ is a standard Wiener process in the reverse time. A neural network is used for predicting *score* by minimizing the loss:

$$L_\theta = \mathbb{E}_{t\in\mathcal{U}(0,T), x\sim p_{data}, x_t\sim q(x_t|x)}[\lambda(t)||s_\theta(x_t, t) - \nabla_{x_t}\log\ q(x_t|x)||^2]. \tag{2.12}$$

Different coefficients, i.e. $\mathrm{f}(x, t)$ and $g(t)$, correspond to different types of SSDE, and in their work, two types of SSDE are proposed: variance preserving (VP) and variance exploding (VE). We call their corresponding models as VPSDE and VESDE and they are continuous diffusion models. Furthermore, under this framework, DDPM and SMLD can be considered discrete VP and VE, respectively.

2.1.3 Sampling

Sampling mechanisms of GANs. The sampling procedure of a GAN is highly efficient, requiring only a single forward pass through the trained GAN. This is because it directly models the data generation process. To be specific, given a

trained GAN G, generated samples x_g can be obtained by latent codes, such as $x_g = G(z), z \sim \mathcal{N}(0, \mathrm{I})$.

Sampling mechanisms of diffusion models. Unlike GANs that can directly generate samples, diffusion models usually need iterative approaches that involve a large number of forward passes. Therefore, there are many sampling methods for diffusion models that aim to speed up the sampling process and maintain or improve the quality of generated samples. In addition, based on the unified framework of SSDE [SSDK+21], sampling methods can be classified into two categories: stochastic sampling and deterministic sampling.

• **Stochastic sampling.** Because a diffusion model can be described as a stochastic differential equation (SDE), we can generate a new sample by solving the corresponding reverse-time SDE [And82]. Existing general-purpose numerical solvers, such as Euler-Maruyama and stochastic Runge-Kutta methods [PBL10], can be used for solving the SDE. Song et al. [SSDK+21] propose Predictor-Corrector methods to further improve the sampling quality by utilizing the score-based model. In this work, we call it a PC sampler.

• **Deterministic sampling.** In addition to solving a reverse-time SDE, Song et al. [SSDK+21] find that a reverse-time SDE also corresponds to a probability flow ordinary differential equation (ODE) in which they have the same marginal probability densities. It indicates that we can generate a new sample by solving a probability flow ODE. Existing black-box ODE solver [DP80] can be used to generate samples. In this work, we call it an ODE sampler.

In addition to directly using a black-box ODE solver, there are many works about designing efficient samplers based on solving the probability flow ODE [LRLZ22, LZB+22, ZC23]. For example, DPM [LZB+22] analyzes the ODE consisting of a linear function of the data variable and a nonlinear function parametrized by neural networks. By deriving an exact formulation for the linear part, DPM can improve the quality of generated samples and speed up the sampling process. In this work, we call it a DPM sampler.

2.2 Threat Model

In this section, we describe a general threat model from two perspectives: adversarial capabilities and adversarial goals. An adversary can develop different levels of attack methods based on a threat model.

2.2.1 Adversarial Capabilities

Adversarial capabilities refer to how much information adversaries can exploit when launching an attack. Figure 2.3 shows components of a generative model. It consists of four parts: training set, generative model, latent codes and generated data. In the training phase, a generative model leans the distribution of a training

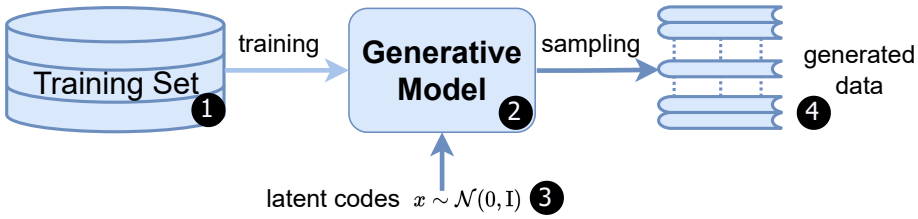


Figure 2.3: Components of a generative model.

set. In the sampling phase, the trained generative model is utilized to synthesize new data by latent codes. We detail each component as follows.

❶ **Training set.** Adversaries may obtain the information on a training set, including the distribution information and/or partial training set. The assumption on the distribution information usually refers that adversaries have a shadow dataset that has the same distribution as the training set, which is widely used in shadow-based membership inference attacks [SSSS17, CHN+23]. The assumption of obtaining a partial training set is often utilized to further enhance the attack performance on membership inference [HMDDC19] or model extraction [JCB+20, HP21b].

❷ **Generative model.** Adversaries might attain the information on a generative model itself, such as the architecture of a generative model, and the partial and/or whole well-trained generative model. Obtaining a whole generative model might happen in a white-box inference attack where adversaries can fully have access to the model's parameters. Obtaining the part of a generative model commonly happens when adversaries mount an attack relying on the discriminator of a GAN, rather than the whole GAN [HMDDC19, HP21a].

❸ **Latent codes.** Latent codes refer to random numbers drawn from a prior distribution such as Gaussian distribution. Adversaries obtaining latent codes mean that they know the prior distribution or can leverage latent codes to interactively query a model's application programming interface (API) provided by model owners. Due to the rising popularity of MLaaS, querying a model is becoming more prevalent. Thus, this is a basic assumption in the query-based attack setting.

❹ **Generated data.** Generated data refers to samples from a generative model. In some scenarios, adversaries are unable to query the target model but only have access to the generated samples which are publicly released by the model provider. Whether adversaries can achieve a good attack performance depends on the amount of released data from model owners. This is also the strictest and most practical assumption.

Depending on the weak or strong assumptions, an adversary may obtain one or several components to conduct an attack.

2.2.2 Adversarial Goals

Adversaries can have distinct goals when attacking a machine learning (ML) model. Generally, we can classify the goals of adversaries into two categories: *integrity* and

privacy [PMSW18, Jag21]. For brevity, we also refer to attacks with the goal of integrity and privacy as integrity attacks and privacy attacks, respectively.

Integrity attacks. Adversaries who aim to break the integrity of an ML model attempt to manipulate an ML model to undermine the model's performance. Integrity attacks can happen in the training phase or in the inference phase (also called the sampling phase in generative models). The attack performance that adversaries want to compromise includes many types, such as decreasing the accuracy of a classification model [BCM+13, SZS+14], increasing the privacy risks of a classification model [TSSJ+22], and exacerbating the unfairness risks of a certain class in a classification model [JSPHO21]. Typical integrity attacks include poisoning attacks and evasion attacks.

• **Poisoning attacks.** Poisoning attacks are a class of training phase attacks where adversaries compromise the model performance via injecting carefully constructed examples into the training set [NBC+08, CLL+17]. In untargeted poisoning attacks, adversaries poison an ML model to reduce its accuracy to chance while target poisoning attacks cause the specific misprediction of a particular sample.

• **Evasion attacks.** Evasion attacks manipulate the inputs to fool an ML model in the inference phase [BCM+13, SZS+14]. They are also called adversarial example attacks. Given a well-trained ML model, adversaries aim to construct perturbed inputs, i.e. adversarial examples, to compromise the accuracy of this ML model. Similar to poisoning attacks, untargeted evasion attacks decrease the accuracy of an ML model randomly while targeted evasion attacks make an ML model misclassify one sample into a certain specific class.

Privacy attacks. Adversaries who target to violate the privacy of an ML model attempt to reveal the information of an ML model itself or the information of its training set. Typical attacks with the goal of privacy include model extraction attacks, membership inference attacks, and property inference attacks.

• **Model extraction attacks.** Model extraction attacks aim to extract or steal information of an ML model itself, such as reconstructing a functionally-equivalent substitute model [TZJ+16]. Jagielski et al. [JCB+20] further classify model extraction of classification models into accuracy extraction and fidelity extraction. Accuracy extraction refers that adversaries steal an ML model which exhibits similar performance to a victim model on the same test set, while fidelity extraction requires that the substitute model cannot only have similar accuracy in correctly-recognized samples and replicate similar errors on a test set. This type of attack, on the one hand, causes the infringement of the intellectual property of model owners. On the other hand, it can serve as a stepping stone for other types of attacks, such as evasion attacks.

• **Membership inference attacks.** Membership inference attacks, one of the most popular privacy attacks, aim to determine whether one sample was used for training an ML model [SSSS17]. They usually infer membership on the assumption that adversaries can have shadow sets and obtain model's predictions, such as labels, loss values, and logit values. Obtaining model parameters and gradients is a strong assumption to conduct membership inference. Membership inference attacks focus

on inferring individual samples of the training set of a trained ML. In extreme cases, there are some attacks aiming to extract or reconstruct training samples, which are also called data extraction attacks [CTW+21]. Nevertheless, data extraction attacks usually require high attack costs.

• **Property inference attacks.** Property inference attacks aim to reveal the macro information of the training set of an ML model [AMS+15]. For example, adversaries could extract the ratio of gender in a classification model used for loan scoring, which is not shared by model owners. Moreover, the attacks allow adversaries to obtain much richer insight about the training set and can be utilized to infer vulnerabilities of a security-critical system [GWY+18].

In the following chapters, we will present a family of privacy attacks against generative models, and the corresponding protection mechanisms are exhaustively discussed and proposed. In addition, a novel evasion attack via generative models will be illustrated to reveal the security vulnerability of discriminative models.

Part I

Model Privacy in Generative Models

Chapter 3

Model Extraction in Generative Adversarial Networks

Once generative adversarial networks (GANs) have been trained and achieved state-of-the-art performance, how significant is the threat of model extraction attacks against GANs? In this chapter, we systematically study the feasibility of model extraction attacks against GANs. Specifically, we first define fidelity and accuracy on model extraction attacks against GANs. Then we study model extraction attacks against GANs from the perspective of fidelity extraction and accuracy extraction, according to the adversary's goals and background knowledge. We further conduct a case study where the adversary can transfer knowledge of the extracted model which steals a state-of-the-art GAN trained with more than 3 million images to new domains to broaden the scope of applications of model extraction attacks. Finally, we propose effective defense techniques to mitigate the risks, considering a trade-off between the utility and model privacy of GANs.

3.1 Introduction

Over the past few years, machine learning, deep learning in particular, has gained significant advances in a variety of areas, such as computer vision [TVDJ19, BDS19, KLA19, KLA+20] and natural language processing (NLP) [DCLT19, LYK+20]. In general, machine learning models are often considered as the intellectual property of model owners and are closely safeguarded. The reasons are from at least two aspects. First, obtaining a practical deep learning model is non-trivial. This is because training a model requires a large number of training data, intensive computing resources, and human resources [RDS+15, YSZ+15, TVDJ19, DCLT19, BMR+20, KLA+20]. Second, deep learning models themselves are confidential, and exposure of deep learning models to potential adversaries poses a threat to security and privacy [LM05, PMG+17, SSSS17, SZH+19, TZJ+16, LTR+19]. However, a model extraction attack — a novel attack surface targeting at duplicating a model only through query access to a target model, has recently emerged and gained significant attention from the research community.

In the early study, Tramèr et al. [TZJ+16] first attempt model extraction on traditional machine learning models and shallow neural networks, such as logistic

regression, decision tree, support vector machine and multilayer perceptrons. Since then, Jagielski et al. [JCB+20] further mount the attack against a million of parameters model trained on billions of Instagram images [MGR+18], which makes model extraction attack more practical. In addition to model extraction on deep convolutional neural networks about image classification, there are some works studying the problem of model extraction in NLP tasks [KTP+20, TYF20]. For instance, with the assumption that victim models are trained based on the pre-trained BERT model, Krishna et al. [KTP+20] show that an adversary can effectively extract language models whose performance is only slightly worse than that of the victim models. However, to the best of our knowledge, these model extraction attacks mainly focus on discriminative models. The attack against generative models, GANs in particular, is still an open question.

Comparing to model extraction attacks on discriminative models, we observe that there exist some differences for generative models. First, adversaries can leverage output information from target models such as labels, probabilities and logits, to mount model extraction attacks on discriminative models [LM05, OSF19, JCB+20, TZJ+16], while generative models do not provide such information but only return images. Second, model extraction attacks on discriminative models are evaluated on a test dataset. In contrast, unsupervised generative models aiming to learn the distribution of training data are evaluated by quantitative measures such as Fréchet Inception Distance (FID) [HRU+17] and multi-scale structural similarity (MS-SSIM) [OOS17], or qualitative measures such as preference judgment [HLP+17, ZXL+17]. Therefore, these differences indicate that model extraction strategies, evaluations and defenses on generative models are very different from these on discriminative models.

In this chapter, we aim to systematically study the feasibility of model extraction attacks against GANs from the perspective of fidelity extraction and accuracy extraction. First, we define *fidelity* and *accuracy* of model extraction on GANs. More specifically, when an adversary mounts model extraction attacks against GANs, *fidelity* measures the difference of data distribution between the attack model and the target model, while *accuracy* ensures the distribution of the attack model is consistent with the distribution of the training set of the target model. In the next step, according to the adversary's goals and the background information that they can have access to (see Figure 3.2), we systematically study two different types of attacks on GANs: fidelity extraction attack and accuracy extraction attack, which are shown in Figure 3.1.

Fidelity extraction attack. Adversaries mounting fidelity extraction focus on *fidelity* and they aim to steal the distribution of a target model. Here, the distribution of a target generative model also refers to the function of this generative model, because a generative model is used for learning the distribution of a training set. For this attack, we assume adversaries have no knowledge of the architecture of target models, and they either obtain a batch of generated data that the model owner has publicly released or query the target model to obtain generated data. It can be considered as a black-box fidelity extraction. After obtaining the generated data, adversaries can train a copy of the target GAN model. We study two different

target models: Progressive GAN (PGGAN) [KALL18] and Spectral Normalization GAN (SNGAN) [MKKY18]. Extensive experimental evaluations show that fidelity extraction can achieve an excellent performance with only about 50K queries (i.e., 50K generated samples). When we continue to increase the number of queries, we find that it cannot bring significant improvement of the *accuracy* of attack models. This is mainly because the discriminator of a target GAN model is often better than its corresponding generator and it is very hard to reach global optimum [AOD+19]. In other words, directly querying the target model enables the attack model to be more consistent with the target generator rather than the real data distribution of the target model (see Figure 3.5 for an example). Therefore, it motivates us to perform accuracy extraction to improve the *accuracy* of attack models.

Accuracy extraction attack. Adversaries mounting accuracy extraction concentrate on *accuracy* and they target at stealing the distribution of the training set of a target model. In order to achieve a high accuracy model extraction attack, we propose to utilize subsampling techniques where generated samples far away from the true distribution are rejected and only samples that are closer to the true distribution are retained (see Figure 3.5). To achieve this goal, we assume that adversaries can obtain more background knowledge. In particular, we assume adversaries can obtain the discriminator from the target GAN model and partial real data. We utilize the discriminator to subsample generated samples. These refined samples are more close to real data distribution, compared to samples that are directly generated by the target model (see Figure 3.5(e)). Then, we use these refined samples and partial real data to train our attack model. Extensive experimental evaluations show that our accuracy extraction attack indeed brings improvement of the *accuracy* of attack models, compared to fidelity extraction attacks (see Figure 3.6). This indicates that the risks of partially releasing training data can be further exacerbated under this type of attack.

Case study. We perform one case study to further demonstrate the impact of model extraction attacks on a large-scale scenario. In this case study — model extraction based transfer learning (Section 3.7), we show that stealing a state-of-the-art GAN model can enable adversaries to enhance the performance of their own GAN model by transfer learning. Specifically, for the target model StyleGAN trained on the 3 million bedroom images [YSZ+15], the adversary first launches a fidelity extraction attack, and the attack performance with 4.12 FID on *fidelity* and 6.97 FID on *accuracy* can be achieved under 50K queries. Furthermore, the adversary transfers the extracted knowledge to new domains, and experimental evaluations show that compared with training from scratch on LSUN-Classroom dataset with 20.34 FID [KALL18], model extraction based transfer learning achieves 16.47 FID, which is the state-of-the-art performance on the LSUN-Classroom dataset.

Defenses. Both fidelity extraction and accuracy extraction attacks on GANs compromise the intellectual property of model providers. In particular, accuracy extraction aiming to steal the distribution of the training set of a target model can further severely breach the privacy of the training set. Therefore, we propose possible defense techniques by considering two aspects: *fidelity* and *accuracy* (Section 3.8). In terms of *fidelity* of model extraction, limiting the number of queries is an effective

method. In terms of *accuracy* of model extraction, we believe that a high accuracy attack model requires adversaries to have access to generated data which can be much closer to real data distribution. The performance of model extraction attacks will be attenuated if adversaries only obtain a partial or distorted distribution of generated data. Thus, we propose two types of perturbation-based defense strategies: input and output perturbation-based approaches, to reveal less distribution information by increasing the similarity of samples or lowering the quality of samples [ASTG19]. The input perturbation-based approaches include linear and semantic interpolation perturbation while the output perturbation-based approaches include random noise, adversarial example noise, filtering and compression perturbation. Extensive experimental evaluations show that, compared to queries from the prior distribution of the target model, the equal amount of queries by perturbation-based defenses can effectively degrade the *accuracy* of attack models (Figure 3.12(a)).

Organization. The rest of this chapter is organized as follows. The next section 3.2 reviews related work. Section 3.3 taxonomizes the space of model extraction attacks on GANs. Section 3.5 and Section 3.6 introduce the fidelity extraction and accuracy extraction, respectively. Section 3.7 presents one case study. In Section 3.8, we discuss possible defense mechanisms. Section 3.9 concludes this chapter.

3.2 Related Work

Generative adversarial networks (GANs). GANs have achieved impressive performance in a variety of areas, such as image synthesis [RMC16, SGZ+16, MKKY18, KALL18, BDS19, KLA19, KLA+20, LCC+19], image-to-image translation [ZKSE16, PLWZ19, LHM+19], and texture generation [LW16, XSA+18], since a framework of GAN was first proposed by Goodfellow et al. in 2014 [GPAM+14]. For image synthesis tasks, the current state-of-the-art GANs [MKKY18, KALL18, BDS19, KLA19] are able to generate highly realistic and diverse images. For instance, SNGAN [MKKY18] generates realistic images by a spectral normalization method to stabilize the training process. PGGAN [KALL18] proposed by Karras et al. is the first GAN that successfully generates real-like face images at a high resolution of 1024×1024, applying a progressive training strategy. Unlike the PGGAN training in an unsupervised method, BigGAN [BDS19] proposed by Brock et al. aims to generate high-quality images from a multi-class dataset by conditional GANs which leverage information about class labels. Recently, StyleGAN [KLA19] has further improved the performance of GANs on high-resolution images through adding neural style transfer [HB17]. In this chapter, *we choose SNGAN and PGGAN as the target models to be attacked by model extraction, considering their impressive performance on image generation. StyleGAN is also used as a target model in a case study in Section 3.7.*

Model extraction attacks. With the availability of machine learning as a service (MLaaS), model extraction attack has received much attention from the research

community [TZJ+16, KTP+20, CJM20, JCB+20, CGZ+21], which aims to duplicate (i.e., 'steal') a machine learning model. This type of attack can be categorized into two classes: accuracy model extraction and fidelity model extraction. In terms of accuracy model extraction, it was first proposed by Tramèr et al. [TZJ+16], where the objective of the attack is to gain similar or even better performance on the test dataset for the extracted model. Since then, various methods attempting to reduce the number of queries have been developed for further improving the attack efficiency, such as model extraction using active learning [PGS+20, CCG+20] or semi-supervised learning [JCB+20]. In terms of fidelity model extraction, it requires the attack model to faithfully reproduce predictions of the target model, including the errors which occur in the target model. Typical works include model reconstruction from model explanation [MSDH19], functionally equivalent extraction [JCB+20] and cryptanalytic extraction [CJM20]. In addition to model extraction attacks on images, there are several work about model extraction in natural language processing [KTP+20, TYF20]. Krishna et al. [TYF20] mount model extraction attacks against BERT-based models and the performance of the extracted model is slightly worse than that of the target model. Overall, these studies mainly focus on discriminative models, such as regression and convolutional neural networks for classification, and recurrent neural networks for natural language processing. *Unlike the existing studies, our work aims to study model extraction attacks against GANs.*

In addition to model extraction attacks, there are other types of attacks in relation to privacy and security [XMSM20, CTW+21, WG18], such as membership inference attacks [SSSS17, SZH+19, SSZ19, HMDDC19, CYZF20] and property inference attacks [GWY+18]. Some efforts have been also made to investigate membership inference attacks against GANs, where queries to a GAN model can reveal information about the training dataset [HMDDC19, CYZF20, HHB19]. Overall, these studies mainly focus on privacy on the training dataset, *while model extraction attacks in this chapter concentrate on machine learning model itself.*

Model extraction defenses. Defense for model extraction can be broadly classified into two categories: restricting the information returned by models [TZJ+16, LEMS19] and differentiating malicious adversaries from normal users [JSMA19]. Tramèr et al. propose a defense where the model should only return class labels instead of class probabilities [TZJ+16]. Recently, a technique PRADA has proposed to guard machine learning models by detecting abnormal query patterns [JSMA19]. Watermarking ML models as a passive defense mechanism recently has been proposed to claim model's ownership [JCCCP21, CJG21, LWZZ19]. However, these defense techniques are used to protect discriminative models where models return probabilities or labels. In this chapter, *we focus on defense approaches safeguarding generative adversarial networks where models return images.*

3.3 Taxonomy of Model Extraction against GANs

In this section, we start with adversary's goal and formally elaborate on our attacks. Next, we illustrate adversary's background knowledge where an adversary can

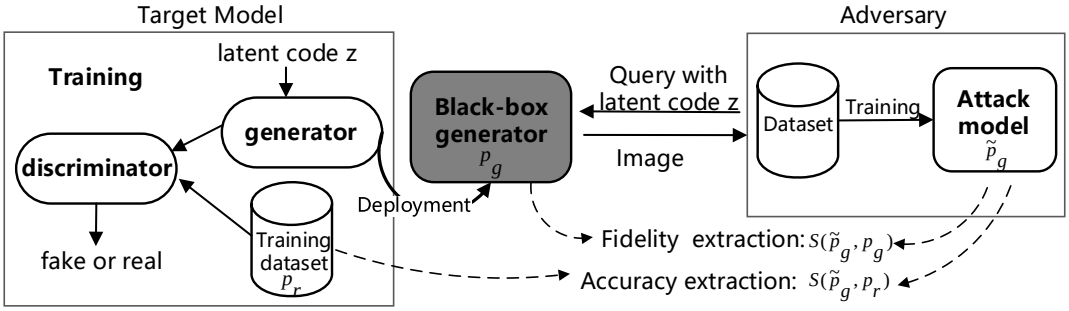


Figure 3.1: Fidelity extraction and accuracy extraction.

mount attacks according to the obtained information. Finally, we detail the metrics to evaluate the attack performance.

3.3.1 Adversary's Goals

In general, model extraction based on adversary's goals can be categorized into either fidelity extraction or accuracy extraction. Unlike supervised discriminative models aiming at minimizing errors on a test set, unsupervised generative models target at learning the distribution of a data set.

Therefore, for model extraction attacks on GANs, fidelity extraction aims to minimize the difference of data distribution between attack models and target models, while accuracy extraction aims to minimize the distribution between attack models and the training set of target models.

Specifically, as shown in Figure 3.1, the goal of fidelity extraction is to construct a $\tilde{G}$ minimizing $S(\tilde{p}_g, p_g)$, where S is a similarity function, $\tilde{p}_g$ is the implicit distribution of the attack generator $\tilde{G}$, and p_g is the implicit distribution of the target generator G. In contrast, accuracy extraction's goal is to construct a $\tilde{G}$ minimizing $S(\tilde{p}_g, p_r)$, where p_r is the distribution of the training set of the target generator G. In this work, we use Fréchet Inception Distance (FID) to evaluate the similarity between two data distributions, mainly considering its computational efficiency and robustness [HRU+17]. It is elaborated in Section 3.3.3. In our work, we study the fidelity extraction in Section 3.5, and accuracy extraction in Section 3.6.

3.3.2 Adversary's Background Knowledge

Adversaries can mount model extraction attacks at different levels based on their obtained information about the target GAN. The more background knowledge adversaries acquire, the more effective they should be in achieving their goals. In general, four components of a GAN can be considered by an adversary. As shown in Figure 3.2, they are respectively: (1) generated data; (2) latent codes used by interactively querying a generator; (3) partial real data from the training dataset of the target GAN; (4) a discriminator from the target GAN.

In the following attack settings, we assume an adversary obtains different levels of background knowledge to achieve accuracy extraction or fidelity extraction.

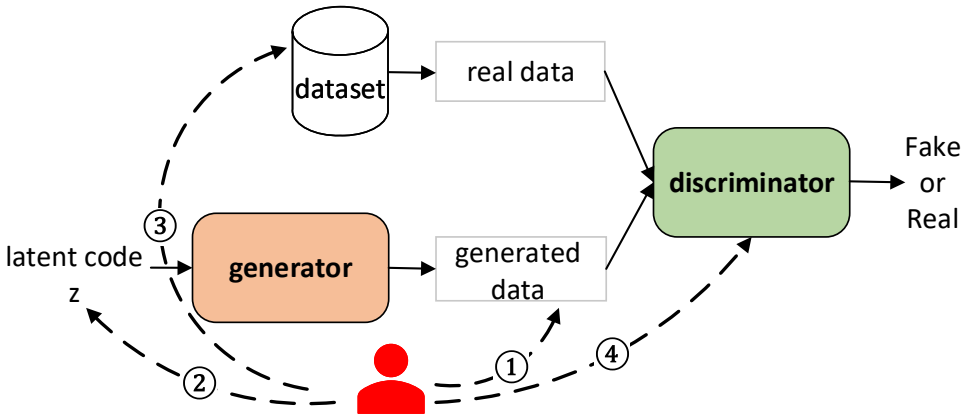


Figure 3.2: Adversary's background knowledge.

Table 3.1: Notations

Notation	Description
p_r	distribution of training set of a GAN
p_g	implicit distribution of a target generator
$\tilde{p}_g$	implicit distribution of an attack generator
fidelity	FID ($\tilde{p}_g$, p_g)
accuracy	FID ($\tilde{p}_g$, p_r)

3.3.3 Metrics

Metrics for GANs. We use the widely adopted FID [HRU+17] to evaluate the performance of GANs. FID measures the similarity between p_g and p_r. Specifically, on the basis of features extracted by the pre-trained Inception network ϕ, it models $\phi(p_r)$ and $\phi(p_g)$ using Gaussian distribution with mean μ and covariance Σ, and the value of FID between real data p_r and generated data p_g in convolutional features is computed as: $FID(p_r, p_g) = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$, where Tr refers to the trace of a matrix in linear algebra. A lower FID indicates that the distribution's discrepancy between the generated data and real-world data is smaller and the generated data is more realistic. In this chapter, FID is computed with all real samples and 50K generated samples.

Metrics for attack performance. In this chapter, we use two FID-based metrics: *fidelity* and *accuracy*, to evaluate the attack performance. *Fidelity* measures the consistency between p_g which is an implicit distribution of a target generator and $\tilde{p}_g$ which is an implicit distribution of an attack generator. Note that, *fidelity* not only measures how close the attack model and the target model are, but also indicates how well the performance of the model itself is. In contrast, *accuracy* measures the consistency of data distribution between p_r and $\tilde{p}_g$. Similar to FID, the smaller the *fidelity* and *accuracy* values are, the better performance attack models achieve. When it is clear from the context, we refer to *accuracy* and *fidelity* as *accuracy* value and *fidelity* value, respectively. The summarized notations of this chapter can be seen in Table 3.1.

Fidelity extraction focuses on *fidelity* and adversaries aim to steal the distribution of a target model. After obtaining an attack model which steals from a target

Table 3.2: Dataset description

Dataset	LSUN-Bedroom	LSUN-Kitchen	CelebA
Size of dataset	3,033,042	2,212,277	202,599
Dataset	LSUN-Classroom	LSUN-Church	
Size of dataset	168,103	126,277	

model, they can directly utilize it to generate new samples. Additionally, they can also transfer knowledge of the stolen model to their own domains through transfer learning. In contrast, accuracy extraction concentrates on *accuracy* and adversaries target at stealing the distribution of the training set of a target model. This type of attack can severely violate the privacy of the training data and it also means that adversaries may steal valuable commercial datasets from a trained GAN. Additionally, adversaries can utilize the stolen high-accuracy model to mount other novel attacks and we leave it for future work.

3.4 Experimental Setups

3.4.1 Dataset Description

We utilize five different datasets in this chapter, which are all widely adopted in image generation. Among them, four datasets are from the LSUN dataset [YSZ+15] which includes 10 scene categories and 20 object categories and we define them as LSUN-Bedroom, LSUN-Church, LSUN-Classroom, and LSUN-Kitchen, respectively. CelebA dataset [LLWT15] consists of about 200K high-quality human face images. Datasets including LSUN-Bedroom, LSUN-Classroom, and LSUN-Kitchen are only used in Section 3.7 to illustrate the attack effects in a case study. The details of the datasets are shown in Table 3.2.

3.4.2 Implementation Details

We implement PGGAN[1] and SNGAN[2] based on following codes indicated in the footnotes. We choose the ResNet architecture for SNGAN and the architecture of PGGAN is the same as the official implementation. We use hinge loss for SNGAN and WGAN-GP loss for PGGAN. For target GAN on synthetic data in Figure 3.5, we use four fully connected layers with ReLU activation for both generator and discriminator and the prior is a 2-dimensional standard normal distribution. The training data is a mixture of 25 2-D Gaussian distributions (each with standard deviation of 0.05). We train it using standard loss function [GPAM+14]. In Section 3.7 about case study, we directly use the pre-trained StyleGAN[3] trained on LSUN-Bedroom dataset as our target model. We resize all images used in our chapter to 64×64, except for the case study where images with a resolution of 256×256 are used. The dimension of latent space of SNGAN, PGGAN and StyleGAN is 256, 512

[1]https://github.com/tkarras/progressive_growing_of_gans
[2]https://github.com/christiancosgrove/pytorch-spectral-normalization-gan
[3]https://github.com/NVlabs/stylegan

and 512, respectively, and their latent codes are all draw from standard Gaussian distribution. For attack models, we use suggested hyperparameters provided by original models and only modify some related to computing resources.

In Section 3.8 about semantic interpolation defense, the semantic information is from attributes of CelebA dataset[4], which has labeled for each image. we only choose 12 (male, smiling, wearing lipstick, mouth slightly open, wavy hair, young, eyeglasses, wearing hat, black hair, receding hairline, bald, mustache) out of 40 facial attributes to learn semantic hyperplanes, because the number of images for each attribute varies largely and some attributes is hard to distinguish when they are applied in target GAN model. We train the prediction model for each attribute based on ResNet-50 model pre-trained on ImageNet[5]. The magnitude of semantic interpolation is set as 3.

3.5 Fidelity Extraction

In this section, we instantiate our fidelity extraction attack strategy. we assume that adversaries have access to either generated samples provided by the model producer or querying the target model to obtain data (see Figure 3.2). We start with target models and attack models. Then, we describe our attack performance. Next, we study the effect of the number of queries. In the end, we perform experiments to deeply understand model extraction on GANs.

3.5.1 Target Models and Attack Models

We choose representative GANs: Progressive GAN (PGGAN) [KALL18] and Spectral Normalization GAN (SNGAN) [MKKY18] as our target models, which both show pleasing performances in image generation. The implementation details can be seen in Section 3.4.2. For training sets LSUN-Church and CelebA, we first resize them to 64×64 and use all records of each dataset to train our target models. As shown in Table 3.3, target GAN models achieve an excellent performance on these datasets and the performance of PGGAN is better than that of SNGAN.

We use GANs as our attack models to extract target models. In practice, adversaries may not know the target model's architecture. Therefore, we study the performance of attack models with different architectures. Specifically, we choose SNGAN and PGGAN as our attack models. There are four different situations for their combinations. For simplification, we define each situation as an attack-target model pair, and they are respectively SNGAN-SNGAN, SNGAN-PGGAN, PGGAN-SNGAN and PGGAN-PGGAN. The reason why we choose SNGAN and PGGAN as the research object is that: 1) they both show good performance in image generation; and 2) they have significant differences in the aspects of training, loss function and normalization, which all facilitate us to study the performance of attack models with different architectures.

[4]http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[5]https://download.pytorch.org/models/resnet50-19c8e357.pth

Table 3.3: Performance of target GANs.

Target model	Dataset	FID
SNGAN	LSUN-Church	12.72
SNGAN	CelebA	7.60
PGGAN	LSUN-Church	5.88
PGGAN	CelebA	3.40

3.5.2 Methodology

As shown in Figure 3.1, for fidelity extraction, we assume that an adversary obtains the generated data by the model provider or querying the target GAN. This scenario is practical, because some model owners need to protect their models through providing the public with some generated data or a black-box GAN model API. In this case, the adversary uses the generated data to retrain a GAN to extract the target model. We do not distinguish whether generated data is from queries or model providers, because our approach only relies on these generated data. However, in Section 3.5.3, we also present the attack performance on queries with different prior distributions.

Note that model extraction on GANs is different from machine learning on GANs. This is because machine learning on GANs requires users to train a GAN on real samples which are collected from the real world. In contrast, model extraction on GANs enables users to train a GAN on generated data from a target GAN model. In essence, model extraction on GANs approximates the target GAN which is a much simpler deterministic function, compared to real samples which usually represent a more complicated function.

3.5.3 Results

Attack performance on different models. Table 3.4 shows the fidelity extraction's performance with 50K queries to the target model. In general, attack models can achieve an excellent performance[6]. For instance, our attack performance of PGGAN-PGGAN on the CelebA achieves 1.02 FID on *fidelity*, which means that the attack model can achieve a perfect extraction attack for the target model. It is noticeable that the attack model achieves such performance only on 50K generated images while the target model is trained on more than 200K images. In Section 3.7, our case study further illustrates that even for a GAN model trained on 3 million samples, our attack still can achieve 4.12 *fidelity* with only 50K queries. In other words, adversaries are able to obtain a good GAN model only by access to the generated data from the target model instead of collecting their own data which is usually labor-intensive and time-consuming.

For the target model PGGAN, if the attack model is SNGAN, we observe that the performance of model extraction is very efficient on both CelebA and LSUN-Church

[6]We say model extraction attacks achieve an excellent performance because we choose the state-of-the-art StyleGAN [KLA19] trained on the LSUN-Bedroom dataset as a reference, where it has the lowest FID 2.65.

Table 3.4: The performance of fidelity extraction with 50K queries to the target model.

Target model	Attack model	Dataset	*Fidelity* FID($\tilde{p}_g$, p_g)	*Accuracy* FID ($\tilde{p}_g$, p_r)
PGGAN	SNGAN	LSUN-Church	6.11	14.05
	SNGAN	CelebA	4.49	9.29
	PGGAN	LSUN-Church	1.68	8.28
	PGGAN	CelebA	1.02	4.93
SNGAN	SNGAN	LSUN-Church	8.76	30.04
	SNGAN	CelebA	5.34	17.32
	PGGAN	LSUN-Church	2.21	14.56
	PGGAN	CelebA	1.39	9.57

datasets and the attack model SNGAN can learn more from the target model PGGAN, compared to the SNGAN-SNGAN case, which indicates that attacking a state-of-the-art GAN is valuable and viable for an adversary. Furthermore, this case SNGAN-PGGAN is the most common situation in the actual attack scenarios, because generally we implicitly assume that the performance of the adversary's model may often be weaker than that of the target model and the structure of the attack model is inconsistent with that of the target model.

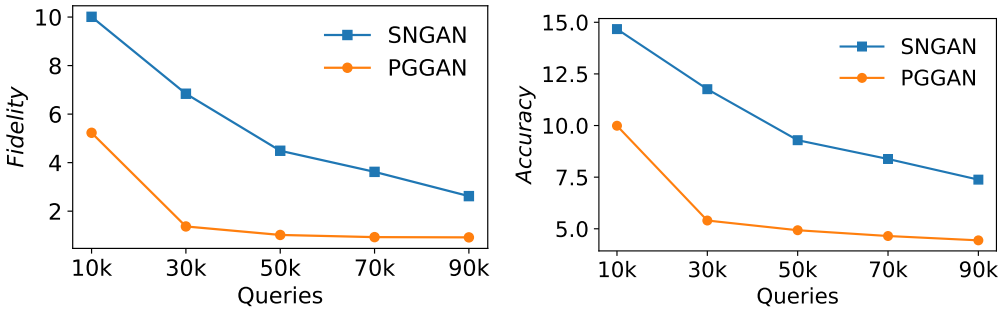
We also report *accuracy* in Table 3.4 and find that for model extraction on GAN models, the *accuracy* of attack models is always higher than that of the target model, in which *accuracy* of attack models represents the similarity between distribution of real dataset p_r and distribution of the attack model $\tilde{p}_g$ and for *accuracy* of a target model, also called FID of target model, it represents the similarity between distribution of real dataset p_r and distribution of the target model p_g. For example, when the target model SNGAN has 12.72 FID on the LSUN-Church dataset, *accuracy* of the attack model SNGAN will increase to 30.04. Even for the PGGAN-PGGAN case, its *accuracy* increases from 3.40 to 4.93 on the CelebA dataset. This is mainly because although theoretically, the distribution of the target model p_g is equal to that of the real training dataset p_r, it is actually not equal because GAN cannot achieve the global optimum. However, we will discuss how to reduce *accuracy* values and achieve high accuracy extraction in Section 3.6.

For the target model SNGAN, if the attack model is PGGAN, the *fidelity* of model extraction is lower than that of the attack model SNGAN. It is mainly because the PGGAN model itself is stronger and able to more accurately approximate the target model. Similarly, PGGAN as an attack model has more lower *accuracy*, in contrast with SNGAN as an attack model. For instance, compared to SNGAN-SNGAN with 17.32 of *accuracy* on CelebA dataset, the *accuracy* of PGGAN-SNGAN is only 9.57, which largely improves the attack performance on accuracy. This indicates that using an attack model which is larger than the target model is an efficient approach to improve attack performance.

Overall, fidelity extraction can achieve an excellent performance in terms of *fidelity*. In general, adversaries can steal a fidelity model, and then use the extracted model for their own purpose. However, unlike discriminative models where adversaries can directly utilize their extracted model, the extracted model of a GAN only generates target model's images. Therefore, in Section 3.7, we will perform a case study

Table 3.5: Performance of fidelity extraction attack with different prior distributions. We use standard normal distribution and uniform distribution over an interval -1 and 1 to generate latent codes. The number of queries is fixed to 50K.

Attack model	Prior distribution	*Fidelity* FID($\tilde{p}_g$, p_g)	*Accuracy* FID ($\tilde{p}_g$, p_r)
PGGAN	Gaussian	1.02	4.93
PGGAN	Uniform	0.98	4.85
SNGAN	Gaussian	4.49	9.29
SNGAN	Uniform	4.29	9.16



(a) *Fidelity* on CelebA (b) *Accuracy* on CelebA

Figure 3.3: Attack performance on the number of queries.

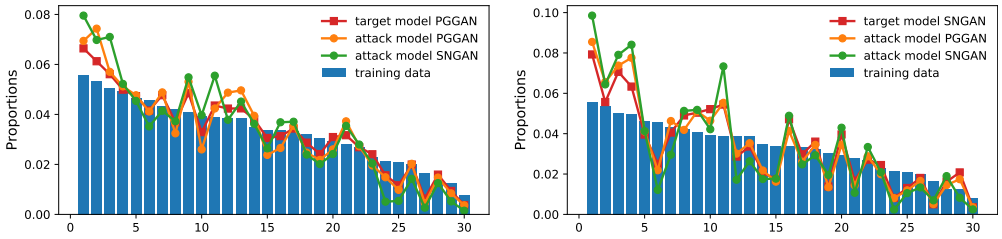
where adversaries can effectively leverage the extracted model to generate images for their own applications rather than target GANs' images through transfer learning.

Attack performance on queries from different prior distributions. Adversaries can query the target model via trying common prior distributions to generate latent codes if they do not know the prior distribution of a target model. Gaussian distribution and uniform distribution are widely used in almost all GANs [KALL18, BDS19, RMC16, MKKY18, KLA19, KLA+20]. Table 3.5 shows the attack performance with two prior distributions. We choose PGGAN trained on CelebA dataset with standard normal prior distribution as the target model. From Table 3.5, we find that adversaries can obtain a similar attack performance no matter what the prior distribution of latent codes is.

Attack performance on the number of queries. We choose PGGAN trained on CelebA dataset as the target model to study the effect of the number of queries due to the best performance among our target models. Figure 3.3 plots the attack performance with respect to the number of queries which are also the size of the training dataset of attack models. As expected, we observe that the attack performance increases with an increase in the number of queries. This indicates that releasing a small number of data by the model owner or restricting the number of queries is a relatively safe measure.

We estimate the monetary cost of the number of queries. Taking the Google Cloud Vision API[7] as an example, the price is $1.50 per 1K queries with the first 1K queries

[7]https://cloud.google.com/vision/pricing



(a) Class distribution differences among the training data, the target model PGGAN, and attack models.

(b) Class distribution differences among the training data, the target model SNGAN, and attack models.

Figure 3.4: Class distribution differences.

Table 3.6: JS distances between models. A smaller value indicates a better performance. The JS distance between the training data and the target model PGGAN is 4.14×10^{-3}. The JS distance between the training data and the target model SNGAN is 16.36×10^{-3}. The JS value shows a consistent trend with Figure 3.4.

Target model	Attack model	$JS_{fidelity}$ ($\times 10^{-3}$)	$JS_{accuracy}$ ($\times 10^{-3}$)
PGGAN	SNGAN	5.88	15.95
	PGGAN	1.83	9.10
SNGAN	SNGAN	8.90	34.12
	PGGAN	1.60	18.56

are free for each month. Thus, the price of the number of queries from 10K to 90K is from $13.50 to $133.50. Although the attack cost is not high in our attacks, designing a more powerful attack to reduce the number of queries is still an interesting research direction. We leave it as future work.

Understanding fidelity extraction on GANs in-depth. We further dissect the difference of distributions between target models and attack models to understand the nature of model extraction on GANs. Specifically, we first transform the training data into 2048-dimension feature vectors by the pre-trained Inception-v3 model[8] which is widely utilized in the evaluation of a GAN model [HRU+17]. Then these feature vectors are clustered into k classes by a standard K-means algorithm. Finally, we calculate the proportions of each class, which can be also considered as a distribution of the training data [BZW+19, RW18]. The blue bar in Figure 3.4(a) shows the distribution of the training data where we set k to 30. For target models and attack models, we query the model to obtain 50K images, then perform the same procedures as the training data.

Figure 3.4(a) shows distribution differences among the training data, the target model PGGAN and attack models. We observe that for the high proportions of classes, which can be considered as prominent features of a distribution, target models can learn more features about these classes while attack models further learn more features by querying the target models. In contrast, for the low proportions of classes, target models learn less features about these classes while attack models further learn less features about these classes. This is one reason why attack models always have higher *accuracy* values than target models. In terms of

[8]https://pytorch.org/hub/pytorch_vision_inception_v3/

fidelity, we observe that there is a consistent trend on proportions of classes for target models and attack models. This is the reason why we can achieve a satisfying performance about *fidelity*.

We also summarize this difference in a single number by computing the Jensen-Shannon (JS) divergence on this representation of distributions, which is shown in Table 3.6. Note that, based on *accuracy* and *fidelity* defined in Section 3.3.1, we mark $JS_{fidelity}$ as the JS divergence between the target model and the attack model, and $JS_{accuracy}$ as the JS divergence between the training data and the attack model.

We also analyze the target model SNGAN trained on CelebA dataset, and similar results are shown in Figure 3.4(b) and Table 3.6.

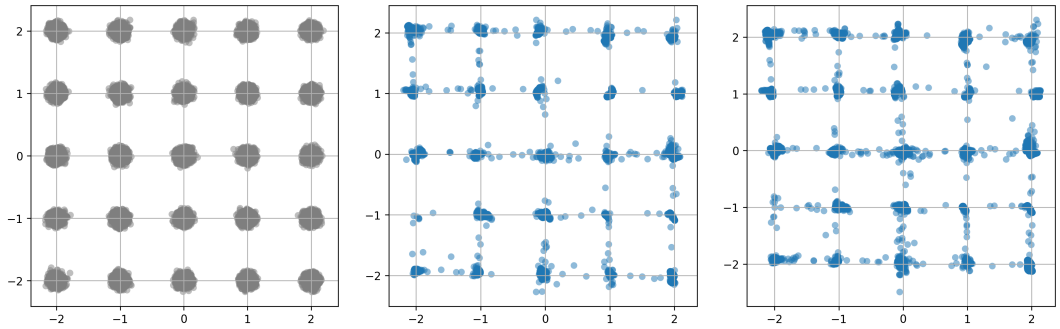
3.6 Accuracy Extraction

In this section, we instantiate our accuracy extraction attack strategy. In addition to fidelity extraction's assumptions, we also assume that adversaries have more background knowledge in order to achieve accuracy extraction, such as partial real data and the target model's discriminator. We start with the motivation and problem formulation of accuracy extraction. Then, we describe the methodology of accuracy extraction. In the end, we present the performance of accuracy extraction.

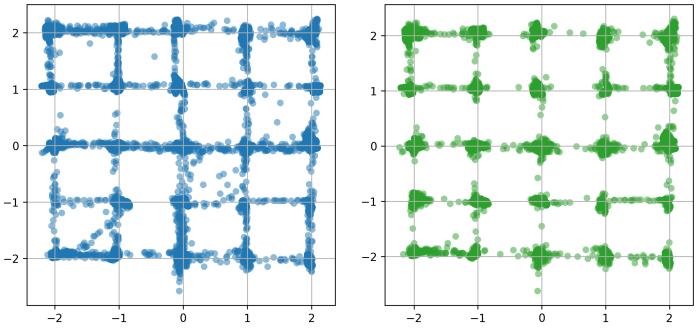
3.6.1 Motivation and Problem Formulation

As shown in Figure 3.1, fidelity extraction can be implemented through querying the generator of the target GAN, because p_g is the generator's distribution. As for accuracy extraction, it is much more difficult due to the lack of availability of real data distribution p_r. Although an approach is to use p_g as an approximation of p_r, we observe that with the increase in the number of queries, *accuracy* of attack models reaches its saturation point and is hard to be improved, which is shown in Figure 3.3(b). For instance, as we increase the number of queries from 50K to 90K for the PGGAN-PGGAN case on CelebA dataset, *accuracy* of the attack model has smaller and smaller improvements from 4.93 to 4.44, while the ideal *accuracy* is 3.40 which is also the performance of the target model. Note that the case PGGAN-PGGAN is the best for the attacker; the attack will perform even worse if the attackers do not choose the same architectures and hyperparameters as the target model.

The reason why there exists a gap between the attack model and the target model in terms of *accuracy* is that the target GAN model is hard to reach global equilibrium and the discriminator is often better than the generator in practice [AOD+19]. As a result, real data distribution p_r is not completely learned by the generator of the target model, which means that $p_g \neq p_r$. Therefore, directly using the generator's distribution p_g does not guarantee the high accuracy and it only minimizes the distribution discrepancy between the attack model and the target model. We explain this by a simple example on Figure 3.5, which is popular in the GAN literature [AOD+19, THF+19, DWW20].



(a) Real Samples 50K (b) Direct Sampling 5k (c) Direct Sampling 10K



(d) Direct Sampling 50K (e) MH Subsampling 50K

Figure 3.5: Difference of distribution between training data and generators. The percentage of "high-quality" samples for Figure 3.5(b), Figure 3.5(c), Figure 3.5(d) and Figure 3.5(e) is 94.36%, 94.31%, 94.15% and 95.64%, respectively. The more we query, the more bad-quality samples we obtain, which affects the performance of model extraction. But if we reduce the number of queries, the performance of attack models still be poor due to insufficient training samples.

Figure 3.5(a) presents real samples drawn from a mixture of 25 two-dimensional Gaussian distributions (each with standard deviation σ of 0.05). Figure 3.5(b) - Figure 3.5(d) show samples which are generated by a target GAN with different queries. We define a generated sample as "high-quality" if its Euclidean distance to its corresponding mixture component is within four standard deviations (4σ = 0.2) [AOD+19]. The architecture and setup information of the target GAN is shown in Section 3.4.2. Overall, we can observe that target GAN's distribution is not completely the same as the training set's distribution, which means that directly extracting a model from the generator of the target GAN makes its distribution similar to the target model's distribution rather than its training dataset's distribution.

Therefore, a natural approach to achieving accuracy extraction is that the adversary can get more high-quality samples that are closer to the real data distribution.

3.6.2 Methodology

Our approach to obtaining high-quality samples is based on subsampling. The key insight here is that we can reject some poor samples from generated samples based on some prior knowledge. In order to achieve it, we suppose that adversaries can obtain additional background information. This is a common assumption that can be found in many works in relation to the security and privacy of machine

Algorithm 1: MH subsampling

Input: target generator G, target discriminator D,
partial real samples $X_r = \{x_{r1}, x_{r2}, \ldots, x_{rm}\}$
Output: N refined images

Sample m fake images $X_g = \{x_{g1}, x_{g2}, \ldots, x_{gm}\}$ from G
Train a calibrated classifier:
$C \leftarrow LogisticRegression(D(X_r), D(X_g))$
$images \leftarrow \emptyset$
while $|images| < N$ **do**
  $x \leftarrow$ a real image from X_r
  for $i = 1$ to K **do**
    Sample x' from G
    Sample u from Uniform$(0, 1)$
    Compute real image's density ratio:
    $r(x) = \frac{C(D(x))}{1-C(D(x))}$
    Compute fake image's density ratio:
    $r(x') = \frac{C(D(x'))}{1-C(D(x'))}$
    $p = min(1, \frac{r(x')}{r(x)})$
    if $u \leq p$ **then**
      $x \leftarrow x'$
    end if
  end for
  if x is not a real images **then**
    Append$(x, images)$
  end if
end while

learning [JLG+15, SSSS17, HMDDC19]. As shown in Figure 3.2, we assume that adversaries can have limited auxiliary knowledge of the discriminator of the target model and partial training samples. This is because the discriminator from the target model can reveal the distribution information of the training data [AOD+19]. Thus, using the information provided by the discriminator, we can subsample the generated data to make the obtained data closer to the real dataset's distribution, which improves accuracy extraction.

Specifically, for accuracy extraction, we first leverage the discriminator of the target model to subsample the generated samples. As a result, these refined samples are much closer to the true distribution. In this work, we use Metropolis-Hastings (MH) subsampling algorithm [THF+19] to subsample the generated data. MH subsampling algorithm utilizes the discriminator through Metropolis-Hastings algorithm [Tie94] to refine samples which are generated by the generator. The discriminator generally needs to be calibrated by partial real samples from training set of the target GAN model, considering that some discriminators of GANs output a score rather than a probability.

We detail MH subsampling in Algorithm 1. Inputs of this algorithm are a target generator (only used to query), a white-box discriminator which is used to subsample generated samples and partial real samples which are used to calibrate the

discriminator (Algorithm 1, line 2). Outputs are refined samples whose distribution is much closer to the distribution of real training data. In our experiments, all discriminators are calibrated through logistic regression. Then we train the attack model on those refined samples. After the training process of the attack model is stable, we add partial real data to further train the attack model.

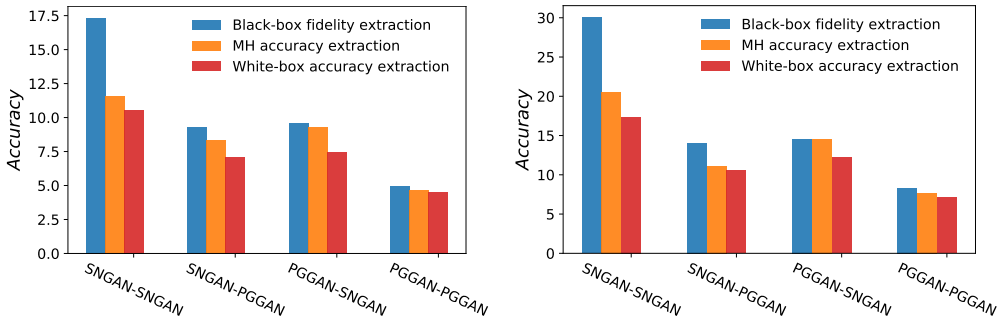
In this scenario, although the number of queries will increase due to subsampling samples, we assume that adversaries eventually obtain 50K refined samples in order to make a comparison with fidelity extraction. Partial real samples used to calibrate the discriminator are fixed to 10% of training data. In addition, these partial real samples will be added into training process of the attack models. Here, we refer the former where only refined samples are used to train the attack model to MH accuracy extraction which is also considered as an indicator to show how well these refined samples are beneficial to *accuracy*. We refer the latter where both refined samples and partial real data are used to train the attack model to white-box accuracy extraction. We refer fidelity attack in Section 3.5 to black-box fidelity extraction.

It is worth noting that we cannot directly choose the lowest *accuracy* value in real attack scenarios due to unavailability of training dataset from target models. Therefore, the *accuracy* value reported in this chapter is chosen when its corresponding *fidelity* value is the lowest in the training process.

3.6.3 Results

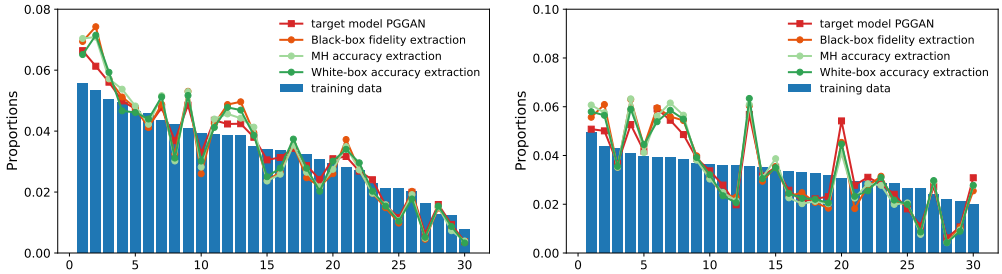
Attack performance. Figure 3.6 plots not only the results of the MH accuracy extraction and the white-box accuracy extraction on both CelebA and LSUN-Church datasets, but also the black-box fidelity extraction for comparison. We can observe that MH subsampling is an effective approach to improve *accuracy* of attack models. For example, when target model is SNGAN, the MH accuracy extraction can significantly improve attack model's accuracy on both datasets because MH subsampling algorithm selects high-quality samples from generated samples of the target model SNGAN. For the MH accuracy extraction and the white-box accuracy extraction which both leverage the refined samples in the training process, the white-box accuracy extraction can further improve *accuracy*. This is because partial real data can further correct the distribution of the attack model and make it closer to the real distribution.

Understanding accuracy extraction on GANs in-depth. Following the same procedure illustrated in Section 3.5.3, we also dissect distribution differences for accuracy extraction. Specifically, we choose the PGGAN-PGGAN case as an example (see Figure 3.6) and the attack models is PGGAN. From Figure 3.7, we observe that for CelebA, white-box accuracy extraction which has minimal *accuracy* values among these methods is more consistent with the distribution of the training data by lowering the highest proportions of classes. For LSUN-Church, similar results also can be observed. Table 3.7 summarizes these differences statistically.



(a) *Accuracy* on CelebA (b) *Accuracy* on LSUN-Church

Figure 3.6: Comparison on *accuracy* for different attack approaches.



(a) The target model PGGAN trained on CelebA. (b) The target model PGGAN trained on LSUN-Church.

Figure 3.7: Distribution differences for accuracy extraction.

Table 3.7: JS distances between models. For the JS distance between training data and the target model, the target model PGGAN on CelebA is 4.14×10^{-3} and the target model PGGAN on LSUN-Church is 14.78×10^{-3}.

Dataset	Methods	$JS_{fidelity}$ ($\times 10^{-3}$)	$JS_{accuracy}$ ($\times 10^{-3}$)
CelebA	Black-box fidelity extraction	1.83	9.10
	MH accuracy extraction	1.42	8.17
	White-box accuracy extraction	1.17	7.53
LSUN-Church	Black-box fidelity extraction	2.32	19.14
	MH accuracy extraction	2.28	19.89
	White-box accuracy extraction	1.61	18.65

3.7 Case Study: Model Extraction based Transfer Learning

In this section, we present one case study where the extracted model serves as a pre-trained model and adversaries transfer knowledge of the extracted model to new domains by means of fine-tuning to broaden the scope of applications based on extracted models. We start with methods of transfer learning on GAN and demonstrate how adversaries can benefit from model extraction, in addition to directly leveraging the extracted model to generate images.

We consider the state-of-the-art GAN model StyleGAN [KLA19] that was trained on more than 3 million bedroom images as the target model. StyleGAN produces high-quality images at a resolution of 256×256, with 2.65 FID on LSUN-Bedroom dataset [YSZ+15]. We suppose adversaries only query the target model StyleGAN

and have no other background knowledge, which is also called black-box fidelity extraction in our chapter. Although an adversary can obtain an extracted model, the model only generates images which are similar to the target model. In this case, the extracted model can only generate bedroom images due to the target model trained on LSUN-Bedroom dataset. Therefore, the adversary's goal is to use the PGGAN as the attack model to extract the target model StyleGAN and leverage transfer learning to obtain a more powerful GAN which generates images that the adversary wishes. *The attack is successful if the performance of models training by transfer learning based on the extracted GAN outperforms models training from scratch.*

Transferring knowledge of models which steal the state-of-the-art models to new domains where adversaries wish the GAN model can generate other types of images can bring at least two benefits: 1) if adversaries have too few images for training, they can easily obtain a better GAN model on limited dataset through transfer learning; 2) even if adversaries have sufficient training data, they can still obtain a better GAN model through transfer learning, compared with a GAN model training from scratch. Therefore, we consider two variants of this attack: one where the adversary owns a small target dataset (i.e., about 50K images in our work) and the other one where the adversary has enough images (i.e., about 168k images in our work).

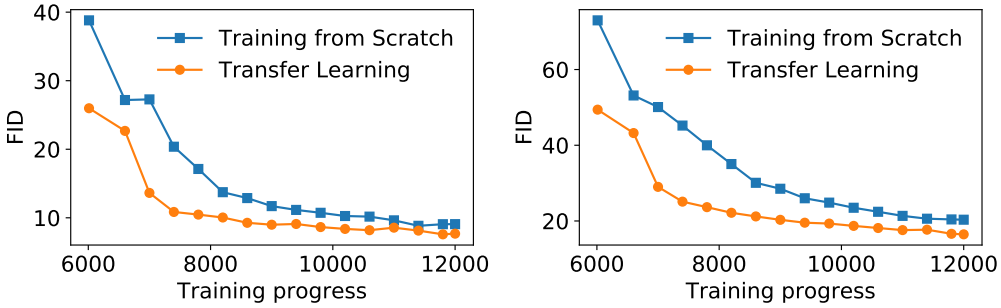
More specifically, after querying the target model StyleGAN and obtaining 50K generated images, adversaries train their attack model PGGAN on the obtained data, as illustrated in Section 3.5.2. Here, *fidelity* of the attack model PGGAN is 4.12 and its *accuracy* is 6.97. Then, we use the extracted model's weights as an initialization to train a model on the adversary's own dataset which is also called the target dataset in the section. We conduct the following two experiments:

1. We first randomly select 50K images from LSUN-Kitchen dataset as a limited dataset. Then, we train the model on these selected data by transfer learning and from scratch, respectively.

2. We train a model on the LSUN-Classroom dataset including about 168k images by transfer learning and from scratch, respectively.

Results. Table 3.8 shows the performance of models trained by transfer learning and training from scratch. We can observe that the performance of training by transfer learning is always better than that of training from scratch on both large and small target datasets. To be specific, on the limited LSUN-Kitchen dataset which contains 50K images, the FID of the model trained by transfer learning decreases from 8.83 to 7.59, compared with the model trained from scratch. It indicates that the extracted model is useful for models trained on other types of images. On the large LSUN-Classroom dataset which contains more than 168k classroom images, the performance of the model significantly improves from model training from scratch with 20.34 FID[9] to training by transfer learning with 16.47 FID. This is also the best performance for PGGAN on the LSUN-Classroom dataset, in contrast with

[9]This value is not equal to 20.36 [KALL18] due to randomness.



(a) FID on LSUN-Kitchen (b) FID on LSUN-Classroom

Figure 3.8: Comparison between transfer learning based on model extraction and training from scratch on LSUN-Kitchen and LSUN-Classroom dataset.

Table 3.8: Comparison between transfer learning based on model extraction and training from scratch. The target model is StyleGAN trained on the LSUN-Bedroom dataset, and the attack model uses PGGAN.

Target dataset	Methods	FID
LSUN-Kitchen	Transfer Learning	7.59
LSUN-Kitchen	Training from Scratch	8.83
LSUN-Classroom	Transfer Learning	16.47
LSUN-Classroom	Training from Scratch	20.34

20.36 FID reported by Karras et al. [KALL18]. We also plot the process of training for both settings on the two datasets, which is shown in Figure 3.8. We can obviously and consistently observe that training by transfer learning based on model extraction is always better than training from scratch during the training process, which indicates that the extracted model PGGAN which duplicates the state-of-the-art StyleGAN on the LSUN-Bedroom dataset can play a significant role in other applications rather than only on generating bedroom images. That reminds us that model extraction on GANs severely violates the intellectual property of the model owners.

3.8 Defenses

Model extraction attacks on GANs leverage generated samples from a target GAN model to retrain a substitutional GAN which has similar functions to the target GAN. In this section, we introduce defense techniques to mitigate model extraction attacks against GANs.

According to the adversary's goals as defined in Section 3.3.1, we discuss defense measures from two aspects: *fidelity* and *accuracy*. In terms of *fidelity* of model extraction, it is difficult for model owners to defend except for limiting the number of queries. This is because adversaries can always design an attack model to learn the distribution based on their obtained samples. The more generated samples adversaries obtain, the more effective they achieve.

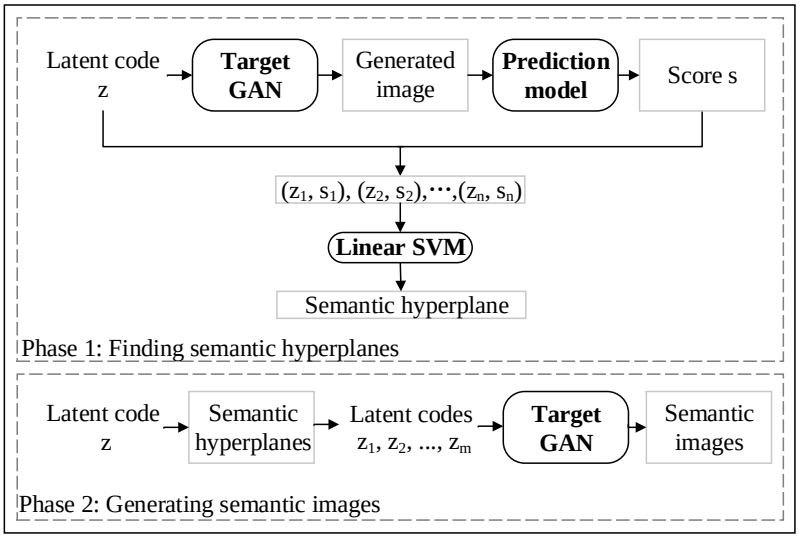


Figure 3.9: Semantic interpolation defense.

In terms of *accuracy* of model extraction, its effectiveness is mainly because adversaries are able to obtain samples generated by latent codes drawn from a prior distribution of the target model, and these samples generated through the prior distribution are close to real data distribution [ASTG19]. However, if adversaries obtain some generated samples which are only representative for partial real data distribution or a distorted distribution, *accuracy* of attack models becomes poor. Based on this, we propose two types of perturbation-based defense mechanisms: input perturbation-base and output perturbation-based approaches. In the rest of this section, we focus on defense approaches which are designed to mitigate *accuracy* of attack models.

3.8.1 Methodology

Input perturbation-base defenses. For this type of defense, we propose two approaches based on perturbing latent codes: linear interpolation defense and semantic interpolation defense.

• **Linear interpolation defense.** For n latent codes queried from users, model providers randomly select two queried points and interpolate k points between the two points. This process is repeated for $\lceil n/k \rceil$ times to get n modified latent codes. These modified latent codes are used to query the target model. In our experiments, we interpolate 9 points. See Figure 3.10(a) for visualization.

• **Semantic interpolation defense.** Unlike linear interpolation defense where target models return a batch of random images, semantic interpolation defense returns various semantic images that are predefined by model providers, which restricts the space of images the adversary queries. Generally, semantic information can be any information that humans can perceive. For instance, for a human face image, it includes gender, age and hair style. We adopt the semantic interpolation algorithm proposed by Shen et al. [SGTZ20].

Specifically, the process of semantic interpolation defense is shown in Figure 3.9. Semantic interpolation defense consists of two phases: finding semantic hyperplanes and generating semantic images. In the first phase, we first train a prediction model for each semantic information. Then the trained prediction model is used to predict semantic score s for each image generated through latent code z. As a result, we get latent code-score pairs and label the highest k scores as positive and the lowest k scores as negative. Finally, we train a linear support vector machine (SVM) on dataset where latent codes as training data and scores as labels. A trained linear SVM contains a hyperplane which separates one semantic information. In the second phase, we can obtain a semantic image for each semantic hyperplane through interpolation. A latent code interpolates points along the normal vector of the hyperplane and corresponding semantic images can be obtained. In our experiments, we train each prediction model for each semantic information, and the prediction model is built on the basis of ResNet-50 network [HZRS16] trained on ImageNet dataset [RDS+15]. In this work, we totally explore 12 semantic information on CelebA dataset. See Figure 3.10(b) for visualization.

Output perturbation-base defenses. Instead of perturbing latent codes, this type of defense directly perturbs the generated samples. Specifically, we propose four approaches: random noise, adversarial noise, filtering and compression. See Figure 3.11 for visualization.

• **Random noise.** Adding random noises on generated samples is a straightforward method. In our experiments, we use Gaussian-distributed additive noises (mean = 0, variance = 0.001).

• **Adversarial noise.** We generate adversarial examples through mounting targeted attacks where all images are misclassified into a particular class by the classifier ResNet-50 trained on ImageNet dataset. In our experiments, all face images are misclassified into the class — goldfish and the C&W algorithm [CW17b] based on L_2 distance are used.

• **Filtering.** The Gaussian filter is used to process generated samples. In our experiments, we use Gaussian filter (sigma = 0.4) provided by the skimgae package [vdWSN+14].

• **Compression.** The JPEG compression algorithm is used to process generated samples. In our experiments, we use the JPEG compression (quality = 85) provided by the simplejpeg package [Fol20].

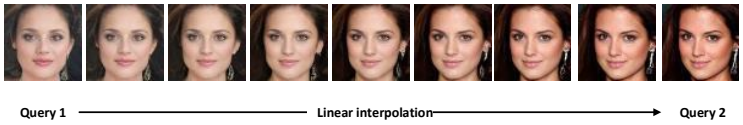
3.8.2 Results

In this experiment, we choose PGGAN trained on CelebA dataset as the target model to evaluate our defense techniques, considering its excellent performance among our target models. We only show the effectiveness of defense techniques on black-box fidelity extraction, considering its more practical assumption: adversaries obtain samples by model providers or queries.

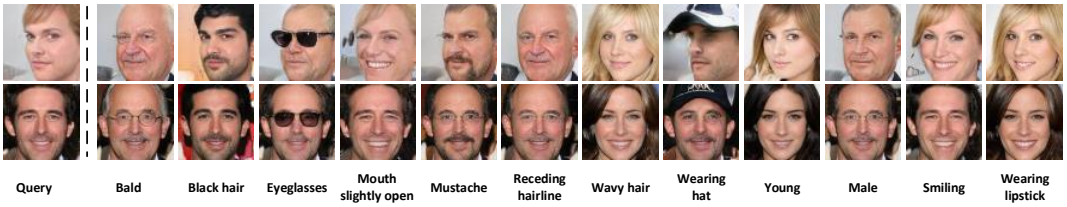
Table 3.9: Defense utility. Each score is an average of 50K image score. Lower is better.

Metrics	No Defense	Semantic	Linear	Gaussian Noise
NIQE	18.87	18.87	18.87	18.87
PIQE	42.66	40.04	42.62	23.64

Metrics	JPEG Compression	Gaussian Filter	Adversarial Noise
NIQE	18.87	18.87	18.87
PIQE	35.99	47.80	37.91



(a) Linear interpolation defense. These linear interpolated images are returned.



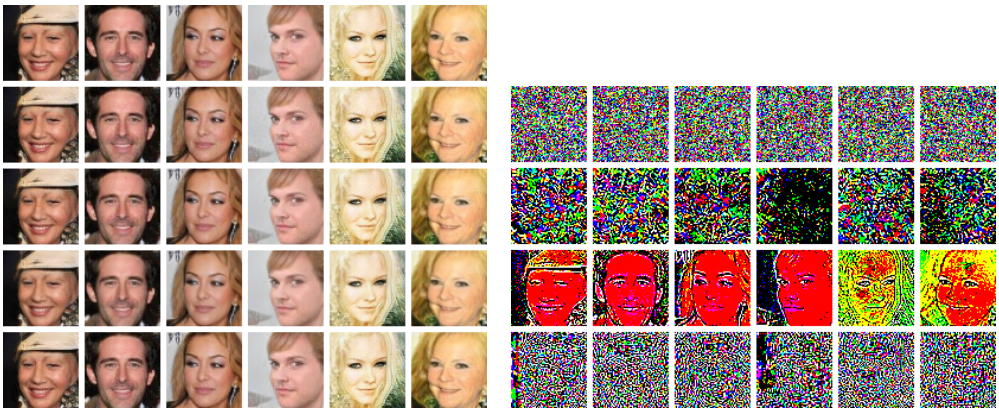
(b) Semantic interpolation defense. For one latent code, 12 latent codes containing semantic information are generated through semantic interpolation and corresponding images are shown above. These semantic interpolated images are returned.

Figure 3.10: Returned images after input perturbation-based defense techniques. Queried images and interpolated images both show good quality in visual comparison, and images generated by linear interpolation show more similarity than that by semantic interpolation.

Defense utility. We quantitatively and qualitatively evaluate the defense utility, i.e. the quality of generated images after deploying defense measures. Figure 3.10 and Figure 3.11 show returned images for input perturbation-based and output perturbation-based defenses. Table 3.9 shows image quality scores. We use two widely-adopted no-reference image quality scores: Naturalness Image Quality Evaluator (NIQE) [MSB12] and Perception based Image Quality Evaluator (PIQE) [VPB+15]. Overall, our defense measures do not significantly impact the quality of generated images.

Defense on black-box fidelity extraction. Figure 3.12(a) plots results of attack model PGGAN on defenses. We observe that attack performance is weakened when the target model PGGAN uses these defense approaches, compared to the target model without any defenses. Gaussian noise and semantic interpolation defenses show stable performance while other defense techniques' performance is gradually weakened with an increase in the number of queries. Figure 3.12(b) also shows similar defense performance for the attack model SNGAN. We further evaluate the defense utility, i.e. the quality of generated images after deploying defense measures. Our quantitative and qualitative measures show that these defense techniques do not impact the visual quality of generated images (see Figure 3.10, Figure 3.11 and Table 3.9).

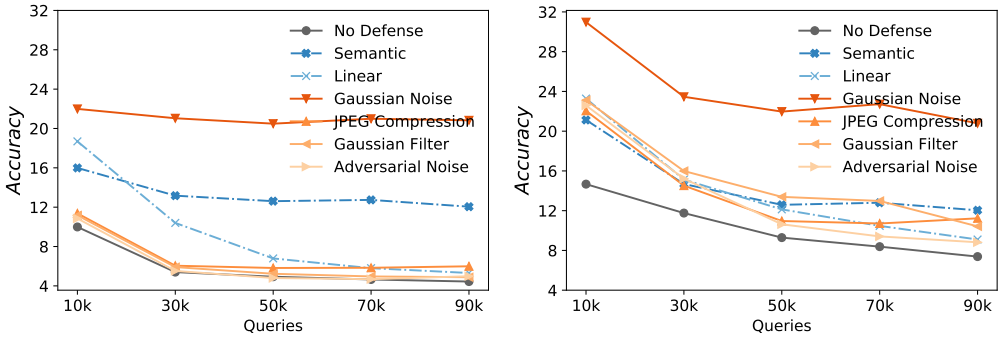
Defense in terms of fidelity. Figure 3.13 shows *fidelity* of attack models under



(a) Output images. From top to bottom: generated images, Gaussian noise images, Adversarial noise images, Gaussian filter images and JPEG compression images.

(b) Noises. For the top two rows, they are Gaussian noises and adversarial noises, respectively. For the third row, it is the differences between Gaussian filter images and generated images. For the last row, it is the differences between JPEG compression images and generated images.

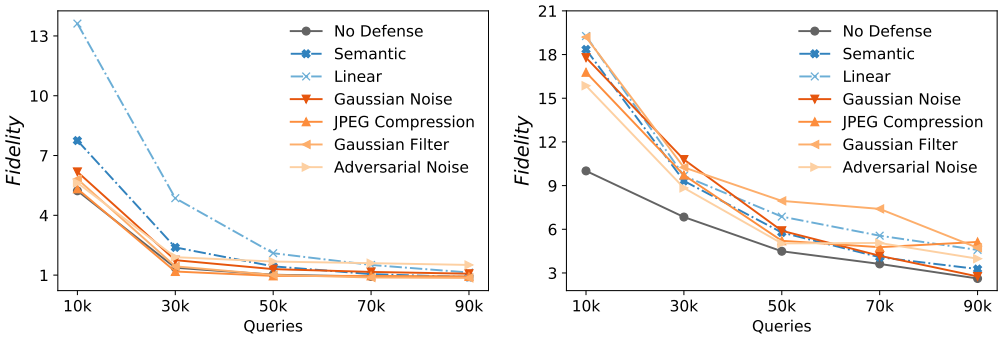
Figure 3.11: Returned images after output perturbation-based defense techniques.



(a) The performance of attack model PGGAN (b) The performance of attack model SNGAN

Figure 3.12: The performance of attack model PGGAN under various defenses.

different defenses for the black-box fidelity extraction scenario. We observe that *fidelity* values of attack models can be largely decreased with an increase in the number of queries.



(a) The performance of attack model PGGAN (b) The performance of attack model SNGAN

Figure 3.13: *Fidelity* of attack models under different defenses for black-box fidelity extraction scenarios. *Fidelity* values of attack models can be largely decreased with an increase in the number of queries.

Discussion. The reason why input perturbation-based defenses can work is at least explained from two aspects: increasing the similarity of generated samples and a distribution mismatch between latent codes produced by interpolation and drawn from prior distributions. For the former, we can see that interpolation operations increase the similarity of images from Figure 3.10. For the latter, latent codes produced by interpolation operations are different from latent codes drawn from the prior distribution that the target model was trained on. This is because latent codes produced by linear operation do not obey the prior distribution of the target model, which also brings a benefit in disguising the true data distribution [ASTG19].

Output perturbation-based defenses can work because they directly perturb these generated samples. In practice, this type of defense requires model providers to trade-off image quality and the model's security through magnitudes of changes. Although Gaussian noise defense shows the best performance, it is possible for adversaries to remove noise.

3.9 Conclusion

In this chapter, we have systematically studied the problem of model extraction attacks on generative adversarial networks, and devised, implemented, and evaluated this attack from the perspective of fidelity extraction and accuracy extraction. For fidelity extraction, extensive experimental evaluations show that adversaries can achieve an excellent performance with about 50K queries. For accuracy extraction, adversaries further improve the accuracy of attack models after obtaining additional background knowledge, such as partial real data from the training set or the discriminator of the target model. Furthermore, we have also performed a case study where the attack model which steals a state-of-the-art target model can be transferred to new domains to broaden the scope of applications based on extracted models.

These effective attacks also motivate us to design two types of defense techniques: input and output perturbation-based defense. They mitigate model extraction attacks through perturbing latent codes and generated samples, receptively. Extensive experimental evaluations show that semantic interpolation and Gaussian noise defenses achieve stable performance.

Finally, we also identify a number of directions for future work. Because GAN models generally are considered as intellectual properties of model owners, protecting GANs through verifying the ownership is an interesting direction. In addition, stealing a GAN model also means the leakage of distribution of the training set. Therefore, training with differential privacy techniques can be utilized to protect the privacy of training data of a model [ACG+16]. However, training time and stability of the training process are big challenges for GANs. For further work, we plan to design new methods based on differential privacy techniques to mitigate *accuracy* of model extraction.

Chapter 4

Ownership Protection in Generative Adversarial Networks

In the previous chapter, we systematically illustrated model extraction attacks in generative adversarial networks (GANs) and proposed several countermeasures to mitigate the attacks by sacrificing model utility. In this chapter, we shift to protecting GANs by verifying their ownership, which can be considered a deterrence against malicious adversaries and does not suffer from the loss of model utility. Specifically, we first propose a new ownership protection method based on the common characteristics of a target model and its stolen models. Extensive experimental results show that our new method can achieve the best protection performance, compared to the state-of-the-art methods. Then, we demonstrate the effectiveness of our method with respect to the number of generations of model extraction attacks, the number of generated samples, different datasets, different target models, and adaptive attacks.

4.1 Introduction

GANs, as one of the most successful generative models, have exerted revolutionary influences on many application domains, such as image synthesis [GPAM+14, MKKY18, KLA19, BDS19, SSG22a], and image editing [ZSZZ20, SZ21, XZY+22]. However, building a well-trained state-of-the-art GAN model is not straightforward. It usually requires the complicated and exhausting process of data collection, expert-level knowledge in model architecture design, elaborate hyperparameter tuning, and extensive computing resources. Therefore, a high-quality GAN model is incredibly costly and should be regarded as the intellectual property of the model owner.

As GAN models are valuable, this simultaneously incentivizes adversaries to steal these models in various ways. On the one hand, adversaries can physically steal a GAN model via malware infection or insider attacks [Sch02]. An insider attack can directly copy the target model (i.e. victim model) through those who are authorized to access the full model. As a result, the stolen model is totally the same as the target model. On the other hand, adversaries can functionally steal a GAN model via

model extraction attacks [HP21b]. This threat exists because an increasing number of technology companies provide Machine Learning as a Service (MLaaS) to their customers, such as Amazon AWS, Google Cloud, and Microsoft Azure. A model extraction attack enables adversaries to obtain a substitute model via exposed interfaces. As a consequence, the stolen model is functionally similar to the target model. In general, both physical stealing attacks and model extraction attacks seriously jeopardize the intellectual property of legitimate model owners. It is paramount to develop ownership protection methods to safeguard the intellectual property of GANs.

Despite confronting these threats, research on ownership protection for GANs is somehow much less explored. There is one work that proposes to verify ownership of GANs by watermarks [OCN+21]. On the one hand, this method needs to retrain target models to embed watermarks, which may compromise the models' generation performance. On the other hand, it relies on specific inputs (i.e. triggers) to extract watermarks. Such dependency might result in its failure to verify models from model extraction attacks.

In this chapter, we develop a new ownership protection method for GANs, which can protect ownership on both physical stealing and model extraction attacks. Our method claims the ownership of a GAN by leveraging common characteristics of a target model and its stolen models. The rationale for our method is that stolen models are derived from the target model while honest models are not. Thus, these common characteristics can be leveraged to differentiate stolen models from honest models. More specifically, we utilize generated samples from GANs to build a discriminative classifier to learn these characteristics. This is because the objective of a GAN is to learn the distribution of a training dataset and the learned implicit distribution of a GAN can be represented by these generated samples [BKP+20] (see Section 4.4.3).

We comprehensively evaluate our new method by comparing it with two state-of-the-art works: watermark-based method (abbreviation as Ong method) [OCN+21] and fingerprint-based method (abbreviation as Yu method) [YSAF21]. Here, note that fingerprint-based methods are proposed for deepfake detection and attribution [MGVP19, YDF19, WWZ+20, YSAF21, YSC+22]. In this work, *we are the first to introduce them to the ownership protection field*, considering their common objective: fingerprints can be used to infer whether a suspect sample is from the model. Extensive experiments show that our method achieves the best performance in all evaluations among these ownership protection methods. In contrast, the other two methods cannot work in many cases, especially for model extraction attacks (see Section 4.6).

Furthermore, we analyze the protection performance by visualizing the characteristics learned by our method (see Section 4.7.1). We also show the stability of our method on the number of generations of model extraction attacks, which is a new emerging threat in the generative AI domain. In contrast, the protection performance of the Yu method presents a significant linear decrease and the Ong method completely fails in providing any protection (see Section 4.7.2). Our analysis with respect to the number of generated samples and different datasets further shows the

effectiveness of our method. Finally, extensive evaluations under adaptive attacks also demonstrate that our method is still effective and robust even if adversaries obtain partial knowledge of our method (see Section 4.8).

4.2 Related Work

Generative adversarial networks. GANs have undoubtedly achieved a series of successes in image generation [MKKY18, KALL18, BDS19, KLA19, KLA+20], image manipulation [SGTZ20, SZ21], and image super-resolution [LTH+17, ZLDQ19]. The seminal GAN introduced in 2014 [GPAM+14] presents a promising result in image synthesis, which significantly inspires more and more researchers to propose various methods to further advance the performance of GANs. Karras et al. [KALL18] propose a progressive training strategy that enables a GAN to synthesize high-resolution images. In addition to generating high-resolution images, Karras et al. [KLA19] further introduce neural style transfer structures into the architecture of GANs and it empowers GANs to generate a variety of style images. Takeru et al. [MKKY18] introduce spectral normalization to normalize the weights of each layer to improve the quality of synthetic images. *Overall, each improvement in GANs requires talented researchers to devote tremendous efforts. In this chapter, instead of further improving the performance of GANs, we target at developing a technique to protect the intellectual property of valuable GANs.*

Ownership protection. There are numerous works aiming to protect ownership of discriminative models [UNSS17, ZGJ+18, LHZG19, LZK21, JCCCP21, CWP+22, DDK+22, CHZ22]. These works can be generally classified into three groups: embedding watermarks into model parameters [UNSS17], using predefined inputs as triggers [ABC+18] and utilizing unique features of models [MYP20, CWP+22]. *However, methods on discriminative models cannot be applied to generative models since they are different machine learning models.*

There are only a few works on ownership protection of generative models. Ong et al. [OCN+21] propose a protection framework for GANs by adding a novel regularization term to the existing loss function. Some works on fingerprints of GANs can be applied into this field. Yu et al. [YSAF21] propose to add fingerprints into training data and then verify the fingerprints on GANs. Additionally, Yu et al. [YSC+22] add a novel fingerprint embedding layer to modulate the generation of fingerprinted images. *However, these works do not perform evaluations on emerging model extraction attacks. In this chapter, we propose a novel protection method and thoroughly evaluate these works on various attacks, including model extraction attacks.*

4.3 Background

4.3.1 Paradigms of Ownership Protection

Current paradigms of ownership protection on GANs can be divided into two classes: watermark-based methods and fingerprint-based methods. Watermark-based methods initially are proposed in the work [OCN+21]. The key idea is that model owners embed specific inputs and outputs into a GAN in the training phase. Then, if specific outputs (e.g. watermarked generated samples) can be obtained from a suspect GAN through specific inputs (e.g. triggers), model owners claim ownership of this GAN. For example, Ong et al. propose a protection framework for GANs by adding a novel regularization term to the existing loss function [OCN+21]. For simplicity, we refer to this ownership protection method by the first author's name, i.e, Ong [OCN+21].

Fingerprint-based methods are initially proposed for deepfake detection and attribution [YSAF21, YSC+22]. *In this work, we are the first to apply these methods to protect ownership of GANs.* This is because they share a common objective that fingerprints can be used to infer whether a suspect sample is from their model. Specifically, the key idea of fingerprint-based methods is that if the fingerprint extracted from generated samples from a GAN is identical to the true fingerprint, model owners can claim ownership of the GAN. To achieve this goal, various methods are proposed, such as adding fingerprints on the training set of a GAN [YSAF21], and designing new architectures of a GAN and loss functions [YSC+22].

In this chapter, considering their excellent performance and the diversity of methods, we choose one watermark-based method—Ong [OCN+21] and one fingerprint-based method—Yu [YSAF21] to evaluate the performance in ownership protection and make comparisons with our proposed method. Note that although there are some works [MGVP19, YDF19, DTL21] about fingerprints of GANs, they focus on differentiating different types of GANs, such as PGGAN [KALL18] and SNGAN [MKKY18]. Thus, we do not compare these methods because they cannot distinguish different GANs from the same type.

4.3.2 Details of Obfuscations

In this subsection, we introduce obfuscation operations, including input perturbation, output perturbation, overwriting, and fine-tuning, which is used to evaluate the robustness of ownership protection methods in Section 4.6.3.

Input perturbation. Given a trained GAN model G, we can get a generated sample x_g from the GAN by a latent code z which is drawn from prior distribution P, i.e., $x_g = G(z), z \sim P$. Input perturbation aims to modify the queries, i.e., latent codes. The reason why we consider this is that some works verify the ownership by specific latent codes $z' = T(z)$. T is a function to transform a normal latent code z to a specific latent code z'. Therefore, an adversary can perturb latent codes to evade this type of verification. in this work, we adopt random input perturbation. Specifically, for any query, a target model resamples latent codes from Gaussian distribution.

Output perturbation. In addition to perturbing latent codes, an adversary can perturb generated samples. in this work, we consider the following four common operations.

a. *Random Noise.* This operation adds noises into a generated sample x_g. Common noises include Gaussian noise and Poisson noise. in this work, we choose Gaussian noise and the mean μ and the variance σ control the strength of noises.

b. *Filtering.* This operation aims to enhance some characteristics of an image. Common operations include mean filter, median filter and Gaussian filter. in this work, we choose the Gaussian filter.

c. *Blurring.* This operation makes a generated sample x_g less sharp by convolution. Common blurring operations include box blurring and Gaussian blurring. in this work, we choose Gaussian blurring.

d. *Compression.* This operation is to compress the size of an image without significantly degrading the image quality. Common compression operations include lossless compression JPEG and lossy compression JPG. in this work, we choose JPEG compression.

Overwriting. This attack is to target a class of ownership protection methods utilizing watermarks or fingerprints. An adversary can encode a different watermark/fingerprint to overwrite the original watermark/fingerprint. Ideally, an ownership protection method should still verify the ownership in this case. In this work, our proposed method does not rely on watermarks and fingerprints, thus intrinsically eliminating the threat of this attack.

Fine-tuning. After obtaining a substitute model, an adversary may further fine-tune the model on their own dataset. Generally, an adversary can partially or wholly fine-tune the model. Wholly fine-tuning refers that all weights of the model are fine-tuned while partially fine-tuning refers that the weights of some layers are frozen and the remaining are fine-tuned. In this work, we consider wholly fine-tuning in which we take the weights of the stolen model as initialization and retrain a GAN model on a new dataset.

4.4 A New Ownership Protection Method

Unlike these paradigms that require forcibly implanting watermarks or fingerprints into target models and retraining target models which are introduced in Section 4.3.1, our method provides a novel paradigm: the common characteristics of a target model and its stolen models are exploited to claim ownership, motivated by emerging model extraction attacks [TZJ+16, HP21b].

4.4.1 Threat Model

We assume that defenders, i.e. model providers who deploy an ownership protection method on their target model, only have access to generated samples from a suspect model deployed by the adversaries. Thus, the defenders make an ownership infringement decision only based on these generated samples. *This is the most practical and strictest assumption for defenders.*

4.4.2 Key Observations

The first key observation is that physical stealing and model extraction attacks are two fundamental but different types of ownership infringement. Physical stealing attacks refer that an adversary physically copies a model G_{sub} from the target model G_{tar}. Therefore, G_{sub} is totally the same as G_{tar}. Model extraction attacks [HP21b] refer that an adversary retrains a substitute model G_{sub} on generated samples from a target model G_{tar}. These generated samples can be obtained by an adversary when model owners release generated samples or provide a querying interface. Thus, G_{sub} is functionally similar to G_{tar}.

The second key observation is that as stolen models (i.e. constructed by physical stealing or model extraction attacks) are derived from the target model but honest models are not, it is thus natural to assume that stolen models and the target model share common characteristics which do not exist in honest models. Therefore, we can learn and leverage such characteristics as an evidence to differentiate stolen models from honest models.

4.4.3 Ownership Protection Algorithm

Overview. In order to extract the characteristics for a target model, our method proposes to learn these characteristics by training a binary classifier on generated samples. Generated samples from model extraction and physical stealing are labelled as positive while samples from honest models, i.e. independently trained models, are labelled as negative. The reason why we use generated samples is that a GAN model is to learn the distribution of a training set. The learned distribution is implicit, which can be represented through these generated samples [BKP+20]. This process is also illustrated in the deployment phase of Figure 4.1.

In practice, it is impossible for the defenders to consider all independently trained models and all models constructed by model extraction. Therefore, our method constructs positive and negative GAN models only by the limited knowledge of the defenders: the architectures of the target model and the training set. Specifically, the architectures of all models (i.e. G_{tar}, G_{sub}, G_{ind}) in the deployment phase are the same. The independent set $\mathfrak{D}_{ind}$ and the target set $\mathfrak{D}_{tar}$ are from the same distribution but disjoint. $\mathfrak{D}_{ind}$ can be easily obtained, e.g. the defenders split a dataset into two parts: one as the independent set and the other as the training set. We emphasize that our method is practical as suspect models constructed by the adversaries can be trained on any (unknown for the defenders) GAN architectures

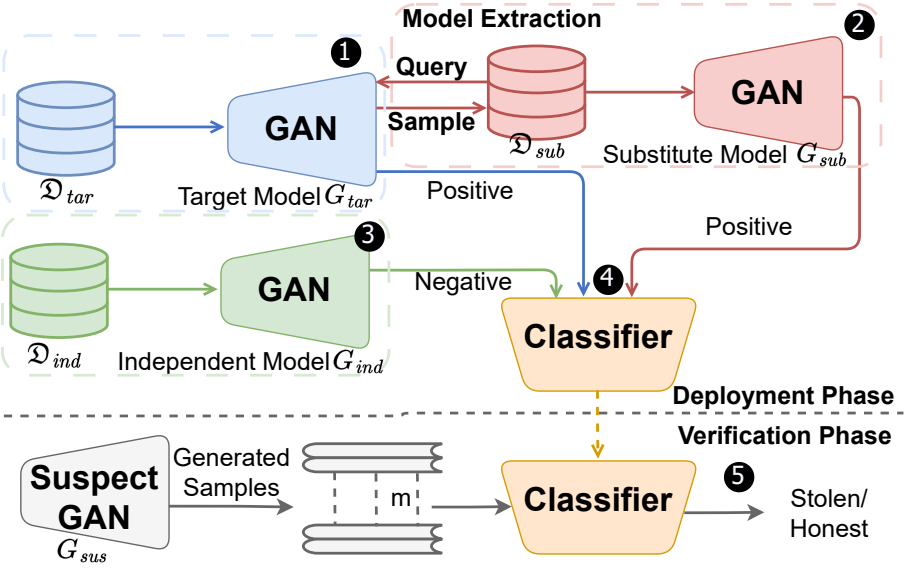


Figure 4.1: Overview of our method. ❶ A target model is trained on a dataset $\mathfrak{D}_{tar}$. ❷ A substitute model is constructed by model extraction. ❸ An independent model is trained on a dataset $\mathfrak{D}_{ind}$ that has the same distribution as the dataset $\mathfrak{D}_{tar}$, but it is disjoint with $\mathfrak{D}_{tar}$. ❹ A classifier is trained to discriminate between stolen models and honest models. ❺ The classifier is used for the verification of a suspect model. Here, the target model can also refer to the physically stealing model. The defenders do not have any information about a suspect model G_{sus}, except generated samples, in the verification phase.

and datasets. Our extensive experiments in Section 4.6.4, demonstrate our method can well generalize beyond these unknown GANs and correctly recognize them.

Details of the algorithm. As illustrated in the function *buildProtection* of Algorithm 2, specifically, given the generator of a target model G_{tar} and an independent dataset $\mathfrak{D}_{ind}$, we first construct a substitute model G_{sub} by extracting the target model G_{tar}. Next, we train a GAN G_{ind} on the independent dataset $\mathfrak{D}_{ind}$. Samples from G_{sub} and G_{tar} are labeled as positive while samples from G_{ind} are labeled as negative. These samples are used for training a classifier and in this work we choose ResNet50 [HZRS16] as the classifier.

After obtaining the trained classifier, we start to perform the verification of ownership. We first collect m generated samples released by a suspect model G_{sus}. These samples are fed into the classifier and m predictions can be obtained. We calculate the percentage of these positive predictions. If it is larger than a predefined threshold, the suspect model is inferred as stealing from the target model. We also analyze how the number of generated samples m affects our performance in Section 4.7.3. This process is also illustrated in the function *performVerification* of Algorithm 2. Note that the classifier used for ownership verification should remain confidential. It is exclusively utilized by model owners and does not provide any interface, such as querying, with users or adversaries.

Algorithm 2: The GAN-Guards Algorithm.

Input: a target model: G_{tar}; an independent dataset $\mathfrak{D}_{ind}$; m samples X_{sus} from a suspect model G_{sus}.
Output: ownership decision: *OwDecision*

def `buildProtection`($G_{tar}, \mathfrak{D}_{ind}$)**:**
  Sample $\tilde{n}$ samples X_{gen} from G_{tar};
  $G_{sub} \leftarrow \mathsf{trainGAN}(X_{gen})$;
  $G_{ind} \leftarrow \mathsf{trainGAN}(\mathfrak{D}_{ind})$;
  Sample n samples X_{gen} from G_{tar}; ▷ labelling positive for physical stealing.
  Sample n samples X_{sub} from G_{sub}; ▷ labelling positive for model extraction.
  Sample $2n$ samples X_{ind} from G_{ind}; ▷ labelling negative for the honest model.
  Classifier $\leftarrow \mathsf{trainClassifier}(X_{gen}, X_{sub}, X_{ind})$;
  return *Classifier*

def `performVerification`(*Classifier*, X_{sus}, τ)**:**
  Initialize prediction array *pred* of length m with 0;
  for $i = 0$ **to** $m - 1$ **do**
    $pred[i] \leftarrow Classifier(X_{sus}[i])$; ▷ Prediction: 1 or 0.
  $ConfiSocre = \mathsf{sum}(pred)/m$;
  ▷ Make a decision based on multiple samples.
  if $ConfiSocre > \tau$ **then** $OwDecision = 1$;
  else $OwDecision = 0$;
  return *OwDecision*

4.5 Experiments

4.5.1 Datasets

We evaluate our method on two datasets: FFHQ [KLA19] and Church [YSZ+15]. They are typically used in image generation. The FFHQ dataset is designed for human face image synthesis and includes 70,000 images. The Church dataset is from the LSUN dataset, which contains 126,277 outdoor church images.

All images are resized to 64×64. For each dataset, we randomly split the dataset into three disjoint equal parts and mark each part as the corresponding dataset name plus 'I', 'II', and 'III', respectively, such as FFHQ-I and FFHQ-II. Dataset I, i.e. $\mathfrak{D}_{tar}$, is used to train a target GAN model. Dataset II is used to train a GAN and later the model (i.e. Ind-a illustrated in Section 4.5.2) is used as negative to test the ownership protection methods. Dataset III, i.e. $\mathfrak{D}_{ind}$, is used to train a GAN model and later the model is used for building a classifier together with the target model. Specifically, we set the size of each part of FFHQ and Church as 20,000 and 40,000, respectively.

4.5.2 Suspect Models

We consider various suspect models. Positive suspect models are considered ownership infringement and these models are derived from the target models via physical stealing and model extraction, and obfuscation attacks, such as input perturbation, output perturbation, overwriting, and fine-tuning attacks. Negative suspect models are host models and they are built from independent training. Similar to

the settings of negative suspect models used in the work [CWP+22], here we also consider two types: *Ind-a* trained on dataset II with the same architectures of target models, and *Ind-b* that is trained on dataset I, with the same architectures of target models but uses different seeds, i.e. different random initializations.

Implementation details of suspect models. For positive suspect models, models from physical stealing (marked as PS) are the same as target models. We use model extraction attacks proposed in the work [HP21b] to construct models from model extraction (marked as ME). Specifically, given m generated samples from a target model G_{tar}, the adversaries retrain a model (also called the substitute model or attack model) on m generated samples. Attack models can use any architecture, such as SNGAN, PGGAN, or StyleGAN. We study protection performance under model extraction attacks with different GANs as attack models in Section 4.6.4.

For negative suspect models, Ind-a is trained with the same architectures of target models but on dataset II. Ind-b is trained with the same architectures and datasets of target models but uses different seeds, i.e. different random initializations. Using different seeds means models trained on different training environments and optimization processes. Theoretically, models trained with different seeds should be different, because they do not derive from model extraction and physical stealing, and they are honest models with independent training. Thus, an ownership protection method should be able to differentiate them. Here, setups for negative suspect models are very similar to those for target models because we aim to test whether a protection method hurt honest model providers in the strong assumption setting. This requires that a protection method should be extremely robust.

4.5.3 Metrics

We use FID [HRU+17] to measure the performance of a GAN. 50K generated samples from a GAN and all training samples are used to compute the FID value.

In terms of protection performance, the Ong method [OCN+21] utilizes the SSIM [WBSS04] score to measure the similarity between the groundtruth watermark and the watermark extracted from a suspect model. If the SSIM score of an image is higher than a threshold, the image is more likely from the target model. The Yu method [YSAF21] calculates a bitwise accuracy between the groundtruth fingerprint and an extracted fingerprint. Claiming ownership of a model based on only one image is not robust enough. Therefore, we make a final decision by computing a confidence score on multiple samples. Specifically, given m samples and each sample gets an output $o \in \{0, 1\}$ from a method, the confidence score that recognizes a suspect model as positive is computed by: $\mathit{ConfidenceScore} = \frac{\sum_{i=0}^{m-1} o_i}{m}$. In this work, we set threshold τ of all methods as 90% for consistency. Thus, a suspect model is predicted as positive (stolen model) if $\tau \geq 90\%$. We fix the number of samples m as 1,000.

Table 4.1: Performance of target model SNGAN trained on FFHQ-I on different methods. (↓ is better).

Methods	Ong	Yu	Ours
FID($\mathfrak{D}, \tilde{G}$) ↓	20.14	26.46	20.25

4.5.4 Experimental Setups

In terms of GANs, we use SNGAN [MKKY18], PGGAN [KALL18] and StyleGAN [KLA19]. These all can achieve excellent performance in image synthesis. We use the official implementation of each GAN to train GANs. For model extraction attacks, considering a trade-off between attack cost and performance, we set the number of generated samples as 50,000, which is also suggested by the work [HP21b].

For our protection method, we use ResNet50 [HZRS16] pre-trained on ImageNet [RDS+15] dataset for our classifier. The SGD optimizer with a learning rate of 0.003 is used and the number of epochs is fixed as 5. As shown in Algorithm 2, we use the Gaussian prior distribution to obtain generated samples and the number of samples n is set as 100,000. Therefore, 400,000 samples in total are used for training the classifier. For the Ong method [OCN+21] and the Yu method [YSAF21], we adopt their official implementations with suggested hyperparameters.

4.6 Evaluation

In this section, we compare our method with two state-of-the-art methods: the Ong method [OCN+21] and the Yu method [YSAF21], and both have been already discussed in Section 4.3.1. We evaluate them from various perspectives, including model utility, verification performance, robustness to obfuscations, and robustness to more model extractions. Through these evaluations, we show how prior works fail in model extraction attacks and our work can perform well on both physical stealing attacks and model extraction attacks.

4.6.1 Model Utility of Target Models

Table 4.1 shows the performance of the target model SNGAN trained on FFHQ-I with different protection methods. The FID is computed by the original training set $\mathfrak{D}$ and the protected GAN $\tilde{G}$. Overall, the watermark-based method Ong and our method achieve similar outstanding performance, while the fingerprint-based method Yu shows worse performance. This is because the Yu method needs to add fingerprints into a training set, which is at the cost of sacrificing model utility.

4.6.2 Verification Performance

Figure 4.2 presents verification performance on different ownership protection methods. The red dashed line is the threshold τ of the confidence score. A model is

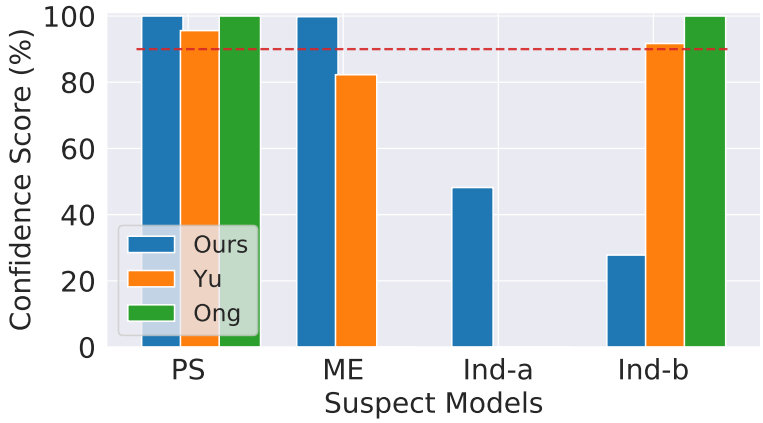


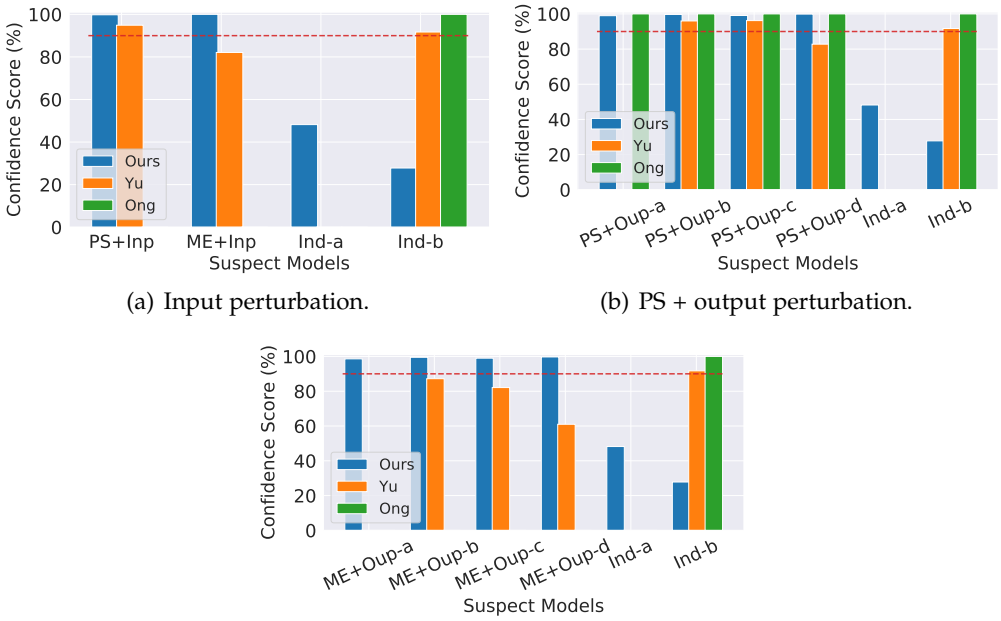
Figure 4.2: Verification performance of all methods. The target model SNGAN is trained on FFHQ-I. PS and ME are positive suspect models while Ind-a and Ind-b are negative suspect models. Note that *green and orange bars in some cases cannot be observed because their scores are 0%.*

predicted as a stolen (positive) model if $\tau \geq 90\%$. PS refers to models from physical stealing while ME refers to models from model extraction. Note that here ME, Ind-a, Ind-b models used in the verification phase are not the same models used in our deployment phase (detailed in Section 4.5.2). Overall, our method can correctly differentiate all positive and negative suspect models, achieving 100% accuracy. However, the Ong method and the Yu method are unable to defend against model extraction attacks.

Additionally, the Ong and Yu methods mistakenly recognize the suspect model Ind-b trained with different initializations as a stolen model. This is because embedded watermarks or fingerprints cannot be changed only owing to different initializations of a training process. Thus, their methods lead to false alarms and hurt honest model providers. In contrast, our method can perfectly deal with this case because our method builds on a well-trained model. Note that the setting of the suspect model Ind-b is also adopted by classification models to detect whether an ownership protection method produces false alarms [CWP+22].

4.6.3 Robustness to Obfuscations

In order to evade verification, advanced adversaries may utilize obfuscation techniques to obfuscate stolen models. In this work, we consider four types of obfuscation techniques: input perturbation, output perturbation, overwriting and fine-tuning. Input perturbation aims to modify the queries, i.e., latent codes, to evade special queries. Here, we adopt random input perturbation. That is, for any query, a target model resamples latent codes from Gaussian distribution. For brevity, we rename it Inp. Output perturbation refers to perturbing generated samples by various post-processing techniques. We use four different output perturbations: additive Gaussian noise, Gaussian filter, Gaussian blurring, and JPEG compression. We briefly rename them Oup-a, Oup-b, Oup-c, and Oup-d, respectively. The magnitude of these perturbations is set as 0.01, 0.4, 0.5, and 0.85, respectively. Overwriting refers to encoding a different watermark/fingerprint to overwrite the original watermark/fingerprint. Our method does not rely on watermarks and fingerprints,



(a) Input perturbation.

(b) PS + output perturbation.

(c) ME + output perturbation.

Figure 4.3: Robustness to Obfuscations. Protection performance for target model SNGAN trained on FFHQ-I. Again, green and orange bars in some cases cannot be observed because their scores are 0% and cannot defend against these attacks.

thus intrinsically eliminating the threat of this attack. In this work, we consider wholly fine-tuning where we take the weights of the stolen model as initialization and retrain a GAN model on a different dataset FFHQ-II. Because these obfuscation operations can be added into physical stealing (PS) or model extraction (ME), there are different combinations between obfuscation operations and PS and ME. Here, we mark them as 'PS+' and 'ME+' corresponding obfuscation operations, such as PS+Inp. Details have been illustrated in Section 4.3.2.

Results. Figure 4.3 shows the protection performance under input and output perturbation operations. Overall, our method can still remain 100% accuracy against input perturbation and output perturbation attacks. In contrast, the Ong method totally cannot resist the input perturbation attack, as shown in Figure 4.3(a) and the Yu method cannot defend against additive Gaussian noise of output perturbation attacks (PS+Oup-a and ME+Oup-a). Again, Figure 4.3(c) shows that the Ong and Yu methods cannot defend against ME+Output perturbation. We analyze the reasons for the Ong and Yu methods in Section 4.6.5.

We perform the evaluation under the overwriting attack. Figure 4.5 shows a new watermark that is used for overwriting attacks for the Ong method. The original watermark is shown in Figure 4.6 (a). We do not report results for our method because our method does not rely on watermarks or fingerprints. We summarize the results in Table 4.2. Overall, the Ong and Yu methods cannot defend against this type of attack and both confidence scores are 0%. It indicates that the overwritten watermarks and fingerprints make their methods unable to extract the expected outputs.

We evaluate the protection performance under the fine-tuning attack. We adopt wholly fine-tuning where all weights of each model are fine-tuned on FFHQ-II.

Table 4.2: Protection performance on overwriting. The target model SNGAN is trained on FFHQ-I. The suspect model PS is the model obtained by physical stealing. Confi.: confidence; Pred.: prediction.

Types	Ong		Yu	
	Confi. Score(%)	Pred.	Confi. Score(%)	Pred.
PS	0.00	0	0.00	0

Table 4.3: Protection performance on fine-tuning. Target model SNGAN is trained on FFHQ-I.

Types	Ong		Yu		Ours	
	Confi. Score(%)	Pred.	Confi. Score(%)	Pred.	Confi. Score(%)	Pred.
PS	0.00	0	0.00	0	40.10	0
ME	0.00	0	0.00	0	28.00	0

Specifically, we take the weights of the stolen model as initialization and retrain a GAN model on FFHQ-II. Table 4.3 reports protection performance under the fine-tuning attack. We observe that all protection methods cannot be robust against this attack. We analyze that this is because the fine-tuned GAN model has learned a different distribution in a new training set and neural networks suffer from catastrophic forgetting [Fre99, GMX+13]. The former makes that our method recognizes this model as an independent training model while the latter makes that the Ong and Yu methods forget embedded watermarks and fingerprints. This also inspires us to think about the ownership boundary of a GAN and develop more powerful protection work in future.

4.6.4 Robustness to More Model Extraction

When mounting model extraction attacks, adversaries can utilize various architectures of GANs to extract a target model. Figure 4.4 shows the results of all methods in terms of robustness to model extraction attacks with different GANs as attack models. The target model is SNGAN. We see that our method still performs well, while the other two methods recognize them as honest models. *This shows our method can recognize models constructed by model extraction attacks regardless of the GAN architectures of adversaries.*

4.6.5 Understanding for Different Methods

Figure 4.6 shows SSIM scores of watermarks under different output perturbations. Specifically, Figure 4.6 (a) is the watermark used in the Ong method. Figure 4.6 (b) - Figure 4.6 (e) show the watermark under different output perturbations. We can observe that only if the Ong method can extract watermarks, output perturbation attacks do not have a significant impact on the final decision. This can explain the reason why the Ong Method on PS+output perturbation can perform well, as shown in Figure 4.3(b).

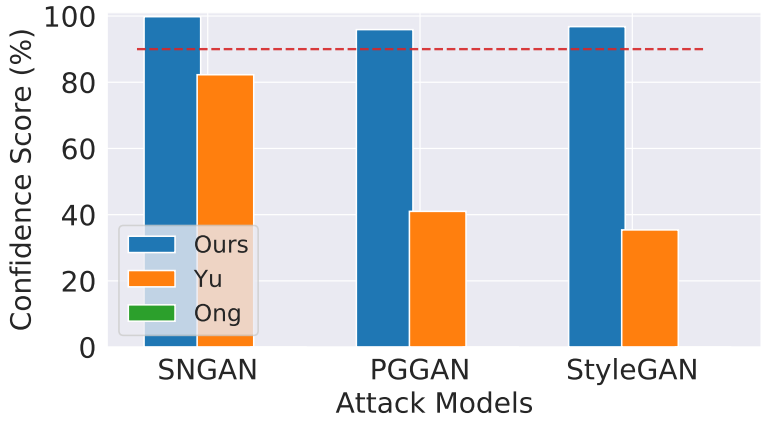


Figure 4.4: Protection performance under model extraction attacks with different GANs as attack models. The target model SNGAN is trained on FFHQ-I.

Figure 4.5: Watermarks used for overwriting attacks.

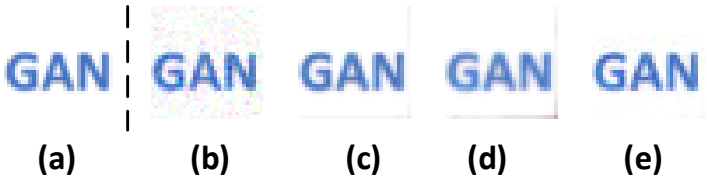


Figure 4.6: Watermarks under different output perturbations. (a) is the original watermark. From (b) to (e), the output perturbation operations are Additive Gaussian Noise, Gaussian Filtering, Gaussian Blurring, and JPEG Compression, respectively. The corresponding SSIM scores between (a) and each output perturbation are 84.085%, 97.47%, 99.34%, 95.43%, respectively.

However, model extraction attacks and their derivative attacks (i.e. ME+obfuscations) make the Ong and Yu methods fail. This is because these attacks severely undermine watermarks and fingerprints. It indicates that methods based on watermarks and fingerprints are not robust to model extraction attacks and are too easily perturbed.

4.6.6 Performance of Suspect Models

In this subsection, we summarize the performance of suspect models used in Section 4.6. Overall, we choose suspect models with the best performance in each setting. For positive suspect models, that is these models which are obtained by physical stealing or model extraction attacks, we use FID($p_{\tilde{g}}, p_r$) and FID($p_{\tilde{g}}, p_g$) to demonstrate the performance. Here, $p_{\tilde{g}}$ is the implicit distribution of a suspect model or an attack model. p_r is the implicit distribution of a training set. p_g is the

Table 4.4: Performance of suspect positive models. The target model is SNGAN trained on FFHQ-I. It is corresponding to Figure 4.2.

Types	Ong		Yu		Ours	
	Attack Performance		Attack Performance		Attack Performance	
	FID($p_{\tilde{g}}, p_r$)	FID($p_{\tilde{g}}, p_g$)	FID($p_{\tilde{g}}, p_r$)	FID($p_{\tilde{g}}, p_g$)	FID($p_{\tilde{g}}, p_r$)	FID($p_{\tilde{g}}, p_g$)
PS	20.14	0.00	26.46	0.00	20.25	0.00
ME	25.47	2.52	30.83	3.23	27.60	3.13

Table 4.5: Performance of suspect negative models. The target model is SNGAN trained on FFHQ-I. FID(p_r, p_g) is used for evaluation.

Types	Ong	Yu	Ours
Ind-a	17.84	17.84	17.84
Ind-b	20.63	29.13	17.15

implicit distribution of a target model. FID($p_{\tilde{g}}, p_g$) represents the similarity between an attack model and a target model, while FID($p_{\tilde{g}}, p_r$) represents the similarity between an attack model and the training set of a target model. In the work [HP21b], they are also called fidelity and accuracy, respectively. Here, we explicitly use FID($p_{\tilde{g}}, p_r$) and FID($p_{\tilde{g}}, p_g$) to report the performance of suspect positive models. For negative suspect models, we use FID(p_r, p_g) for evaluation.

Table 4.4 and Table 4.5 show positive and negative suspect models for target model SNGAN trained on FFHQ. Table 4.6 shows the performance of suspect positive models constructed by model extraction with different GANs. Overall, all suspect models show excellent performance.

4.7 Analysis

In this section, we intensively examine the protection performance of our method in terms of the learned characteristics, the number of generations of model extraction attacks, and the number of generated samples. We also show its protection performance on different datasets and target models.

Table 4.6: Performance of suspect positive models. The target model is SNGAN trained on FFHQ-I. It is corresponding to Figure 4.4.

Attack Models	Ong		Yu		Ours	
	Attack Performance		Attack Performance		Attack Performance	
	FID($p_{\tilde{g}}, p_r$)	FID($p_{\tilde{g}}, p_g$)	FID($p_{\tilde{g}}, p_r$)	FID($p_{\tilde{g}}, p_g$)	FID($p_{\tilde{g}}, p_r$)	FID($p_{\tilde{g}}, p_g$)
SNGAN	25.47	2.52	30.83	3.23	27.60	3.13
PGGAN	23.58	2.33	29.29	1.80	23.11	2.56
StyleGAN	21.62	2.70	27.27	2.62	21.07	2.97

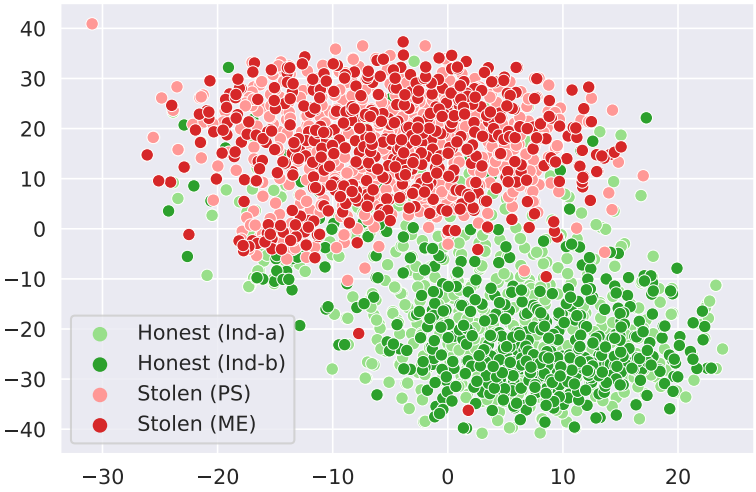


Figure 4.7: T-SNE visualization of characteristics learned by our method for stolen models and honest models.

4.7.1 Visualization of Characteristics

Figure 4.7 shows the T-SNE visualization of characteristics learned by our method. We plot the T-SNE figure by using outputs from the penultimate layer of the classifier and the dimension of the outputs is 2,048. We clearly see that characteristics from stolen models including PS and ME are entangled together and have a clear boundary with that from honest models.

4.7.2 Generations of Model Extraction Attacks

Theoretically, model extraction attacks on GANs can continue forever like the process of biological heredity, as shown in Figure 4.8. Models produced during this process, such as $G^{(i)}$, should be correctly identified by ownership protection methods. This motivates us to investigate whether the protection performance will decrease with the number of generations of model extraction attacks. We emphasize this is our newly identified threat, which is not discussed in the literature about GAN ownership protection. Moreover, this threat will become more common considering the popularity of generative AI.

Here, we fix the number of generated samples as 1,000 and the target model is SNGAN trained on FFHQ-I. We mark the target model SNGAN as SNGAN$^{(0)}$, and the first generation of model extraction is marked as SNGAN$^{(1)}$, which means an adversary uses an attack model SNGAN to extract the target model SNGAN. We do not show the performance of the Ong method because it cannot defend against model extraction attacks.

As shown in Figure 4.9, we can clearly observe that with the increase in the number of generations of model extraction, the Yu method becomes less and less confident. It also indicates that the fingerprint is not robust and more and more generated samples cannot extract the corresponding fingerprint. In contrast, our method still remains almost 100% confident to verify ownership of the target model.

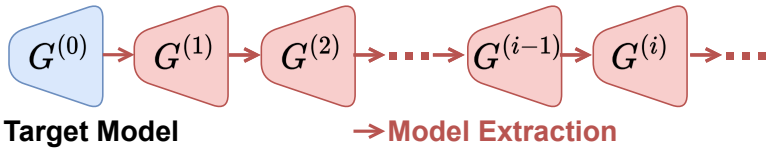


Figure 4.8: Generations of model extraction attacks.

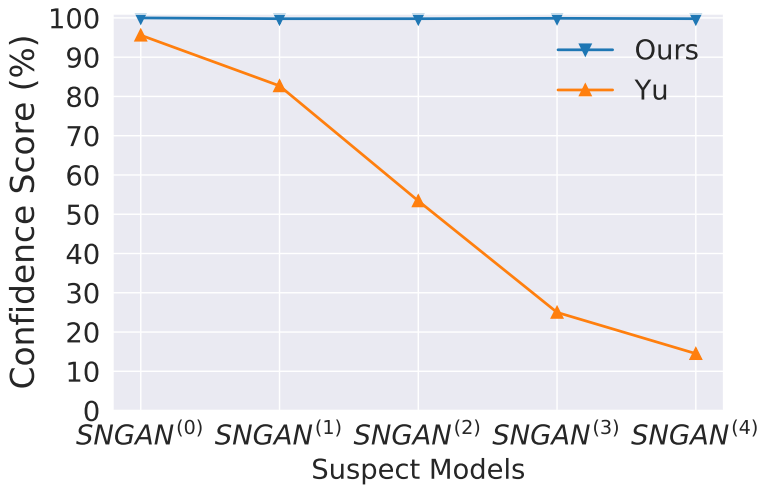


Figure 4.9: Protection performance with regard to the number of generations of model extraction attacks.

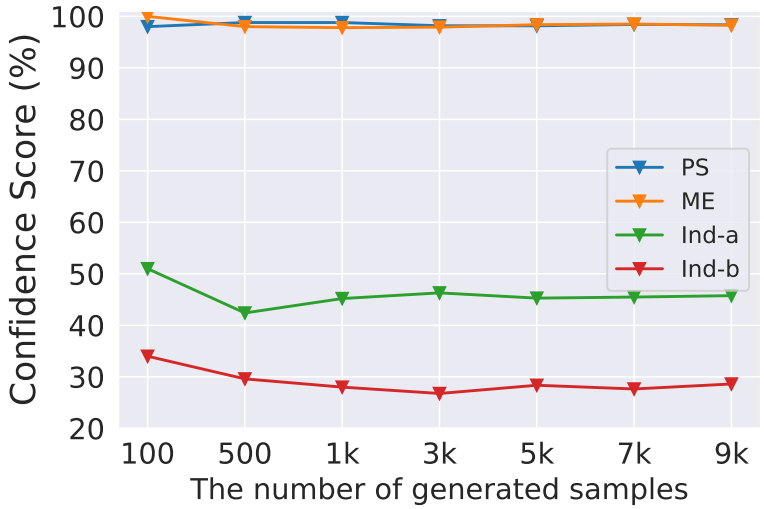
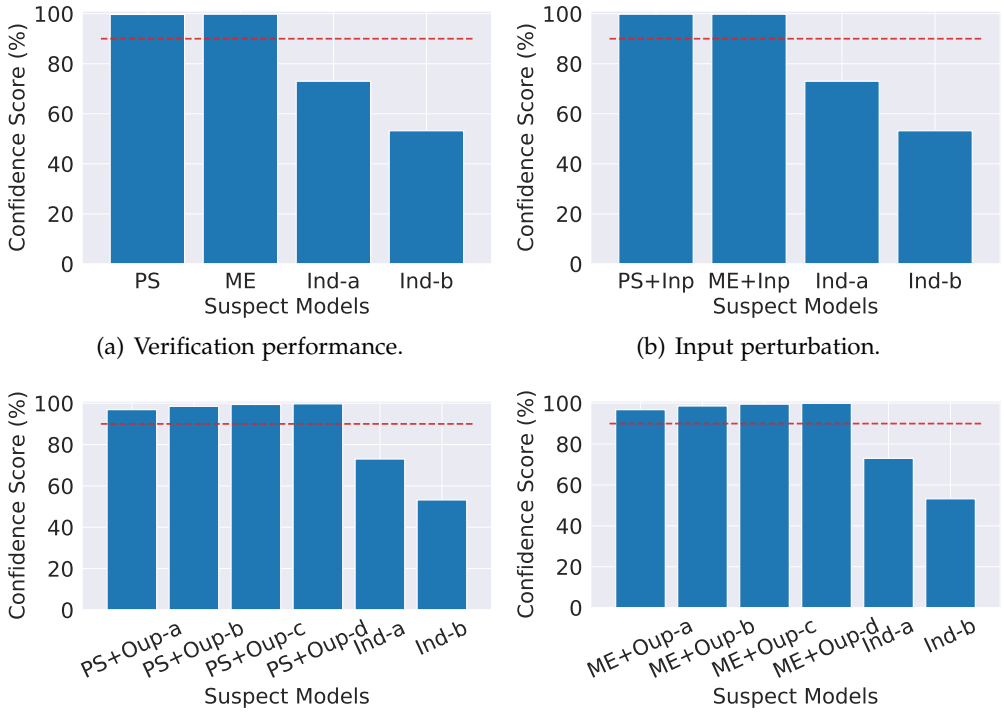


Figure 4.10: Protection performance with regard to the numbers of generated samples.

4.7.3 Number of Generated Samples

Figure 4.10 presents the protection performance of our method under the different numbers of generated samples. The target model SNGAN is trained on FFHQ-I. The ground truth of PS and ME is positive while that of Ind-a and Ind-b is negative. We can clearly see that the confidence scores gradually remain stable after 1,000 generated samples on all suspect models. It also shows that our protection method has advantages with respect to the *efficiency*, i.e. it requires as few as 1,000 samples.



(a) Verification performance.

(b) Input perturbation.

(c) PS + output perturbation.

(d) ME + output perturbation.

Figure 4.11: Protection performance on target model SNGAN trained on Church-I.

4.7.4 Different Datasets

We now present the performance of our method on the Church dataset which is widely used in scene synthesis. The detail of this dataset is also discussed in Section 4.5.1. The target model is SNGAN trained on the Church-I dataset and achieves 12.96 FID.

Overall, our method on the Church dataset can achieve the same exceptional protection performance as that on the FFHQ dataset. Figure 4.11(a) shows verification performance on our method. We can clearly observe that our method can verify the positive suspect models as positive with high confidence. As shown in Figure 4.11(b), Figure 4.11(c) and Figure 4.11(d), our method can still remain 100% accuracy under the input perturbation and output perturbation. In terms of the protection performance of our method on fine-tuning attacks, we take the weights of the stolen model as initialization and retrain a GAN model on Church-II. Our evaluation shows that it fails to recognize positive suspect models as positive.

Figure 4.12 shows the results of our method in terms of robustness to different model extractions. We observe that our method still performs perfectly regardless of the types of attack models.

4.7.5 Different Target Models

We also show the protection performance of our method on the target model StyleGAN. The StyleGAN is trained on FFHQ-I and achieves 8.76 FID.

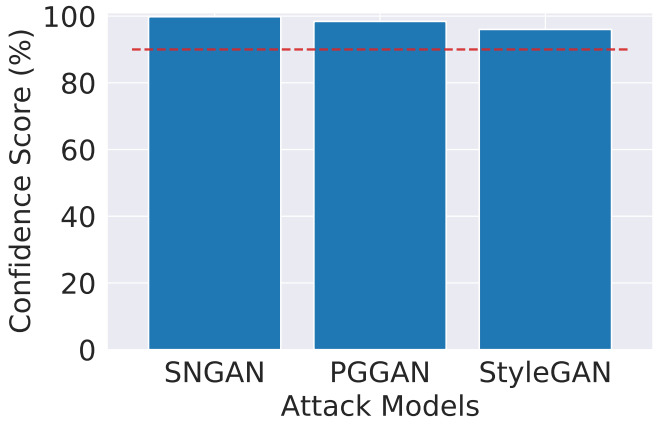
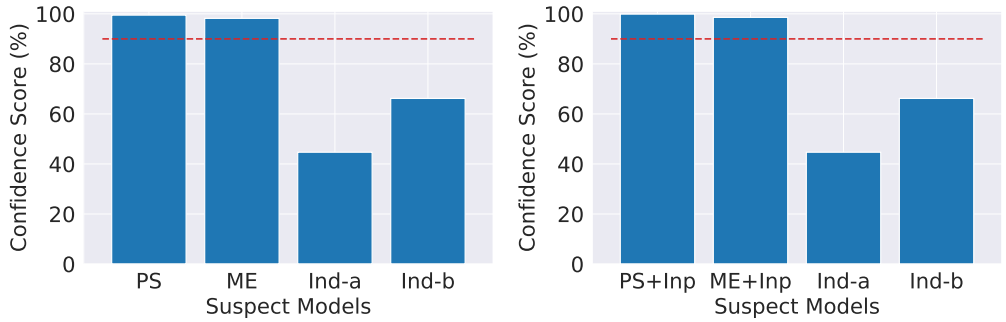
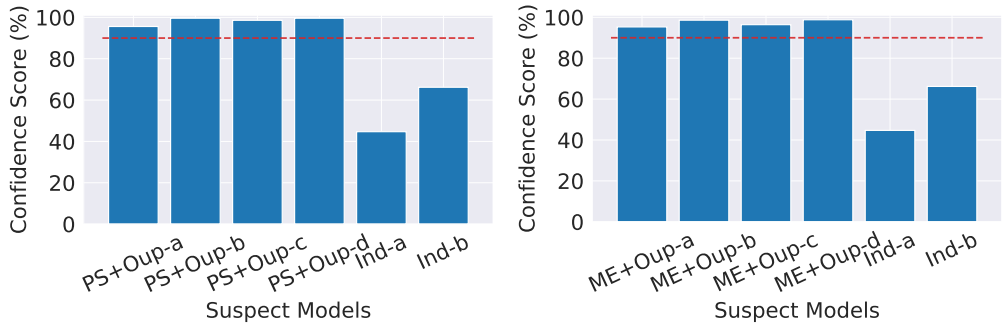


Figure 4.12: Robustness to more model extraction attacks. Protection performance under model extraction attacks with different GANs as attack models. Target model SNGAN is trained on Church-I.



(a) Verification performance.

(b) Input perturbation.



(c) PS + output perturbation.

(d) ME + output perturbation.

Figure 4.13: Protection performance on target model StyleGAN trained on FFHQ-I.

Overall, our method still has competitive protection performance on the target model StyleGAN. Our method can achieve 100% accuracy on verification performance for all suspect models, as shown in Figure 4.13(a). In terms of obfuscations, 100% accuracy can be seen on input perturbation attacks and four types of output perturbation attacks, as depicted in Figure 4.13(b), Figure 4.13(c) and Figure 4.13(d) respectively. In the face of fine-tuning attacks, our method still fails to recognize these positive suspect models. Figure 4.14 shows the results of our method in terms of robustness to different model extractions. We observe that our method still performs perfectly regardless of the types of attack models.

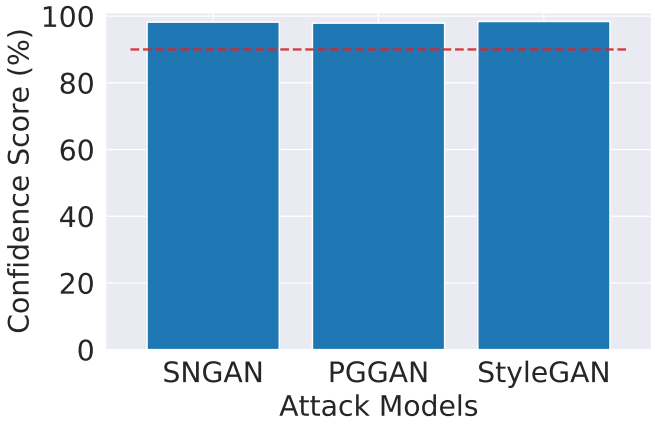


Figure 4.14: Robustness to more model extraction. Protection performance under model extraction with different GANs as attack models. Target model StyleGAN is trained on FFHQ-I.

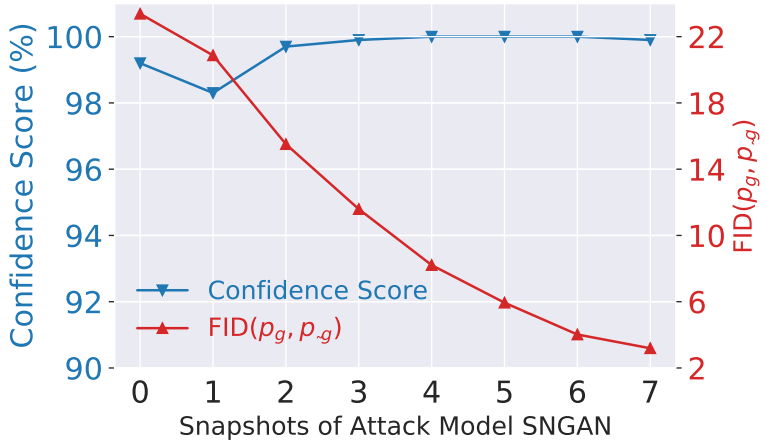


Figure 4.15: Protection performance under the adaptive attack I. The target model SNGAN is trained on FFHQ-I.

4.8 Adaptive Attacks

Although researchers studying defense techniques strongly advocates that a new defense should be evaluated on adaptive attacks [TCBM20], prior works on ownership protection on GANs do not adopt it. In this work, we present the performance of our method under adaptive attacks. That is, we assume that adversaries have some knowledge of our protection method, and design a series of specific attacks to evade our method.

We discuss two types of adaptive attack scenarios. The main design principle is that we assume that adversaries perceive our method which is based on the common characteristics of a target model and its stolen models. Therefore, the adversaries attempt to decrease the confidence score of our method by sacrificing model utility (i.e. the quality of generated images). Specifically, in adaptive attack I, adversaries choose an inferior performance GAN from multiple snapshots of a GAN when mounting model extraction attacks. In adaptive attack II, adversaries evade our verification by designing a series of output perturbations by choosing the magnitude of the perturbation.

Table 4.7: Magnitudes of output perturbation of the adaptive attack II. a: Additive Gaussian Noise; b: Gaussian Filtering; c: Gaussian Blurring; d: JPEG Compression.

Strategy	Output Perturbation			
	a	b	c	d
I	0.001	0.1	0.1	95
II	0.005	0.2	0.3	90
III	0.01	0.4	0.5	85

4.8.1 Results on Adaptive Attack I

Figure 4.15 shows protection performance under the adaptive attack I. Here, we choose an attack model SNGAN to extract the target model SNGAN trained on FFHQ-I. We choose eight snapshots of SNGAN during model extraction attacks. The performance of the attack model SNGAN, i.e. FID(p_g, $p_{\tilde{g}}$), ranges from 22 to 2, as depicted in the red line. p_g and $p_{\tilde{g}}$ are the implicit distribution of the target model and the attack model, respectively. We can observe that confidence scores begin to decrease, then increase and remain at 100%, with the decrease in FID of the attack model SNGAN. In particular, the confidence scores of all snapshots are above 98%, which indicates that our method can correctly recognize all snapshots as stolen models.

4.8.2 Results on Adaptive Attack II

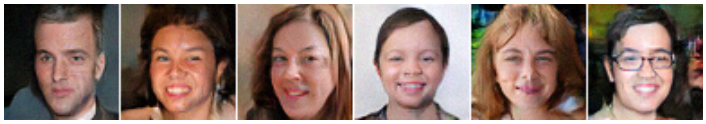
Considering the model utility, we design three strategies (strategy I, strategy II and strategy III) based on different magnitudes of four types of output perturbation. Table 4.7 summarizes the magnitudes of output perturbation of each strategy. Note that we combine four types of output perturbation instead of single output perturbation. Figure 4.16 shows visually present images generated by each strategy and the quality of generated images becomes much noisier and blurrier from strategy I to strategy III. Generally, adversaries need to trade off model utility, i.e. the quality of generated images, and copyright infringement risks.

Figure 4.17 shows protection performance under the adaptive attack II. Overall, we can observe in Figure 4.17 that our method still can recognize all positive suspect models, although the confidence score of each suspect model decreases from strategy I to strategy III. In addition, although strategy III can lower the confidence of our method, the model almost cannot be used due to the low quality of generated images visually. In practice, the loss in model utility can make the adversaries less competitive in market share, compared to legitimate model owners.

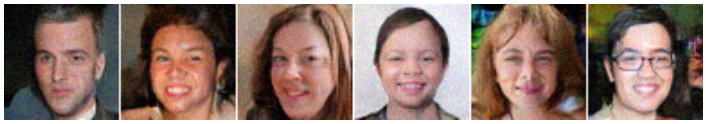
4.9 Conclusion

In this chapter, we have proposed a novel method to protect GAN ownership by leveraging the common characteristics of a target model and its stolen GANs. Extensive experimental evaluations demonstrate that: (a) In terms of model utility,

(a) No perturbation.



(b) Strategy I.



(c) Strategy II.

(d) Strategy III.

Figure 4.16: Adaptive attack II. From (a) strategy I to (c) strategy III, the magnitude of output perturbation gradually increases. The corresponding magnitudes are shown in Table 4.7. The average SSIM score for strategy I, strategy II, and strategy III is 92.20%, 82.97%, and 82.10%, respectively.

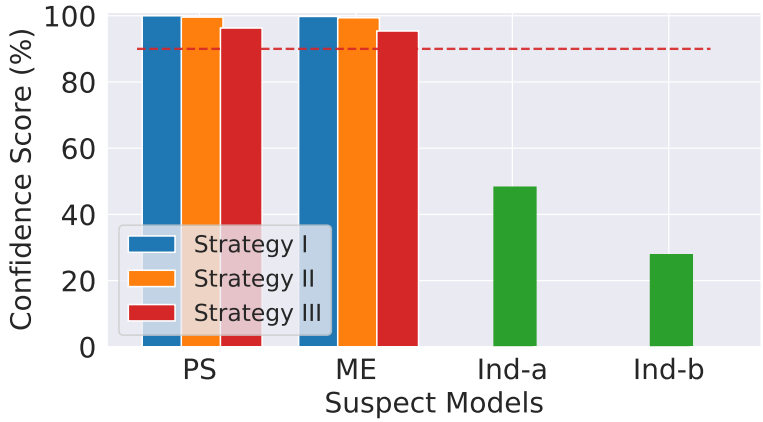


Figure 4.17: Protection performance under the adaptive attack II. The target model SNGAN is trained on FFHQ-I.

our method can bring lossless fidelity, compared to models without protection, because it does not modify well-trained target models. (b) In terms of robustness, our method can achieve new state-of-the-art protection performance, compared with watermark-based methods and fingerprint-based methods. Furthermore, we have also shown that our method is still effective under two types of carefully designed adaptive attacks. (c) In terms of undetectability, our method is undetectable for adversaries because it builds on a target model with normal training and does not rely on watermarks or fingerprints. (d) In terms of efficiency, our method requires about 1,000 generated samples to confidently verify the ownership of a GAN. Finally, we also have performed a fine-grained analysis of our method from various aspects, such as visualizing learned characteristics, the stability of the performance

with regard to the number of generations of model extraction attacks, the number of generated samples, different datasets, and different target models.

Fine-tuning attacks remain a challenge for ownership protection on GANs. In future, we plan to design more powerful methods to defend against these types of attacks. In addition, applying our protection method to other domains, such as table data synthesis and text generation, is also an interesting research direction.

Part II

Data Privacy in Generative Models

Chapter 5

Membership Inference in Diffusion Models

In addition to the privacy risks of the model itself elaborated in the first part of this dissertation, how about the training set privacy of a generative model? In this chapter, we present a comprehensive study about membership inference attacks against the latest generative model - diffusion models, which aims to infer whether a sample was used to train the model. Two attack methods are proposed, namely loss-based and likelihood-based attacks. Then our attack methods are evaluated on several state-of-the-art diffusion models, over different datasets in relation to privacy-sensitive data. Furthermore, we exhaustively investigate various factors which can affect membership inference. Finally, we evaluate the membership risks of diffusion models trained with differential privacy.

5.1 Introduction

Diffusion models [SDWMG15] have recently made remarkable progress in image synthesis [HJA20, SSDK+21, KAAL22], even being able to generate better-quality images than generative adversarial networks (GANs) [GPAM+14] in some situations [DN21]. They have also been applied to sensitive personal data, such as the human face [SE19, KAAL22] or medical images [PTD+22, KAH+22], which might unwittingly lead to the leakage of training data. As a consequence, it is paramount to study privacy breaches in diffusion models.

Membership inference (MI) attacks aim to infer whether a given sample was used to train the model [SSSS17]. In practice, they are widely applied to analyze the privacy risks of a machine learning model [SM20, MS20]. To date, a growing number of studies concentrate on classification models [SSSS17, SZH+19, CCN+22, YMMS22, LZBZ22], GANs [HMDDC19, CYZF20], text-to-image generative models [WYL+22], and language models [CLE+19, CTW+21]. However, there is still a lack of work on MI attacks against diffusion models. In addition, data protection regulations, such as GDPR [PotEU16], require that it is mandatory to assess privacy threats of technologies when they involve sensitive data. Therefore, all of these drive us to investigate the membership vulnerability of diffusion models.

In this chapter, we systematically study the problem of membership inference of diffusion models. Specifically, we consider two threat models: in threat model I, adversaries are allowed to obtain the target diffusion model, and adversaries also can calculate the loss values of a sample through the model. This scenario might occur when institutions share a generative model with their collaborators to avoid directly sharing original data [PMG+18, LJW+20]. We emphasize that obtaining losses of a model is realistic because it is widely adopted in studying MI attacks on classification models [SSSS17, CCN+22, YMMS22, LZBZ22]. In threat model II, adversaries can obtain the likelihood value of a sample from a diffusion model. Providing the exact likelihood value of any sample is one of the advantages of diffusion models [SSDK+21]. Thus, here we aim to study whether the likelihood value of a sample can be considered as a clue to infer membership. Based on both threat models, two types of attack methods are developed respectively: loss-based attack and likelihood-based attack. They are detailed in Section 5.2.

We evaluate our methods on four state-of-the-art diffusion models: DDPM [HJA20], SMLD [SE19], VPSDE [SSDK+21] and VESDE [SSDK+21]. We use two privacy-sensitive datasets: a human face dataset FFHQ [KLA19] and a diabetic retinopathy dataset DRD [Kag15]. Extensive experimental evaluations show that our methods can achieve excellent attack performance, and provide novel insights into membership vulnerabilities in diffusion models (see Section 5.4). For instance, the loss-based attack demonstrates that different diffusion steps of a diffusion model have significantly different privacy risks, and there exist high-risk regions which lead to leakage of training samples. The likelihood-based attack shows that the likelihood values of samples from a diffusion model provide a strong indication to infer training samples. We also analyze attack performance with respect to various factors in Section 5.5. For example, we find that the high-risk regions still exist with the increase in the number of training samples (see Figure 5.6). This indicates that it is urgent to redesign the current noise mechanisms used by almost all diffusion models. Finally, we evaluate our attack performance on a classical defense - differential privacy [Dwo08] (see Section 5.6). Specifically, we train target models using differentially-private stochastic gradient descent (DP-SGD) [ACG+16]. Extensive evaluations show that although the performance of both types of attack can be alleviated on models trained with DP-SGD, they sacrifice too much model utility, which also gives a new research direction for the future.

In the end, we want to emphasize that although we study membership inference from the perspective of attackers, our proposed methods can directly be applied to audit the privacy risks of diffusion models when model providers need to evaluate the privacy risks of their models.

5.2 Methodology

The objective of MI attacks is to infer if a sample was used to train a model. This offers model providers methods to evaluate the information leakage of models. In this section, we first introduce threat models and then present our MI methods.

5.2.1 Threat Models

Threat Model I. In this setting, we assume adversaries can only obtain the target model, i.e. the victim diffusion model. This setting is realistic because institutions might share generative models with their collaborators instead of directly utilizing original data, considering privacy threats or data regulations [PMG+18, LJW+20]. We emphasize that adversaries do not gain any knowledge of the training set. Obtaining the target model indicates that adversaries can get the loss values through the model, and this is realistic because most MI attacks on classification models also assume adversaries can get loss values [SSSS17, CCN+22, YMMS22, LZBZ22]. Under this threat model, we propose a loss-based MI attack.

Threat Model II. In this setting, we assume adversaries can have access to the likelihood values of samples from a diffusion model. Diffusion models have advantages in providing the exact likelihood value of any sample [SSDK+21]. Here we aim to study whether the likelihood values of samples can be utilized as a signal to perform membership inference. Under this threat model, we propose a likelihood-based MI attack.

5.2.2 Intuition

We propose MI attacks based on the following two intuitions.

Intuition I. As introduced in Section 2.1.2, a diffusion model aims to minimize the loss values over the training set in the training phase. One intuition is that member samples, i.e. the training samples, should have smaller loss values, compared to nonmember samples. This is because training/member samples involve the training process and their loss values could be minimized.

Intuition II. A diffusion model is a generative model that learns the distribution of a training set. Therefore, the likelihood values of training/member samples should be higher than these of nonmember samples. This is because training/member samples are from the distribution of the training set.

5.2.3 Attack Methods

Problem Formulation. Given a target diffusion model G_{tar}, the aim of MI attacks is to infer whether a sample x from a target dataset X_{tar} is used to train the G_{tar}.

Loss-based Attack. For threat model I and following intuition I, we develop a loss-based attack. As illustrated in Section 2.1.2, diffusion models can add an infinite or finite number of noise distributions, which correspond to continuous or discrete SDE, respectively. Therefore, we can calculate the loss value of a sample at each diffusion step t. Specifically, based on Equation 2.9, the loss of a sample x at t diffusion step of DDPM is calculated by:

$$L = \frac{1}{m} \sum ||\varepsilon - \varepsilon_{\theta^\star}(\sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\varepsilon, t)||^2, \tag{5.1}$$

where m is the dimension of x and $\varepsilon_{\theta^\star}(.)$ is the trained network. By Equation 2.10, the loss of a sample x at t diffusion step of SMLD is calculated by:

$$L = \frac{1}{m}\sum \lambda(\sigma_t)||s_{\theta^\star}(x_t, \sigma_t) - \nabla_{x_t}\log q(x_t|x)||^2, \tag{5.2}$$

where $s_{\theta^\star}(.)$ is the trained network. Based on Equation 2.12, the loss of a sample x at t diffusion step of VPSDE and VESDE is:

$$L = \frac{1}{m}\sum \lambda(t)||s_{\theta^\star}(x_t, t) - \nabla_{x_t}\log q(x_t|x)||^2. \tag{5.3}$$

Then, we make a membership inference directly based on the loss value of a sample at one diffusion step. Namely, if a sample's loss value is less than certain thresholds, this sample is marked as a member sample. For one sample, we can get T or infinite losses, depending on continuous or discrete SDEs. In this work, in order to thoroughly demonstrate the performance of our attack, we compute losses of all diffusion steps T for the discrete case. We randomly select T diffusion steps for the continuous case although it has infinite steps.

Likelihood-based Attack. For threat model II and following intuition II, we propose to utilize the likelihood value of a sample to infer membership. We compute the log-likelihood of a sample x based on the following equation proposed by [SSDK+21].

$$\log p(x) = \log p_T(x_T) - \int_0^T \nabla \cdot \tilde{f}_{\theta^\star}(x_t, t)dt, \tag{5.4}$$

where $\nabla \cdot \tilde{f}_{\theta^\star}(x, t)$ is estimated by the Skilling-Hutchinson trace estimator [GCB+18]. If the log-likelihood value of a sample is higher than certain thresholds, this sample is predicted as a member sample. As introduced in Section 4.3, the work SSDE [SSDK+21] is a unified framework. In other words, DDPM, SMLD, VPSDE and VESDE can be described by Equation 2.11. Therefore, Equation 5.4 can be applied to these models to estimate the likelihood of one sample. In this work, we compute the likelihood values of all training samples.

5.3 Experiments

5.3.1 Datasets

We use two different datasets to evaluate our attack methods. They cover the human face and medical images, which are all considered privacy-sensitive data.

FFHQ. The Flickr-Faces-HQ dataset (FFHQ) [KLA19] is a new dataset that contains 70,000 high-quality human face images. In this work, we randomly choose 1,000 images to train target models. We also explore the effect of the size of the training set in Section 5.5.1.

DRD. The Diabetic Retinopathy dataset (DRD) [Kag15] contains 88,703 retina images. In this work, we only consider images that have diabetic retinopathy, which

is a total of 23,359 images. Furthermore, we randomly choose 1,000 images to train target models. Note that images in all datasets are resized to 64×64 just for the purpose of computation efficiency.

5.3.2 Metrics

Evaluation metrics for diffusion models. We use the popular Fréchet Inception Distance (FID) metric to evaluate the performance of a diffusion model [HRU+17]. A lower FID score is better, which implies that the generated samples are more realistic and diverse. Considering the efficiency of sampling, in our work the FID score is computed with all training samples and 1,000 generated samples.

Evaluation metrics for MI attacks. We primarily use the full log-scale receiver operating characteristic (ROC) curve to evaluate the performance of our attack methods, because it can better characterize the worst-case privacy threats of a victim model [CCN+22]. We also report the true-positive rate (TPR) at the false-positive rate (FPR) as it can give a quick evaluation. We use average-case metrics — accuracy as a reference, although it cannot assess the worst-case privacy.

5.3.3 Experimental Setups

In terms of target models, we use open source codes [Son21] to train diffusion models, and their recommended hyperparameters about training and sampling are adopted. More specifically, the number of training steps for all models is fixed at 500,000. For discrete SDEs, T is fixed as 1,000 while T is set as 1 for continuous SDEs. In terms of our attack methods, we evaluate the attack performance using all training samples as member samples and equal numbers of nonmember samples.

5.4 Evaluation

In this section, we first present the performance of target models. Then, we show the performance of our two attacks: loss-based and likelihood-based attacks.

5.4.1 Performance of Target Models

Considering their excellent performance in image generation, we choose DDPM [HJA20], SMLD [SE19], VPSDE [SSDK+21] and VESDE [SSDK+21] as our target models. These diffusion models are detailed as preliminaries in Section 2.1.2. They are trained on the FFHQ dataset containing 1k samples. Target models with the best FID during the training progress are selected to be attacked. Table 5.1 shows the performance of the target models. Figure 5.1 shows the qualitative results for these target models. Overall, all target models can synthesize high-quality and realistic images.

(a) DDPM (b) SMLD (c) VPSDE (d) VESDE

Figure 5.1: Generated images from different target models trained on FFHQ.

Table 5.1: The performance of target models on FFHQ.

Target Models	DDPM	SMLD	VPSDE	VESDE
FID	57.88	92.81	20.27	63.37

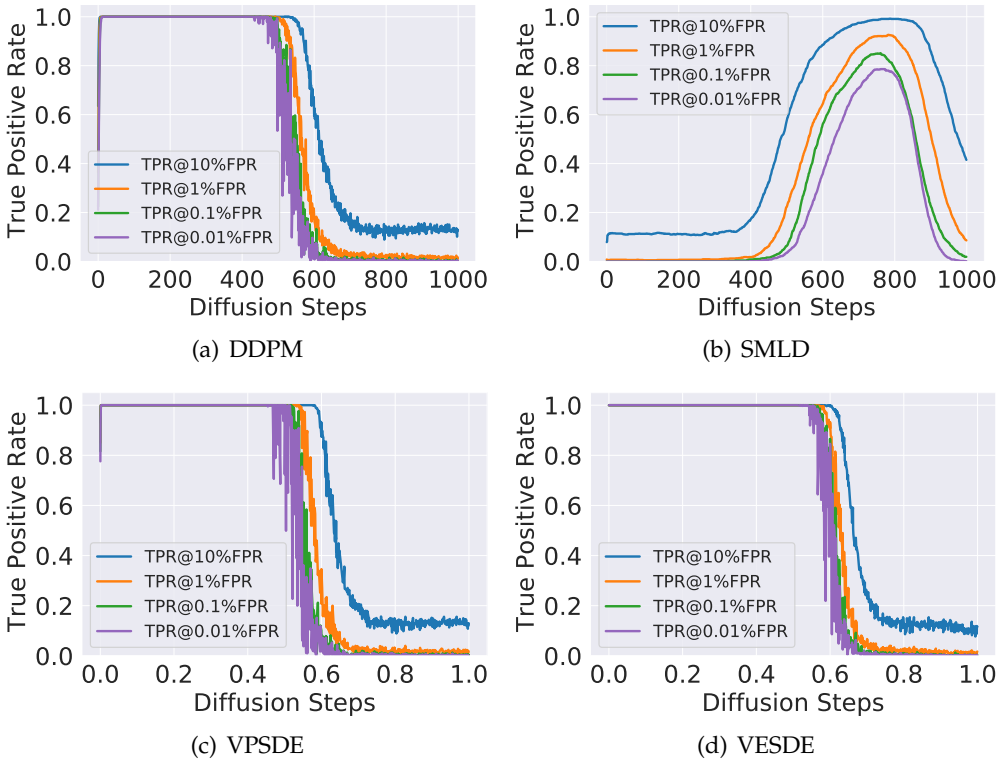
5.4.2 Performance of Loss-based Attack

We present our attack performance from two aspects: TPRs at fixed FPRs for all diffusion steps and log-scale ROC curves at one diffusion step. The former aims to provide the holistic performance of our attacks in diffusion models. In contrast, the latter concentrates on one diffusion step and is able to exhaustively show TPR values at a wide range of FPR values, which is key to assessing the worst-case privacy risks of a model.

TPRs at fixed FPRs for all diffusion steps. Figure 5.2 shows the performance of our loss-based attack on four target models trained on FFHQ. We plot TPRs at different FPRs with regard to diffusion steps for each target model. Recall DDPM and SMLD models are discrete SDEs while VPSDE and VESDE models are continuous SDEs. Thus, the number of diffusion steps for DDPM and SMLD is finite and is fixed as 1,000, while for VPSDE and VESDE models, we uniformly generate $1,000$ points within $[0,1]$ and compute corresponding losses. Overall, all models are vulnerable to our attacks, even under the worst-case, i.e. TPR at 0.01% FPR, depicted by the purple line of Figure 5.2.

We observe that *our attack presents different performances in different diffusion steps.* To be more specific, there exist high privacy risk regions for diffusion models in terms of diffusion steps. In these regions (i.e. diffusion steps), our attack can achieve as high as 100% TPR at 0.01% FPR. Even for the SMLD model, close to 80% TPR at 0.01% FPR can be achieved. Recall the training mechanisms of diffusion models. Different levels of noise at different diffusion steps are added during the forward process. DDPM and VPSDE and VESDE progressively introduce the growing levels of noise while SMLD starts with maximum levels of noise and gradually decreases the levels of noise. Thus, we can see that these models (DDPM and VPSDE, and VESDE) are more vulnerable to leak training samples in the first half part of the diffusion steps while the SMLD model shows membership vulnerability in the second half part of the diffusion steps.

In addition, as shown in Figure 5.2, we also see that four curves that represent the true positive rate at different false positive rates almost overlap or are very close



(a) DDPM (b) SMLD

(c) VPSDE (d) VESDE

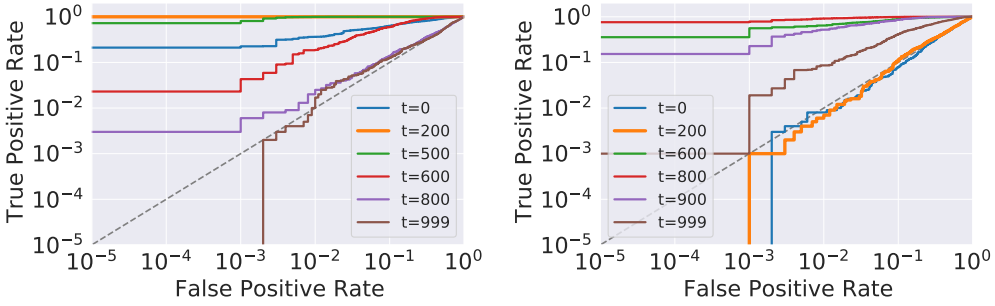
Figure 5.2: Performance of the loss-based attack on all diffusion steps. Target models are trained on FFHQ.

in most diffusion steps. It indicates that our attack can still be effective and robust even at the low false positive rate regime. Note that TPR at low FPR is able to characterize the worst-case privacy risks.

In brief, all models are prone to suffer from membership leakage in low levels of noise while they become more resistant in high levels of noise. In fact, in these diffusion steps where high levels of noise are added to training data, perturbed data is almost close to pure Gaussian noise, which can to some degree enhance the privacy of training data. We also notice that at the starting diffusion step, our attack performance suffers from a decrease. This is because there is an instability issue at this step during the training process [SSDK+21]. Despite this, these peak regions still show the effectiveness of our attack.

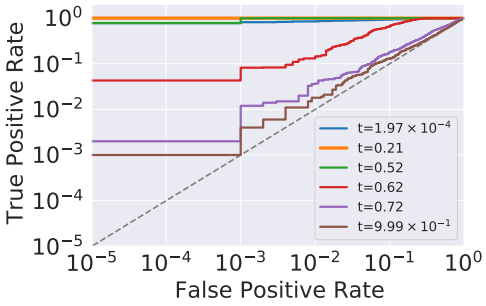
Log-scale ROC curves at one diffusion step. Figure 5.3 plots full log-scale ROC curves of the loss-based attack on four target models. We choose six different diffusion steps for each target model. The rules of choosing diffusion steps for discrete SDEs (i.e. DDPM and SMLD) are: starting and ending diffusion step and the diffusion step that experiences significant changes in terms of attack performance. For continuous SDEs (i.e. VPSDE and VESDE), we first get $1,000$ points that are uniformly sampled from $[0,1]$. Then, we choose diffusion steps from these points based on the same rule of discrete SDEs. Overall, our excellent attack performance is exhaustively shown through log-scale ROC curves.

We can observe that when the levels of noise are not too large, our method can achieve a perfect attack, such as at $t = 200$ for the DDPM model, $t = 800$ for the SMLD model, and $t = 0.21$ for the VPSDE and VESDE models. Again, we can

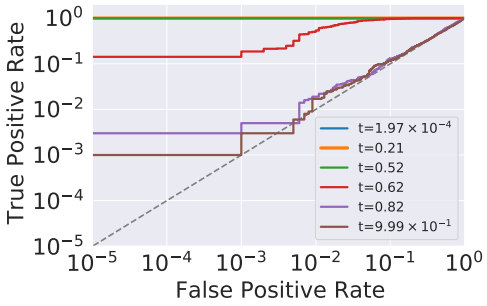


(a) DDPM

(b) SMLD



(c) VPSDE



(d) VESDE

Figure 5.3: Performance of the loss-based attacks at one diffusion step. Target models are trained on FFHQ.

Table 5.2: Performance of the loss-based attack on target models trained on FFHQ.

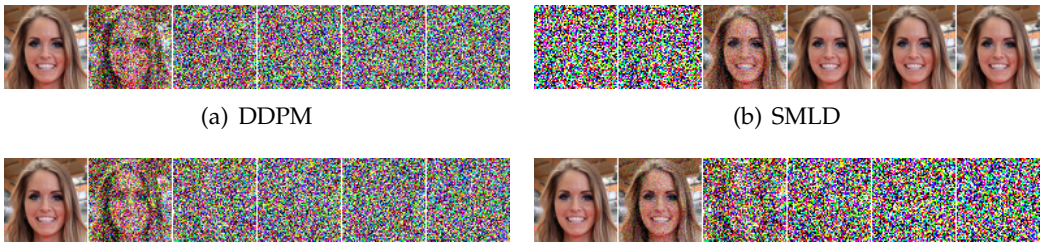
Models	T	TPR@ 10%FPR	TPR@ 1%FPR	TPR@ 0.1%FPR	TPR@ 0.01%FPR	Accuracy	Models	T	TPR@ 10%FPR	TPR@ 1%FPR	TPR@ 0.1%FPR	TPR@ 0.01%FPR	Accuracy
DDPM	0	63.50%	36.40%	22.50%	21.10%	78.25%	SMLD	0	7.90%	0.80%	0.00%	0.00%	51.20%
	200	100.00%	100.00%	100.00%	100.00%	100.00%		200	11.20%	0.70%	0.10%	0.00%	52.30%
	500	100.00%	99.50%	80.80%	72.50%	99.30%		500	88.50%	64.40%	56.10%	35.70%	89.50%
	600	59.50%	18.80%	4.30%	2.30%	81.15%		800	99.10%	91.70%	78.60%	76.10%	96.40%
	800	13.90%	2.50%	0.60%	0.30%	52.80%		900	85.80%	52.00%	22.80%	15.30%	88.80%
	999	12.60%	1.70%	0.00%	0.00%	52.45%		999	41.50%	8.60%	1.90%	0.10%	70.55%
VPSDE	1.97×10^{-4}	93.00%	85.00%	81.60%	77.60%	93.15%	VESDE	1.97×10^{-4}	100.00%	100.00%	100.00%	100.00%	100.00%
	0.21	100.00%	100.00%	100.00%	100.00%	100.00%		0.21	100.00%	100.00%	100.00%	100.00%	100.00%
	0.52	100.00%	100.00%	99.50%	78.40%	99..90%		0.52	100.00%	100.00%	100.00%	99.90%	99.95%
	0.62	66.50%	14.50%	8.20%	4.30%	85.70%		0.62	96.00%	53.60%	18.60%	14.20%	93.25%
	0.72	17.90%	3.70%	1.20%	0.20%	57.30%		0.82	13.10%	1.90%	0.50%	0.30%	52.50%
	9.99×10^{-1}	13.00%	1.8%	0.40%	0.10%	52.20%		9.99×10^{-1}	11.60%	1.70%	0.30%	0.10%	51.50%

clearly see that the ROC curves on all target models are more aligned with the grey diagonal line with the increase in the magnitudes of noise. The grey diagonal line means that the attack performance is equivalent to random guesses. For example, the ROC curves are almost close to the grey diagonal line when the maximal level of noise is added, such as the DDPM model at $t = 999$, the SMLD model at $t = 0$, and the VPSDE and VESDE models at $t = 9.99 \times 10^{-1}$. It is not surprising because at that time the input samples are perturbed as Gaussian noise data in theory and indeed do not have something with original training samples.

Table 5.2 summarizes our attack performance on four target models with regard to diffusion steps and FPR values. We also report the average metric accuracy for reference. Here, we emphasize that only focusing on average metrics cannot assess the worst-case privacy risks. For instance, for the DDPM model at $t = 800$, the attack accuracy is 52.80%, which indicates the model at this diffusion step almost does not lead to the leakage of training samples, because it is close to 50% (the accuracy of

random guesses). In fact, the TPR is 0.3% at the false positive rate of 0.01%, which is 30 times more powerful than random guesses. It means that adversaries can infer confidently member samples under extremely low false positive rates.

Figure 5.4 shows perturbed data of four target models under different diffusion steps. The diffusion steps in Figure 5.4 are corresponding to those in Figure 5.3. We observe that even when some perturbed data that is almost not recognized by human beings is used to train the model, it seems not to prevent model memorization. For example, for the DDPM model at $t = 600$, the perturbed image is meaningless for humans. However, the attack accuracy is as high as 81.15%. At the same time, the TPR at 0.01% FPR is 2.30%, which is 230 times more powerful times than random guesses. It indicates that models trained on perturbed data, except for Gaussian noise data, can still leak training samples. *The noise mechanism of diffusion models does not provide privacy protection.*



(a) DDPM (b) SMLD

(c) VPSDE (d) VESDE

Figure 5.4: Perturbed data of four target models under different diffusion steps. The diffusion steps correspond to these in Figure 5.3. Specifically, from left to right for each model: DDPM (0, 200, 500, 600, 800, 999); SMLD (0, 200, 600, 800, 900, 999); VPSDE (1.97×10^{-4}, 0.21, 0.52, 0.62, 0.72, 9.99×10^{-1}); VESDE (1.97×10^{-4}, 0.21, 0.52, 0.62, 0.82, 9.99×10^{-1}).

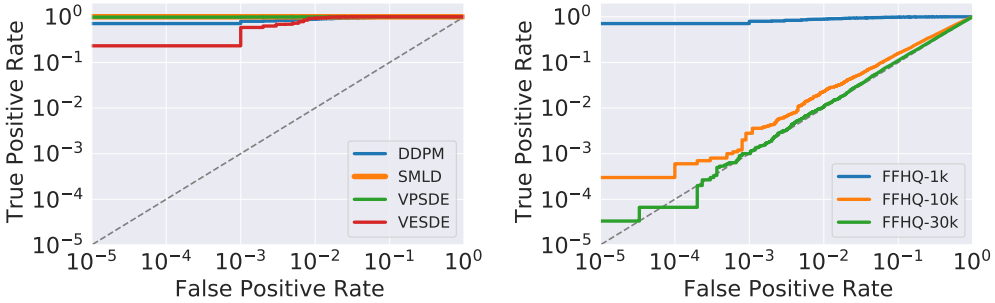
5.4.3 Performance of Likelihood-based Attack

Figure 5.5(a) demonstrates our likelihood-based attack performance on four target models. Overall, our attacks still perform well on all target models. For example, our attack on the SMLD and VPSDE models almost remains 100% true positive rates on all false positive rate regimes. For the VESDE model, attack results are slightly inferior to the SMLD model, yet still higher than the 10% true positive rate at an extremely low 0.001% false positive rate.

Table 5.3 shows our attack results at different FPR values for all target models. Once again, we can clearly see that even at the 0.01% FPR, the lowest TPR among all models is as high as 23.10%, which is 2,310 times than random guesses. In addition, we also observe that the attack accuracy is above 98% for all target models. Our attack results also remind model providers that they should be careful when using likelihood values.

5.4.4 Takeaways

Our loss-based attack utilizes loss values to make a membership inference. Although the loss-based attack requires adversaries to choose a suitable diffusion



(a) Likelihood-based attack on different target models.

(b) Likelihood-based attacks on target models trained on different sizes of datasets.

Figure 5.5: Performance of the likelihood-based attack.

Table 5.3: Likelihood-based attack. Target models are trained on FFHQ.

Models	TPR@ 10%FPR	TPR@ 1%FPR	TPR@ 0.1%FPR	TPR@ 0.01%FPR	Accuracy
DDPM	98.00%	89.00%	79.70%	71.00%	95.75%
SMLD	100.00%	100.00%	100.00%	100.00%	100.00%
VPSDE	100.00%	99.60%	98.90%	98.20%	99.45%
VESDE	100.00%	93.80%	58.40%	23.10%	98.50%

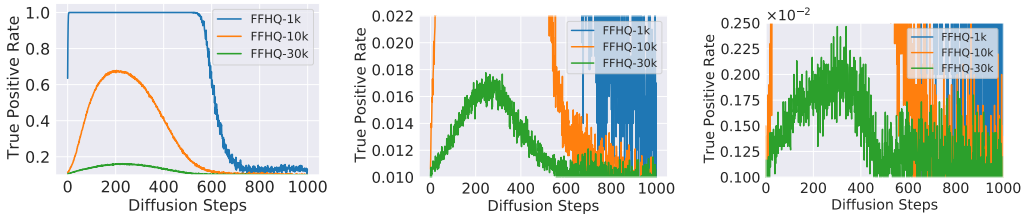
step to mount the attack, our extensive experiments identify the high privacy risk region. More importantly, our loss-based attack reveals the relationship between membership risks and the generative mechanism of diffusion models. This provides a new angle to mitigate membership risks by designing novel noise mechanisms of diffusion models. Our likelihood-based attack does not need to choose a diffusion step and infers membership directly based on likelihood values. Both loss and likelihood information can lead to the leakage of training samples.

5.5 Analysis

In this section, we first analyze our attack performance with regard to the size of a training set. Then, we report our results on a medical image dataset DRD.

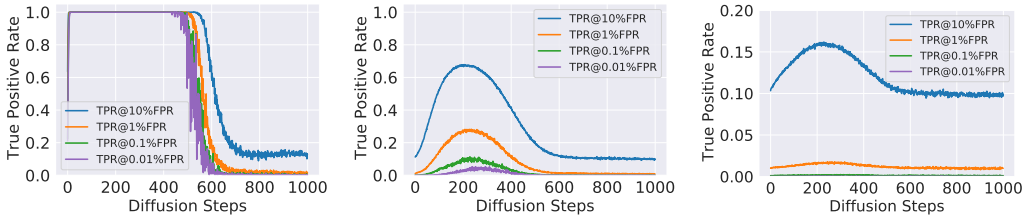
5.5.1 Effects of Size of a Training Dataset

We study attack performance with regard to different sizes of the training set of a target model. Here, we choose the DDPM models trained on FFHQ as target models. We use FFHQ-1k, FFHQ-10k, and FFHQ-30k to represent different sizes of a dataset, which refer to $1,000$, $10,000$, and $30,000$ training samples in each dataset respectively. The FID of the target model DDPM trained on FFHQ-1k, FFHQ-10k, and FFHQ-30k are 57.88, 34.34, and 24.06, respectively. In the following, we present the performance of both attacks.



(a) TPR at 10% FPR (b) TPR at 1% FPR (c) TPR at 0.1% FPR

Figure 5.6: Performance of loss-based attack with different sizes of datasets. The target model is DDPM trained on FFHQ. Each subfigure shows attack performance with different sizes of datasets on fixed FPRs.



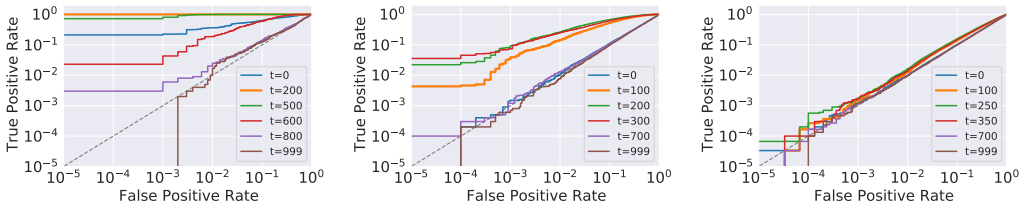
(a) FFHQ-1k. (b) FFHQ-10k. (c) FFHQ-30k.

Figure 5.7: Performance of loss-based attacks with different sizes of datasets. The target model is DDPM trained on FFHQ. Each subfigure shows attack performance with different FPRs on fixed dataset sizes.

Performance of loss-based attack. Figure 5.6 depicts the performance of loss-based attacks on all diffusion steps under different sizes of a training set. Overall, we can observe that attack performance gradually becomes weak when the size of training sets increases. For example, at diffusion step $t = 200$, the TPR at 10% FPR decreases from 100% to about 15% when the training samples increase from 1k to 30k. Here, note that the starting points of the y-axis in Figure 5.6 are not 0 and we set them as the probability of random guesses. Thus, as long as the lines can be shown in the figure, it indicates this is an effective attack.

However, *the peak regions still exist even if the number of training samples increases to 30k and the FPR value is as low as 0.1%.* For instance, as shown in Figure 5.6(c), it shows our attack performance of 0.1% FPR on all models. Diffusion steps in the range of 0 to 400 are still vulnerable to our attack, compared to other steps. It indicates that these diffusion steps indeed lead a model to more easily leak training data. We further show the attack performance based on each dataset in Figure 5.7. Again, we can observe that the peak regions exist, even if the size of a training set increases.

Figure 5.8 shows ROC curves of our attack against target models trained on different sizes of training sets. Based on the same rules described in Section 5.4.2, we select several different diffusion steps and plot their ROC curves. On the one hand, we can see that indeed models become less vulnerable as the number of training samples increases. For instance, Figure 5.8(c) shows the DDPM trained on FFHQ-30K is more resistant to MI attacks on the full log-scale TPR-FPR curve. On the other hand, when diffusion step t equals 250, our attack shows higher attack performance than random guesses at the low false positive rate, such as 10^{-4}. This is also corresponding to the peak steps in Figure 5.6.



(a) FFHQ-1k (b) FFHQ-10k (c) FFHQ-30k

Figure 5.8: Performance of loss-based attack with different sizes of datasets. The target model is DDPM. TPR-FPR Curves under different diffusion steps.

We also observe from Figure 5.8 that TPR values in diffusion steps of high privacy risks do not further go down with the increase in FPR values, especially in extremely low FPR regimes. Take the DDPM trained on FFHQ-30K as an example (see Figure 5.8(c)), the TPR value at diffusion step $t = 250$ are still about 10^{-4} at the FPR value of 10^{-5}, while at $t = 999$, the TPR value at 10^{-5} FPR value is 0. This indicates that at $t = 250$, there are some training samples whose loss values are always smaller than the minimal loss value of the nonmember sample. Otherwise, the green line ($t = 250$) will go down to zero, similar to the brown line ($t = 999$). In other words, there are partial training samples that can be inferred with 100% confidence at this diffusion step. Note that in reality, even if only one sample can be inferred as a member confidently, it still constitutes a severe privacy violation [LF20, HP21a, CCN+22].

Performance of likelihood-based attack. Figure 5.5(b) shows the performance of likelihood-based attacks in terms of different sizes of training sets. Similar to the loss-based attack, the performance of the likelihood-based attack decrease with an increase in the sizes of training sets. Specifically, the likelihood-based attack shows perfect performance on the target model trained on FFHQ-1k. When the size of a training set increases to 10K, there is a significant drop but still better than random guesses on the full log-scale ROC curve. In particular, in the extremely low false positive rate regime, such as 10^{-4}, the true positive rate is about 6×10^{-4}, which is 6 times more powerful than random guesses. In the model trained on FFHQ-30K, the ROC curve is almost close to the diagonal line, which indicates that adversaries are difficult to infer member samples through likelihood values.

5.5.2 Effects of Different Datasets

In this subsection, we show our attack performance on a medical image dataset about diabetic retinopathy. We choose the medical image dataset because the number of images that have diabetic retinopathy is usually insufficient in practice [KAH+22]. These types of images could be used for training a diffusion model and later the trained model is utilized to generate more novel images. We have described this dataset DRD in Section 5.3.1. We choose the SMLD as the target model and the number of training samples is $1,000$. Overall, the SMLD model can achieve excellent performance in image synthesis, with an FID of 33.20. Figure 5.9 visualizes synthetic samples, which all show good quality.

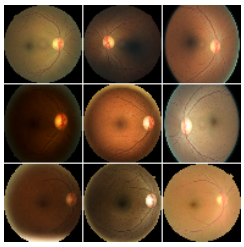
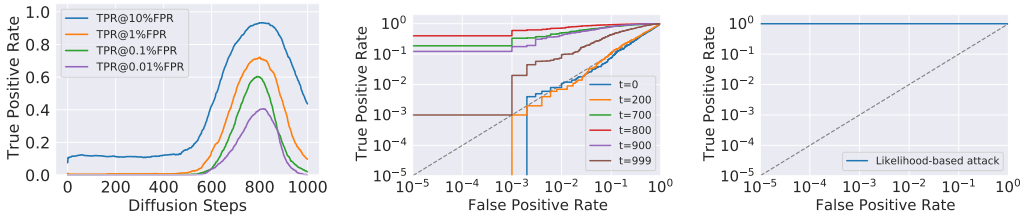


Figure 5.9: Generated images from the target model SMLD trained on the DRD dataset.



(a) Loss-based attack (b) Loss-based attack (c) Likelihood-based attack

Figure 5.10: Attack performance on the DRD dataset.

Performance of loss-based attack. Figure 5.10 shows the performance of loss-based attacks for the target model SMLD trained on DRD. Here, note that the levels of the noise of the SMLD model gradually become small with an increase in diffusion steps. Figure 5.10(a) shows the performance of our loss-based attack on all diffusion steps. Figure 5.10(b) depicts ROC curves for different diffusion steps on target model SMLD trained on DRD. We can again observe our attacks can still perform perfectly on DRD at diffusion steps of low levels of noise. Similar to FFHQ, peak (high-risks) regions can be also seen on SMLD trained on DRD.

Performance of likelihood-based attack. Figure 5.10(c) reports the performance of our likelihood-based attack on the SMLD model trained on DRD. As expected, our attack still shows excellent performance. We can clearly find that the attack achieves 100% TPR on all FPR values, which means that all member samples are inferred correctly. Table 5.4 reports the quantitative results of both attacks.

Table 5.4: Quantitative results of our attacks on SMLD trained on DRD.

Attack	T	TPR@ 10%FPR	TPR@ 1%FPR	TPR@ 0.1%FPR	TPR@ 0.01%FPR	Accuracy
Loss-based	0	7.50%	1.10%	0.00%	0.00%	50.25%
	200	11.20%	0.70%	0.10%	0.00%	52.25%
	700	80.60%	50.50%	33.34%	18.80%	85.45%
	800	93.30%	72.20%	60.00%	40.10%	92.25%
	900	79.80%	42.40%	17.70%	12.30%	86.35%
	999	43.60%	9.70%	2.00%	0.10%	70.95%
Likelihood-based	-	100.00%	100.00%	100.00%	99.90%	99.95%

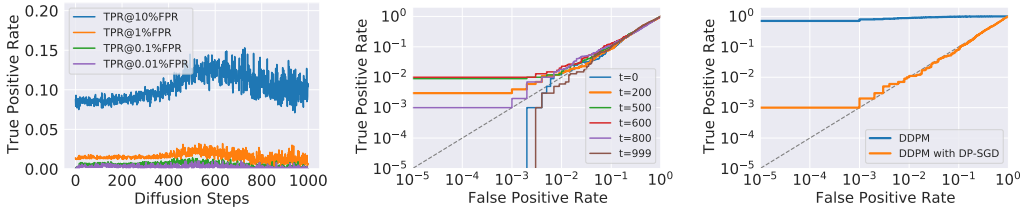
5.6 Defenses

Differential privacy (DP) [ACG+16, Dwo08] is considered as a common defense measure for preventing the leakage of training samples of a machine learning model. In practice, although differential privacy can guarantee individual-level privacy, it often sacrifices significantly model utility, especially for the quality of generated images, when it is applied to generative models. In this section, we present our attack results on diffusion models using the DP defense technology.

We adopt differentially-private stochastic gradient descent (DP-SGD) [ACG+16] to train diffusion models. DP-SGD is widely used for privately training a machine learning model. Generally, DP-SGD achieves differential privacy by adding noise into per-sample gradients. In our work, we implement DP diffusion models through the Opacus library [YSS+21] which allows us to set privacy budgets through hyperparameters. Here, we set the clip bound C and the failure probability δ as 1 and 5×10^{-4}. The batch size and the number of epochs are 64 and $1,800$. Thus, the final privacy budget ϵ is 19.62. Generally, a smaller privacy budget means a higher privacy setting and more severe model utility loss. The common choice of privacy budget is $\epsilon \leq 10$ [YSS+21, ACG+16], and in this chapter, we choose a higher privacy budget because we consider the utility of a diffusion model. We choose the DDPM model as the target model. It is trained on FFHQ containing $1,000$ training samples, and the FID is 393.94.

Performance of loss-based attack. Figure 5.11 shows the performance of both types of attacks on DDPM trained with DP-SGD on FFHQ. In Figure 5.11(a), we present the performance of the loss-based attack on all diffusion steps. Clearly, we can see that although differentially training DDPM, i.e. DDPM with DP-SGD, indeed can significantly decrease the membership leakages, the peak regions can be still identified between 400 and 800 diffusion step. Figure 5.11(b) further shows ROC curves of our loss-based attack on different diffusion steps. Again, we can observe that in the low FPR regimes some training samples are still inferred with a higher probability, such as 10^{-2} TPR at 10^{-4} FPR at t=500. This is higher than 100 times than random guesses (TPR is 10^{-4} at 10^{-4} FPR).

Performance of likelihood-based attack. Figure 5.11(c) shows the performance of the likelihood-based attack on DDPM training with DP-SGD on FFHQ. Again, we can see that differentially private training of a diffusion model indeed can mitigate our attack. At the same time, we also see at the low false positive rate regime, our attack still remains at 0.1% true positive rate, which illustrates the effectiveness of our attack even in the worst-case. Here, we also note that the FID of the target model is 393.94, which means that the utility of the target model suffers from a severe performance drop. We leave developing more usable techniques to train a diffusion model with DP-SGD as future work. Table 5.5 summarizes the quantitative results of both attacks.



(a) Loss-based attack (b) Loss-based attack (c) likelihood-based attack

Figure 5.11: Attack performance on DDPM with DP-SGD.

Table 5.5: Quantitative results of our attacks on DDPM trained with DP-SGD.

Attacks	T	TPR@ 10%FPR	TPR@ 1%FPR	TPR@ 0.1%FPR	TPR@ 0.01%FPR	Accuracy
Loss-based	0	8.80%	1.40%	0.00%	0.00%	52.25%
	200	8.60%	1.40%	0.40%	0.30%	53.20%
	500	**10.70%**	**1.60%**	**0.90%**	**0.90%**	**51.85%**
	600	**13.00%**	**2.30%**	**1.00%**	**1.00%**	**51.85%**
	800	11.60%	2.10%	0.30%	0.30%	51.75%
	999	10.40%	0.60%	0.00%	0.00%	53.90%
Likelihood-based	-	8.40%	1.10%	0.20%	0.10%	51.75%

5.7 Related Work

Diffusion models. Diffusion models have attracted increasing attention in the past years. Sohl-Dickstein et al. [SDWMG15] first introduce nonequilibrium thermodynamics to build generative models. The key idea is to slowly add noise into data in the forward process and learn to generate data from noise through a reverse process. Ho et al. [HJA20] further propose to use parameterization techniques in diffusion models, which enable diffusion models to generate high-quality images. Song et al. [SE19] present to train a generative model by estimating gradients of data distribution, i.e. score. Furthermore, Song et al. [SSDK+21] propose a unified framework to describe these diffusion models through the lens of stochastic differential equations. Beyond image synthesis, diffusion models are also applied to various domains, such as image restoration [SCC+22, WYZ22], and text-to-image translation [NDR+21, RDN+22], even audio and video synthesis [KPH+21, HSG+22]. However, in this work, we study diffusion models from the perspective of privacy.

Membership inference attacks. There are extensive works on membership inference (MI) attacks on classification models [SSSS17, SZH+19, CCN+22, YMMS22, LZBZ22]. Various attack methods under different threat models are proposed, such as using fewer shadow models [SZH+19], using loss values [SSSS17, CCN+22, YMMS22, LZBZ22] and using labels of victim models [CCTCP21, LZ21].

In addition to classification models, there are several MI attacks on early generative models [HMDDC19, HHB19, CYZF20]. Hayes et al. [HMDDC19] leverage the discriminator of a GAN to mount attacks. Chen et al. [CYZF20] perform MI attacks by finding a reconstructed sample on the generator of a GAN. Nevertheless, all attacks

are more specific to GANs and heavily rely on the unique characteristics of GANs, such as discriminators or generators. They cannot be extended to diffusion models, because diffusion models have different training and sampling mechanisms. Therefore, our work on MI of diffusion models aims to fill this gap.

Another recent work proposed by Somepalli et al. [SSG+22b] investigates data replication in diffusion models. However, their work is different from our work. Data replication assumes that adversaries can have the whole training set. Given a generated sample from the diffusion model, they search the training set based on similarity metrics. If the similarity value is higher than a threshold, it is considered a replication for this generated sample. In contrast, our work does not assume that adversaries obtain the training set. Our work aims to infer whether a training sample is used to train the model, given a diffusion model.

Membership inference attacks in diffusion models. In this paragraph, we discuss our work and its relation to several similar/concurrent works studying MI attacks in diffusion models. Wu et al. [WYL+22] study MI attacks against text-to-image generative models. One diffusion-based text-to-image generative model, LDM [RBL+22], is attacked by their methods based on query data pair, i.e. text and corresponding output image. Unlike text-to-image generative models, we focus on unconditional diffusion models. Furthermore, our MI attack methods, such as the loss-based attack, are totally different from their methods [WYL+22]. Subsequently, there are several concurrent works that investigate MI attacks against diffusion models based on the loss information [CHN+23, MMY23, DKW+23, ZCGF23]. However, they only consider discrete diffusion models where the number of noise distributions is finite. *Our work systematically studies both discrete and continuous diffusion models.* Although Carlini et al. [CHN+23] design more sophisticated and effective methods, they require extraordinarily huge computation resources, such as training hundreds of shadow diffusion models or millions of queries from diffusion models. In contrast, our method only utilizes loss values, which is much more computationally efficient. In addition to the attack method based on the loss information, we also propose the likelihood-based method which is not considered in these works [CHN+23, MMY23, DKW+23, ZCGF23].

5.8 Conclusion

In this chapter, we have developed two types of attack methods: loss-based attack and likelihood-based attack. We have evaluated our methods on four state-of-the-art diffusion models and two privacy-related datasets (human faces and medical images). Our methods have demonstrated the connection between membership inference risks and the generative mechanism of diffusion models. To be more specific, our loss-based attack reveals that in terms of diffusion steps, there exist high-risk regions where training samples can be inferred with high precision. Although membership inference becomes more challenging with the increase in the number of training samples, the high-risk regions still exist. Our experimental results on classic privacy protection mechanisms, i.e. diffusion models trained with

DP-SGD, further show that DP-SGD alleviates our attacks at the expense of severe model utility.

Designing an effective differential privacy strategy to produce high-quality images for diffusion models is still a promising and challenging direction. Moreover, developing novel noise mechanisms for diffusion models to prevent leakage of training samples from the loss-based attack, is an appealing research avenue. Finally, it is an interesting direction to study MI attacks of diffusion models in stricter scenarios, such as only obtaining synthetic data.

Chapter 6

Property Inference in Diffusion Models

In the previous chapter, we studied membership inference risks of diffusion models, which focus on inferring individual samples of a training set. In this chapter, we investigate property inference risks of diffusion models, which concentrate on extracting the sensitive global information of a training set from a well-trained diffusion model, such as the proportion of the training data for certain properties. Specifically, we consider the most practical attack scenario: adversaries are only allowed to obtain synthetic data. Under this realistic scenario, we evaluate the property inference attacks on different types of samplers and diffusion models. Furthermore, one case study on off-the-shelf pre-trained diffusion models is also performed in practice. Finally, we propose a new model-agnostic plug-in method PriSampler to mitigate the property inference of diffusion models. PriSampler shows its significantly superior performance to diffusion models trained with differential privacy on both model utility and defense performance.

6.1 Introduction

Diffusion models [SDWMG15], as an emerging class of generative models, have gained widespread adoption in a large number of application areas, such as image synthesis [SE19, HJA20, SSDK+21, KAAL22], text-to-image generation [NDR+21, RDN+22], even text generation [LTG+22, GLF+23], and video creation [KPH+21, HSG+22]. However, when sensitive and private datasets, such as human face data, are applied to train diffusion models, it might cause various privacy breaches.

In general, there are two main types of attacks in relation to privacy: membership inference attacks [SSSS17] and property inference attacks [AMS+15]. Membership inference attacks aim to infer whether one sample was used for training a machine learning model. Recent works have already demonstrated that diffusion models are vulnerable to membership inference attacks and can memorize some training samples [WYL+22, HP23a, CHN+23, MMY23, DKW+23, ZCGF23]. Unlike membership inference attacks focusing on revealing the privacy of the individual samples of a training dataset, the goal of property inference attacks is to infer global properties

of the whole training set of a machine learning model, such as inferring the proportion of certain sensitive properties of a training set. Although a considerable body of works has studied property inference attacks in classification models, such as support vector machines [AMS+15], fully-connected neural networks [GWY+18], convolution neural networks [SE22, MGC22, CAO+23], and generative adversarial networks[ZCSZ22], even graph neural networks [ZCB+22], property inference of diffusion models has not been explored to date. Considering the popularity of diffusion models and the fact that diffusion models are now the dominant paradigm in deep generative modeling [DN21, KAH+22, YZS+22], it is paramount to systematically study the property inference risks of diffusion models.

Attacks. In this chapter, we investigate the privacy risks of diffusion models through the lens of property inference attacks. Our threat model assumes that adversaries can only have access to generated samples from a diffusion model. Based on generated samples, our property inference attack aims to estimate the proportion of various properties of a diffusion model by a property classifier (see Section 6.3.2). We explore four different types of diffusion models, including the discrete variance preserving (VP) model — DDPM [HJA20], the discrete variance exploding preserving (VE) model — SMLD [SE19], the continuous VP model — VPSDE [SSDK+21] and the continuous VE model — VESDE [SSDK+21]. Unlike other generative models, such as generative adversarial networks [GPAM+14] or variational autoencoder [KW14], for a trained diffusion model, there are different sampling methods to generate samples, in which these methods aim to improve the quality of generated samples or sampling speed during the sampling process. Therefore, we study three samplers over two different types of sampling mechanisms, including stochastic sampling — PC sampler [SSDK+21] and deterministic sampling — the black-box ODE sampler [DP80] and the DPM sampler [LZB+22].

Our comprehensive experiments on the human face dataset CelebA [LLWT15] show that state-of-the-art diffusion models and their samplers are vulnerable to property inference attacks (see Section 6.3.4). Adversaries can precisely infer the proportion of sensitive properties with 0% absolute difference in the best case and below 7% absolute difference in the worst case, where the absolute difference refers to the difference between the inferred proportion and the real proportion. We further explore the performance of property inference in terms of different properties, the number of generated samples, model performance, and the size of training sets.

Case study. We conduct one case study where target models are from off-the-shelf diffusion models EDM [KAAL22] trained on a human face dataset FFHQ [KLA19]. We again see similar excellent attack performance on EDM models. Our attack results show that at least 0.23% absolute difference and at most 3.97% absolute difference can be achieved on inferring the property Young. For the property `gender`, the absolute difference does not exceed 1.23%, which indicates the inferred proportion is highly close to the real proportion (see Section 6.4).

Defenses. To defend against property inference attacks, we propose a property aware sampling method — PriSampler, which manipulates diffusion models in the

sampling process to conceal the real proportion of sensitive properties. More specifically, our defense method first finds the hyperplanes of properties in the diffusion models, and the learned hyperplanes are utilized to guide one sampler to synthesize samples in this property space (see Figure 6.8). We show the effectiveness of our defense method PriSampler on different types of samplers, diffusion models, more properties, and the number of generated samples and different diffusion steps (see Section 6.5.5). We also compare PriSampler with differentially private diffusion models (DDPMs) [DCVK22], and evaluations show that PriSampler is superior to DDPMs on model utility and defense performance (see Section 6.5.6). More importantly, PriSampler does not require re-training a diffusion model.

6.2 Motivation and Threat Model

We elaborate on the motivations for our research in Section 6.2.1 and introduce our threat model in Section 6.2.2.

6.2.1 Motivation

Diffusion models are widely used in many application domains [YZS+22, KAH+22], and one of the primary purposes is to leverage these state-of-the-art diffusion models to generate a diversity of novel images. Beyond these purposes, malicious adversaries might reveal some sensitive information of a training dataset through diffusion models, which usually do not intend to be shared by model owners or model providers. For example, a diffusion model trained on a human face dataset is used to generate various different and realistic images. When malicious adversaries obtain this model, instead of only synthesizing novel samples, they could use the shared model to infer sensitive properties, such as the proportion of gender, ethnicity, age, or sentiment. Similarly, even if adversaries only have access to generated samples that are released by model owners, it is possible to infer sensitive information of the training set leaked through generated samples. As a consequence, by successfully inferring sensitive information from a training set, the adversaries can know the proportion of gender or ethnicity of a training set, even the sentiment inclination (negative or positive). These sensitive properties are usually protected and model owners do not intend to share them.

In addition, some properties, such as gender, usually are related to the fairness of a machine learning model. Even though model owners can ensure the fairness of a training set in terms of these properties, a diffusion model might induce unfairness from other factors, such as generative algorithms themselves. Once the adversaries correctly estimate the proportion of these properties and make them public, model owners will face accusations from data regulation institutions.

Therefore, in this chapter, our goal is to systematically study the feasibility of property inference attacks against diffusion models by considering different state-of-the-art diffusion models and their samplers. Furthermore, based on a wide range of

investigations in property inference of diffusion models, we will develop a brand-new and effective defense method to mitigate this type of attack. In the end, we hope our work can elevate the awareness of preventing property inference attacks and encourage privacy-preserving synthetic data release.

6.2.2 Threat Model

Our threat model considers that adversaries only obtain generated samples from a diffusion model. The adversaries do not know the type of diffusion models and the type of their samplers, which is usually the strictest and most practical scenario. Furthermore, we assume that the adversaries have a shadow dataset which contains properties that adversaries intend to infer.

6.3 Property Inference Attacks

The objective of a property inference attack is to predict the proportion of a property in the training set of a trained diffusion model. This makes adversaries reveal some sensitive information that is not shared by model owners. For instance, in addition to directly utilizing generated samples from a diffusion model, adversaries could also attempt to infer sensitive information disclosed by these generated samples, such as the proportion of the property `male`. It is thus important to investigate the feasibility of property inference attacks against diffusion models. This section starts with problem formulation, we then introduce the attack method and experimental setups. Finally, we present attack results and novel insights.

6.3.1 Problem Formulation

A training dataset $\mathcal{D}$ has different properties including sensitive ones. Each property is binary and has a real proportion p_{s_i} in dataset $\mathcal{D}$. A diffusion model $\mathcal{G}$ is trained on the dataset $\mathcal{D}$. Now, given m generated samples X from the diffusion model $\mathcal{G}$, and a property s_i, adversaries aim to infer the proportion of the property $\hat{p}_{s_i}$, in order to make $\hat{p}_{s_i}$ as close p_{s_i} as possible. More specifically, the adversaries need to design an attack algorithm $\mathcal{A}$ to estimate $\hat{p}_{s_i} = \mathcal{A}(X)$.

6.3.2 Attack Method

The intuition of our attack is that generated samples have a similar distribution to the training set because a diffusion model learns the distribution of a training set and these generated samples are produced by the diffusion model. Therefore, adversaries might infer the proportion of the (sensitive) properties of the training set from these generated samples. To achieve this, we will first deploy a property classifier to predict the property of generated samples, and then use the average statistics of these generated samples containing the property as the inferred proportions.

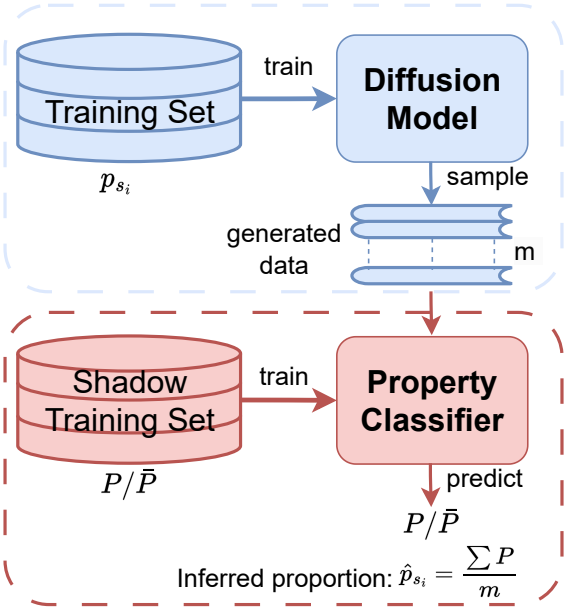


Figure 6.1: The attack process of the property inference attack.

Figure 6.1 illustrates the attack process of the property inference attack. Firstly, a property classifier is trained on a shadow training set. The shadow training set is labeled by the property that adversaries are interested in. P refers to one sample containing this property while $\bar{P}$ refers to one sample not containing this property. After finishing the training on the shadow data set, the property classifier takes as input generated data from the diffusion model and outputs binary predictions. Finally, the inferred proportion of one property is estimated by $\hat{p}_{s_i} = \frac{\sum P}{m}$. For k properties, we will train k property classifiers. Note that, unlike property inference attacks that consider them as a classification problem, i.e. inferring whether a machine learning model contains a property [GWY+18, MGC22, ZCB+22], here we directly estimate the proportion of a property for diffusion models, which is more precise.

6.3.3 Experimental Setups

Datasets. We conduct our experiments on the CelebA dataset which includes 202,599 images of celebrity faces [LLWT15]. The reason why we choose this dataset is that it provides a large number of property information. This allows us to more systematically study property inference attacks by considering different proportions of properties. Specifically, the CelebA dataset annotates 40 binary properties for each image. Similar to works [GWY+18, MGC22, ZCSZ22, CAO+23], we choose four representative properties including `male`, `young`, `smiling`, and `wearing eyeglasses`, which are related to `gender`, `age`, `sentiment`, and `personal style`. In this chapter, when it is clear from the context, we will interchangeably use the property name annotated by CelebA, such as `male` and `young`, and the high-level semantic name, such as `gender` and `age`.

We design ten datasets with five different proportions of private properties and two different sizes of the training set (i.e. $10 = 5 \times 2$), to investigate the privacy risks

of diffusion models. Two sizes of training set include 1k samples and 50k samples, respectively. Five different proportions refer to that male face images account for 10%, 20%, 30%, 40%, and 50% in a dataset, respectively. Different proportions of the `male` property also mean different proportions of the other properties, such as `young` and `smiling`, although the proportions of these properties are not as in sequential order as that of the property `male`. To briefly express its meaning, we mark a dataset as CelebA-size-proportion, such as CelebA-1k-10%. All datasets are resized to 64×64, considering the factors of computation efficiency.

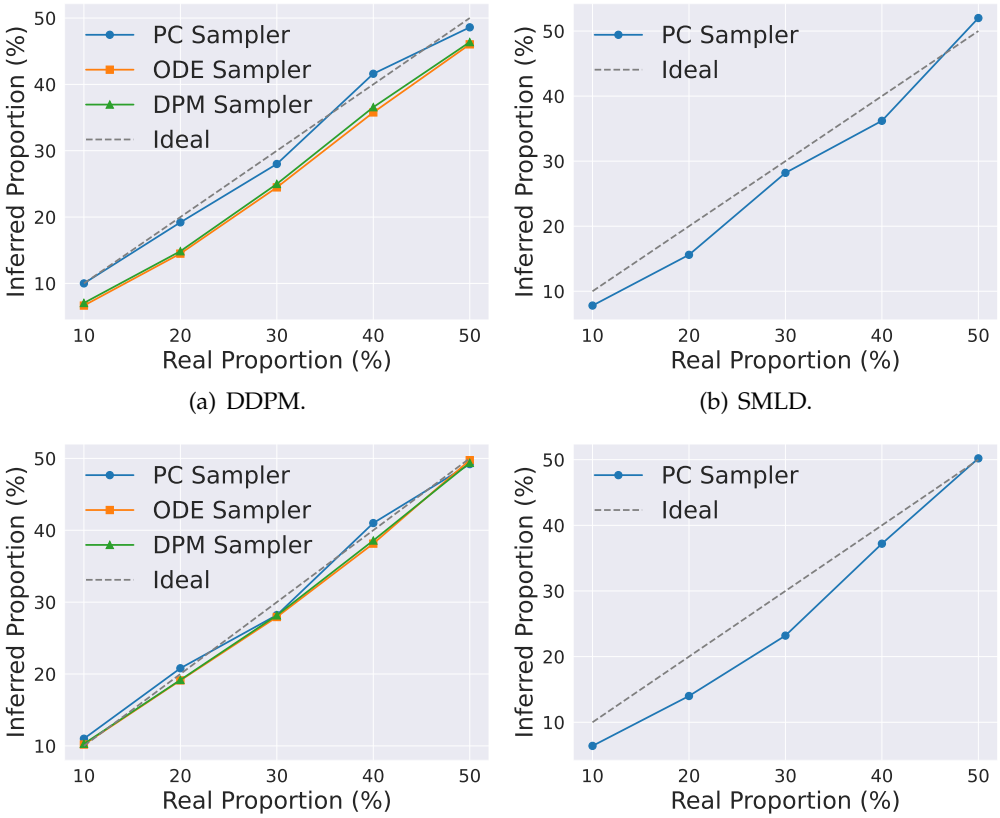
Target models and samplers. We use four types of diffusion models: DDPM, SMLD, VPSDE, and VESDE, as the target models, which have been introduced in Section 2.1.2. DDPM and VPSDE are trained on 1k samples with five different proportions, while SMLD and VESDE are trained on 50k samples with five different proportions. In total, 20 diffusion models are trained on different training sets. We use open source codes in this library[1] with their suggested training hyperparameters to train each diffusion model. Specifically, the number of training steps for all models is fixed at $500,000$.

We choose three typical samplers: one stochastic sampling — PC sampler, and two deterministic samplings — ODE sampler and DPM sampler. We detail these samplers in Section 2.1.3. We adopt this library[1] for PC and ODE samplers and this library[2] for the DPM sampler. The recommended sampling hyperparameters in each implementation are adopted. Specifically, the number of sampling steps of DPM is fixed at 40. For the PC sampler, the number of steps of predictor and corrector is 1,000 and 1, respectively. We only study PC samplers for VESDE and SMLD, because they do not support deterministic samplings. However, we investigate all types of samplers for DDPM and VPSDE.

Attack models. We use the ResNet-50[3] pre-trained on ImageNet [RDS+15] to train a property inference classifier. The shadow training set used for training the classifier is from the remaining samples of the CelebA dataset. In other words, one part of the whole CelebA dataset is used for training diffusion models while the other part, i.e. the shadow training set, is used for training property classifiers. Note that they are disjoint. This is a common practice in the community of privacy in machine learning [GWY+18, ZCSZ22, CAO+23]. In the case study of Section 6.4, we show that the classifier works just as well for diffusion models trained on different humane face datasets. More specifically, we train classifiers with the stochastic gradient descent optimizer. The learning rate and weight decay of all property classifiers are both 0.01, except for the classifier of the property `young` where both values are set as 0.005. The number of training epochs is set as 5.

Metrics. In terms of the performance of diffusion models, we use the widely-adopted Fréchet Inception Distance (FID) metric. A lower FID means that a sampler of a diffusion model can generate more realistic and diverse samples. In this work, by default, we compute an FID with all training samples and 50k generated samples

[1] https://github.com/yang-song/score_sde_pytorch
[2] https://github.com/LuChengTHU/dpm-solver
[3] https://download.pytorch.org/models/resnet50-19c8e357.pth



(a) DDPM. (b) SMLD.

(c) VPSDE. (d) VESDE.

Figure 6.2: Attack performance on different diffusion models, different samplers, and different proportions of the private property. Here, the sensitive property is `male`.

for the ODE and DPM samplers, and 500 generated samples for the PC sampler. This is because the PC sampler requires a much longer time to synthesize data, compared with deterministic ODE and DPM samplers.

In terms of attack performance, our property inference attacks predict a real number, i.e. the proportion of a property. Thus, we show the attack performance by directly presenting the predicted value. In addition, we also report the absolute difference Δs_i between the predicted value and the real value, i.e. $\Delta s_i = |\hat{p}_{s_i} - p_{s_i}|$. A smaller absolute difference value means a more precise inference.

6.3.4 Attack Results

In this section, we present the attack results in terms of different samplers, diffusion models, properties, the number of generated samples, FID values, and the size of training sets.

Attack performance on different samplers. Figure 6.2 shows attack performance with regard to different samplers over four types of diffusion models. Each type of diffusion model is trained on datasets with different proportions of the sensitive property `male`. Here, the real proportion of the property `male` is set as 10%, 20%, 30%, 40%, and 50%, respectively. An ideal attack means that the inferred proportion is equal to the real proportion. We take it as a reference and it is shown as the grey diagonal line in Figure 6.2. Overall, all types of samplers cannot defend against

the property inference attack. Our inferred proportions are consistently close to the real proportions with the increase in the real proportion of the property `male`. We do not show the attack performance on the ODE sampler and DPM sampler for SMLD and VESDE models, because both samplers do not support these models.

In Table 6.1, we show the corresponding quantitative attack results. Again, we can observe that the best inference performance can be seen on the PC sampler for the DDPM trained on CelebA-1k-10%, where the absolution difference is 0%. Even in the worst case for adversaries, at most 6.8% absolution difference can be achieved on the PC sampler for the VESDE trained on CelebA-50k-30%. Table 6.2 summarizes attack performance in terms of different samplers. We can see that our attacks show the best performance in the PC sampler and slightly inferior performance on the ODE sampler. In a nutshell, our attacks can have at most a 2.78% absolute difference among the three types of samplers.

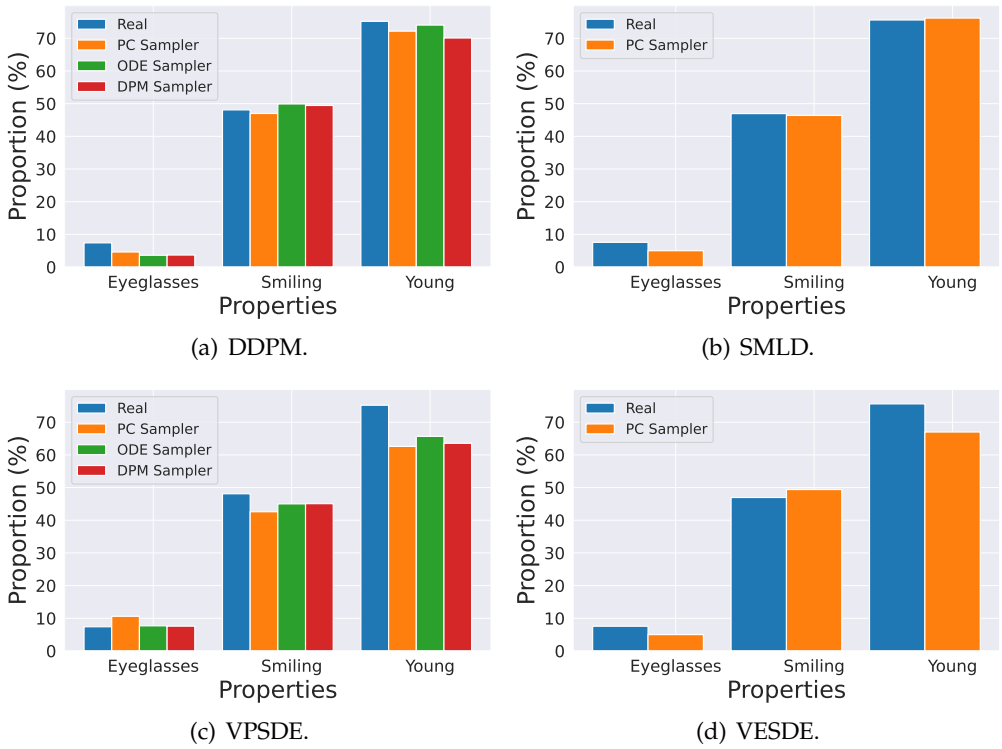
Table 6.1: The qualitative attack results for the sensitive property `male`. Prop.: proportion. Abs. Diff. : absolute difference.

Model	Real Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID	Model	Real Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID	Model	Real Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID	Model	Real Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID
	PC Sampler									ODE Sampler									
	10	10.00	0.00	24.45		10	7.80	2.20	23.24		10	6.66	3.34	16.80		10	10.19	0.19	5.59
	20	19.20	0.80	24.85		20	15.60	4.40	24.64		20	14.49	5.51	17.12		20	19.11	0.89	5.64
DDPM	30	28.00	2.00	25.55	SMLD	30	28.20	1.80	24.72	DDPM	30	24.47	5.53	17.41	VPSDE	30	27.93	2.07	5.97
	40	41.60	1.60	26.57		40	36.20	3.80	25.08		40	35.77	4.23	17.61		40	38.14	1.86	5.95
	50	48.60	1.40	28.96		50	52.00	2.00	25.48		50	46.03	3.97	18.46		50	49.75	0.25	6.22
	PC Sampler									DPM Sampler									
	10	11.00	1.00	19.22		10	6.40	3.60	37.75		10	7.05	2.95	22.60		10	10.29	0.29	8.26
	20	20.80	0.80	20.97		20	14.00	6.00	42.08		20	14.85	5.15	23.47		20	19.18	0.82	8.54
VPSDE	30	28.20	1.80	20.22	VESDE	30	23.20	6.80	35.94	DDPM	30	24.99	5.01	23.34	VPSDE	30	28.16	1.84	8.68
	40	41.00	1.00	20.62		40	37.20	2.80	55.89		40	36.56	3.44	23.43		40	38.57	1.43	8.75
	50	49.20	0.80	21.65		50	50.20	0.20	39.31		50	46.37	3.63	24.47		50	49.37	0.63	8.96

Table 6.2: Summary of attack performances with different types of samplers and diffusion models. Here, we report the average absolution difference (with standard deviation in parentheses) and the best and worst absolution difference.

		Average (%)	Best (%)	Worst (%)
Sampler	PC	2.24 (1.84)	0.00	6.80
	ODE	2.78 (2.02)	0.19	5.53
	DPM	2.52 (1.78)	0.29	5.15
Model	DDPM	3.24 (1.75)	0.00	5.53
	VPSDE	1.04 (0.62)	0.19	2.07
	SMLD	2.84 (1.18)	1.80	4.40
	VESDE	3.88 (2.64)	0.20	6.80

Attack performance on different diffusion models. As shown in Figure 6.2, we present the attack performance on twenty diffusion models encompassing four different types. Each subfigure presents the attack performance of each type of diffusion model. Table 6.1 shows the corresponding quantitative results of each diffusion model. Overall, all diffusion models are vulnerable to property inference attacks. Although each trained diffusion model can utilize different samplers, we can see that adversaries can still efficiently extract these sensitive information of a training



(a) DDPM. (b) SMLD.

(c) VPSDE. (d) VESDE.

Figure 6.3: Attack performance on different properties.

set, regardless of the used samplers. In particular, our attack can achieve almost perfect inference on VPSDE models for all samplers. The reason why the property inference attack is effective on these diffusion models is that all existing samplers mainly focus on improving the quality of generated samples or sampling speed. The other equally important issue, i.e. privacy, is not considered in their design. In Section 5.6, we will take the first step to provide privacy protection by developing a property aware sampling method for diffusion models.

Table 6.2 summarizes attack results about different types of diffusion models. We can see our attacks show the best performance on the VPSDE with an average absolution difference of 1.04%, and show marginally worse performance on the VESDE where the average absolute difference is 3.88%.

Attack performance on different properties. In addition to the property `male`, we also choose other properties. As introduced in Section 6.3.3, we choose three more properties based on their different proportions in the CelebA dataset. The three properties are `eyeglasses`, `smiling`, `young` and their real proportions are roughly below 10%, close to 50%, and above 70%, respectively. The specific real proportions of these properties are plotted by blue bars in Figure 6.3. Here, we choose each type of the model with 50% `male` as the target model. Again, we can observe that our attack still remains effective on inferring the proportions of these properties on all diffusion models and samplers. No matter what the smaller proportion of the property, such as `eyeglasses`, or the larger proportion of the property, such as `young`, the inferred proportions are very close to the real proportions.

Attack performance on different numbers of generated samples. Figure 6.4 shows the attack performance on different numbers of generated samples. Here, we

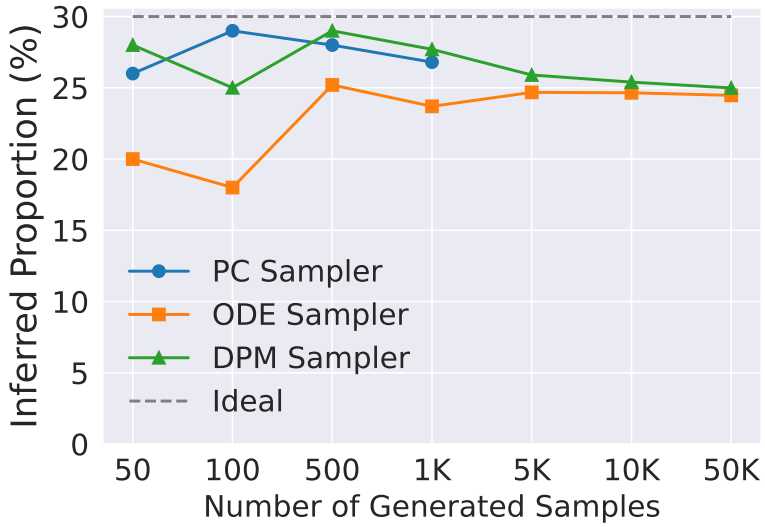


Figure 6.4: Attack performance with respect to different numbers of generated samples. The target model is DDPM trained on CelebA-1k-30%.

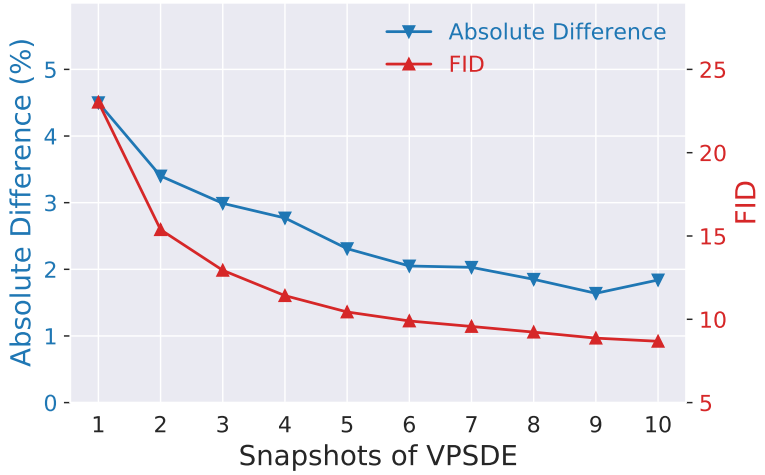


Figure 6.5: Performance with respect to different FID values. The target model is VPSDE trained on CelebA-1k-30%.

choose the DDPM model trained on a dataset that contains 30% `male` training samples, as the target model. We can observe that with the increase in the number of generated samples, the attack performance will gradually become stable. Our attack can be still successful on the PC sampler and DPM sampler even if model owners publish only 50 generated samples, where the absolute difference of both samplers is still below 5%. On the other hand, our attack requires more generated samples for the case of ODE sampler in to achieve a good inference performance but its attack performance shows stable after 500 generated samples.

Attack performance on different FID values. Figure 6.5 shows attack performance in terms of different FID values of target models. Here, we choose the DPM sampler and the VPSDE model trained on CelebA-1k-30%. Furthermore, we choose ten snapshots of VPSDE during the training process. The left axis presents the absolution difference of a target model marked as the blue line, while the right axis shows the FID values of a target model which is marked as the red line. Overall, our attack becomes more accurate with the increase in the performance of target models. Note that a smaller FID means a better utility performance of a target model. This

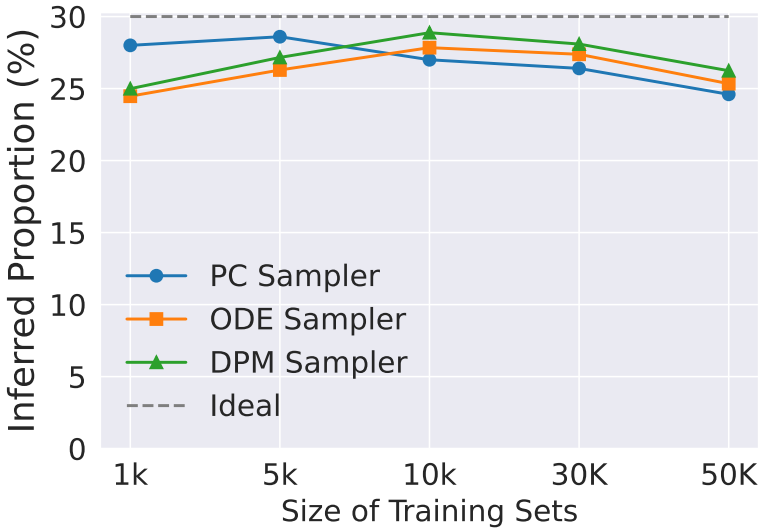


Figure 6.6: Attack performance with respect to different sizes of training sets. The target models are DDPM models trained on CelebA with the property `male` of 30%.

also indicates that pursuing the good utility performance of a diffusion model can lead to more severe privacy risks. Both model utility and privacy risks should be considered when diffusion models involve sensitive data.

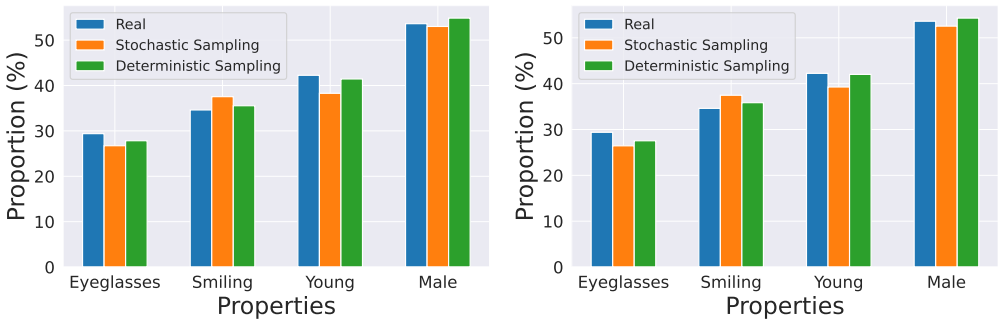
Attack performance on different sizes of training sets. Figure 6.6 plots attack performance in terms of sizes of training sets. Here, the target models are the DDPM models trained on a dataset containing 30% male training samples. Therefore, the real proportion of the property `male` is 30%. We can see that the inference performance slightly decreases with the increase in the size of training sets. For example, for the DPM sampler, when the size of training sets increases from 10k to 50k, the inferred proportions decrease from about 29% to around 26%. Overall, the inferred proportions for all samplers fluctuate between 25% and 30%.

Takeaways. In summary, (1) both stochastic sampling and deterministic sampling are susceptible to the property inference attack. (2) Different types of diffusion models cannot defend against this attack. (3) No matter how large or small the proportion of certain properties is, our attack can precisely infer them. (4) Inference performance becomes gradually stable after releasing 500 generated samples. (5) The better the utility performance of a diffusion model is, the better the attack performance of property inference.

6.4 Case Study: Attacks in Practice

In this section, we further demonstrate the property inference risks in practice through one case study in which we perform property inference attacks against publicly available well-trained diffusion models.

We choose EDM models proposed by Karras et al. [KAAL22] as target models. EDM models achieve competitive performance in image synthesis by a design space to decouple complex components. Similar to SSDE introduced in Section 2.1.2, EDM models include VP and VE formulations, and in this chapter, we



(a) VPEDM. (b) VEEDM.

Figure 6.7: Attack performance on the EDM models.

call them VPEDM and VEEDM, respectively. For each model type, they also have two types of sampling methods to synthesize samples: stochastic sampling and deterministic sampling. In our experiments, we conduct property inference attacks on VPEDM and VEEDM. They are both trained by their original authors on the Flickr-Faces-HQ (FFHQ) dataset which contains 70,000 human face images [KLA19]. All samples of the FFHQ dataset used for training have 64×64 resolution.

Similarly, we assume that only generated samples can be obtained by adversaries. Because the FFHQ dataset does not annotate the properties of each image, here we use the proportion of the property in the training set inferred by our property inference classifier as the real proportion. Although this might bring some errors due to a lack of human annotation, we report the attack performance by both the inferred proportion and the absolute difference. The absolute difference can eliminate this type of error because it shows attack performance by how close the real and inferred proportions are. In this case study, we directly use the property classifier used in Section 6.3 to infer the proportion of different properties, which also illustrates the shadow dataset does not have to be from the same dataset of the target models. 50,000 generated samples for all sampling methods and diffusion models are used to perform the property inference. We choose four properties: `eyeglasses`, `smiling`, `young` and `male`.

Results. Figure 6.7 presents the performance of property inference attacks against EDM models over four properties. Overall, our attack can achieve a rather precise estimation for the proportion of each property. Although the real proportion of these four properties has wide ranges from 29% to 53%, we can observe that the inferred proportions of various private properties are all close to the real proportions. In addition, different types of sampling methods show similar high privacy risks in all properties and diffusion models.

Table 6.3 describes quantitative attack results on the EDM models. We can see that all samplers can achieve a good performance, obtaining an FID value between 2 and 3. We also report the absolute difference. The minimal absolute difference is 0.23%, which can be seen in inferring the property `young` on VEEDM using deterministic sampling. When inferring the property `young` in VPSDE under the stochastic sampling, our attack shows a little inferior performance with an absolute difference of 3.97%. To sum up, our attack on the EDM models can achieve at most a 4% absolute difference.

Table 6.3: Quantitative attack results on the EDM models. Stoch.: Stochastic; Deter.: Deterministic.

Model	Sampler	FID	Property	Real Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)
VPEDM	Stoch.	2.87	Eyeglasses	29.39	26.73	2.66
			Smiling	34.61	37.54	2.93
			Young	42.25	38.28	3.97
			Male	53.62	53.01	0.61
	Deter.	2.47	Eyeglasses	29.39	27.83	1.56
			Smiling	34.61	35.54	0.93
			Young	42.25	41.44	0.81
			Male	53.62	54.85	1.23
VEEDM	Stoch.	2.85	Eyeglasses	29.39	26.44	2.95
			Smiling	34.61	37.47	2.86
			Young	42.25	39.27	2.98
			Male	53.62	52.55	1.07
	Deter.	2.57	Eyeglasses	29.39	27.54	1.85
			Smiling	34.61	35.85	1.24
			Young	42.25	42.02	0.23
			Male	53.62	54.3	0.68

6.5 Defenses

In this section, we shift our focus to mitigating property inference attacks. We first discuss several potential defenses. Then we will introduce a property aware sampling method and present the defense results. Finally, we discuss the defense of diffusion models trained with differential privacy.

6.5.1 Key Idea of Defenses

Property inference attacks leverage generated samples from a diffusion model to estimate the proportions of the properties. To defend against this type of attack, model owners could manipulate the output of a diffusion model to disguise the real proportion of the property p_{s_i}.

In this work, we consider a binary sensitive property s_i, i.e. $s_i = \{0, 1\}$ and $p_{s_i} + \bar{p}_{s_i} = 1$. The goal of our designed defenses is to make adversaries infer the proportion of a property $\tilde{p}_{s_i}$ as close as 0.5.[4] There are at least two reasons. Firstly, it can disguise the real proportion of the sensitive property. Secondly, because some properties, such as `gender`, are usually related to fairness, this choice can also ensure the fairness of a diffusion model.

[4]This can also be a predefined value by the model provider.

6.5.2 Potential Defenses

Based on the key idea that the designed defenses make adversaries infer the proportion of a property as about 0.5, we discuss the following potential defenses.

Dropping some samples of a larger proportion of the property. If the real proportion of a property p_{s_i} is not equal to 0.5, model owners can choose to drop some generated samples of a larger proportion of the property to achieve a balance. To achieve this, model owners need property classifiers for protected sensitive properties and calculate the real-time statistics. In detail, model owners first collect all generated samples before releasing these samples. Then, model owners train a property inference classifier for each property. Finally, the trained classifier is used for predicting properties and some generated samples that have a high proportion of the properties will be dropped.

We assume there are n binary and independent sensitive properties. Furthermore, the proportion of the property s_i satisfies $p_{s_i} + \bar{p}_{s_i} = 1$ and $p_{s_i} \in (0, 0.5)$, where p_{s_i} is the proportion containing this property and $\bar{p}_{s_i}$ is the proportion not containing this property. In the worst case, the proportion of dropping samples is, at most $1 - 2^n \prod_{i=1}^{n} p_{s_i}$. For instance, considering that there is one sensitive property and its real proportion is 10%, i.e. $p_{s_1} = 10\%$ and $n = 1$, then 80% samples among all generated samples will be discarded to achieve the balance, which is quite not economical. In particular, the sampling of diffusion models is time-consuming. For the number of sensitive properties more than one, the number of dropping samples is exponentially increasing. Therefore, this method is simple but not scalable.

Using a balanced dataset for sensitive properties. This method requires model owners to prepare a balanced dataset for sensitive properties, such as using a dataset containing 50% male samples for the property `gender`. Our extensive experiments in Section 6.3.4 show that this method indeed has a positive effect to some degree. However, in some cases even for balanced properties, we also observe that the learned diffusion models will still produce imbalanced generated samples. For instance, the DPM sampler on the DDPM model trained on a 50% `male` dataset produces `male` samples which are about 46.37% of all generated samples. In addition, it is complicated to collect sufficient training samples if we need to consider balancing more sensitive properties.

Property aware sampling. In addition to above discussed methods, we can design a new type of sampling mechanism that can automatically balance the proportion of sensitive properties. In this way, we can avoid the waste of generated samples and fastidious dataset selection. We illustrate this method in the next subsection.

6.5.3 Defense Method — PriSampler

The main purpose of the property aware sampling method is to balance the proportion of sensitive properties in the sampling process of diffusion models. As a result, the inferred proportion always remains at about 0.5. Our method is inspired by the semantic latent space of generative adversarial network (GANs) in

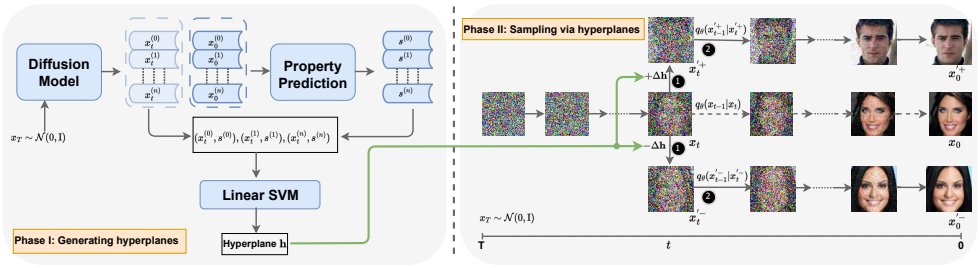


Figure 6.8: The process of the defense method PriSampler. Phase I learns hyperplanes of sensitive properties. Phase II synthesizes samples vis the learned hyperplanes. In ❶, at the t diffusion step, PriSampler changes the intermediate samples to the space of the specific sensitive property via the learned hyperplane. In ❷, PriSampler continues to denoise samples step by step. Eventually, the desired samples ($x_0'^{+}$, $x_0'^{-}$) are obtained in the final step.

this work [SGTZ20]. Due to the essential difference in the sampling process between diffusion models and GANs, we adapt the method to be well suitable for diffusion models and propose PriSampler as a general-purpose defense for diffusion models.

The key idea of PriSampler is to guild one sampler to generate novel samples in the latent space of sensitive properties. For instance, for the gender property, given the corresponding property hyperplane, samplers generate male samples on the side of this hyperplane and female samples on the opposite of this hyperplane. In order to find such a hyperplane, we use a linear support vector machine (SVM) to learn a decision boundary for each sensitive property. Then, given a base sampler, we directly use the boundary to guild the sampler to synthesize new samples. Here, the base sampler can be any sampler that is used for sampling in prior work.

The process of PriSampler**.** Figure 6.8 shows the process of our defense — PriSampler. It consists of two phases: generating hyperplanes and sampling via hyperplanes. In phase I, our method aims to find a latent space in terms of a sensitive property from the diffusion model. To achieve this goal, we leverage a linear SVM to learn the hyperplane of the sensitive property. To be more specific, given a diffusion model, we can get many different types of generated samples from different diffusion steps. As shown in the left part of Figure 6.8, starting from Gaussian noise sample $x_T \sim \mathcal{N}(0, \mathrm{I})$, the diffusion model can produce the sample x_t at intermediate diffusion step t and the final sample x_0 at the $t = 0$ step. The final sample x_0 is also the sample that we finally use. Then, the samples $X_0 = \{x_0^{(0)}, x_0^{(1)}, ..., x_0^{(n)}\}$ are inputted to a property prediction classifier and the corresponding prediction scores can be obtained, i.e. $S = \{s^{(0)}, s^{(1)}, ..., s^{(n)}\}$. Instead of using X_0, we pair X_t and S. The data samples (X_t, S) will be used for training a linear SVM. A hyperplane corresponding to this property can be obtained from the well-trained SVM.

In phase II, our method aims to sample via the learned hyperplane. As shown in the right part of Figure 6.8, our method manipulate samples at the t diffusion step. To be specific, given Gaussian noise sample $x_T \sim \mathcal{N}(0, \mathrm{I})$, we can get the sample x_t. At the t diffusion step, we change the sampling direction via the learned hyperplane, and the samples with and without the corresponding sensitive property can be obtained. In the remaining diffusion step, the samples will continue to synthesize in the specific sensitive property latent space. Finally, samples with the balanced

sensitive property can be generated. For different numbers of sensitive properties, the generated samples are calculated as follows.

Single property. Given a hyperplane $\mathbf{h}$ obtained in phase I, and a sample x_t at step t, we can get x'_t:

$$x'_t = x_t + \alpha \mathbf{h}. \tag{6.1}$$

α is a hyperparameter, and we take a value $\alpha > 0$ means that a positive sample $x_t^{'+}$ is obtained and it has this property. A value $\alpha < 0$ means a negative sample $x_t^{'-}$ is obtained and does not have this property. In this work, depending on different base samplers and diffusion models, we choose different α. We provide the details in Table 6.4.

Multiple properties. When there are multiple sensitive properties, the key idea is that we manipulate one property while keeping others unchanged. That is, we need to find a new hyperplane that is orthogonal to other hyperplanes. Take two properties as an example, we first manipulate the first one and manipulate the second condition on the first one. In this way, we can get generated samples with balanced properties. To be more specific, given two hyperplanes $\mathbf{h_1}$ and $\mathbf{h_2}$ obtained in phase I and a sample x_t. We first get a new hyperplane:

$$\mathbf{h'_2} = \mathbf{h_1} - (\mathbf{h_1^T h_2})\mathbf{h_2}, \tag{6.2}$$

where $(\mathbf{h_1^T h_2})\mathbf{h_2}$ is the projection of $\mathbf{h}_1$ onto $\mathbf{h}_2$. The new hyperplane $\mathbf{h'_2}$ equals the vector difference between $\mathbf{h}_1$ and the projection of $\mathbf{h}_1$ onto $\mathbf{h}_2$. Therefore, $\mathbf{h'_2}$ is orthogonal to $\mathbf{h_1}$. Put another way, $\mathbf{h'_2}$ can achieve that the second property is changed without impacting the first property. Then, based on Equation 6.1, we can get x'_t through x_t and $\mathbf{h_1}$. Given x'_t and $\mathbf{h'_2}$, we can obtain x''_t. Here, we require that the hyperplanes of multiple properties are independent, i.e. they are not in the same space. Otherwise, it is hard to find a hyperplane to guarantee that manipulating samples in this hyperplane does not affect the others. As introduced in Section 2.1.3, there are two types of samplings for diffusion models: stochastic sampling and deterministic sampling. Therefore, we implement our method on the PC sampler and the DPM sampler. In the following, we provide the implementation details of this algorithm.

Implementation details of PriSampler**.** Specifically, in Algorithm 3, the function `GeneratingHyperplanes` is utilized to learn the hyperplanes of properties from a given diffusion model, which corresponds to Phase I of Figure 6.8, while the function `sampling` is utilized to synthesize new samples in the learned hyperplanes, which corresponds to Phase II of Figure 6.8. For a single property, we use the function `samplingSingle` to synthesize samples while the function `samplingMulti` is used to generate samples for the cases of multiple properties.

In our defense, we consider two types of sampling: stochastic sampling — the PC sampler, and deterministic sampling — the DPM sampler. Therefore, the base sampler $q_\theta(x_{t-1}|x_t)$ of Algorithm 3 in the concrete implementation is the PC sampler and the DPM sampler, respectively. Note that the PC sampler consists of a predictor and a correcter in their original paper [SSDK+21]. We only use the predictor of the

Algorithm 3: The PriSampler Algorithm

Input: a frozen pre-trained diffusion model: $\mathcal{G}_\theta$; a base sampler: $q_\theta(x_{t-1}|x_t)$; the trained property classifier of the private property s_i: $\mathcal{P}_{s_i}$; the number of generated samples: m; the number of hyperplanes: k;
Output: Generated samples: X_{gen}

def `generatingHyperplanes`($\mathcal{G}_\theta, \mathcal{P}_{s_i}, q_\theta, k$)**:**
  H= [];
  for $i = 1$ **to** k **do**
    Sample n samples X and n intermediate samples X_t from $\mathcal{G}_\theta$ using a base sampler $q_\theta(x_{t-1}|x_t)$;
    $S \leftarrow \mathcal{P}_{s_i}(X)$; ▷ get prediction scores.
    $\mathcal{L} \leftarrow$ trainLinearSVM(X_t, S);
    h $\leftarrow$ getHyperplanes($\mathcal{L}$);
    H.append(**h**);
  return H

def `sampling`(H, m, q_θ)**:**
  X_{gen} = [];
  while $i < \lceil \frac{m}{k+1} \rceil$ **do**
    initial sample $x_T \sim \mathcal{N}(0, \mathrm{I})$;
    if $k == 1$ **then**
      X_{gen}.append(samplingSingle(H,q_θ,x_T)); ▷ For single property.
    if $k > 1$ **then**
      X_{gen}.append(samplingMulti(H,q_θ,x_T)); ▷ For multiple properties.
  return X_{gen}

H $\leftarrow$ generatingHyperplanes($\mathcal{G}$, $\mathcal{P}_{s_i}$,q_θ, k);
$X_{gen} \leftarrow$ sampling(H,m,q_θ);
return X_{gen}

Algorithm 4: The samplingSingle Algorithm

def `samplingSingle`(H, q_θ, x_T)**:**
  for $i = T - 1$ **to** 0 **do**
    if $i > t$ **then**
      $x_{i-1} \leftarrow q_\theta(x_{i-1}|x_i)$;
      ▷ Denoise when i > t.
    if $i == t$ **then**
      $x_i'^{+}, x_i'^{-} \leftarrow$ getSamples($H[0], x_i$);
      ▷ At the t step, get samples in the corresponding hyperplane by Equation 6.1.
    if $i <= t$ **then**
      ▷ Continue denoising when i <= t.
      $x_{i-1}'^{+} \leftarrow q_\theta(x_{i-1}'^{+}|x_i'^{+})$;
      $x_{i-1}'^{-} \leftarrow q_\theta(x_{i-1}'^{-}|x_i'^{-})$;
  return $x_0'^{+}, x_0'^{-}$

PC sampler when $i <= t$, in order to avoid the change of the sampling direction by the correcter.

Algorithm 5: The samplingMulti Algorithm

def samplingMulti(H, q_θ, x_T):
  for $i = T-1$ **to** 0 **do**
    if $i > t$ **then**
      $x_{i-1} \leftarrow q_\theta(x_{i-1}|x_i)$;
    if $i == t$ **then**
      $x_i^{'+}, x_i^{'-} \leftarrow$ getSamples($H[0], x_i$);
      ▷ get samples in the $H[0]$ by Equation 6.1.
      $h_2^{'} \leftarrow$ getCondHyper($H[0], H[1]$);
      ▷ get a new hyperplane by Equation 6.2.
      $x_i^{''++}, x_i^{''+-} \leftarrow$ getSamples($h_2^{'}, x_i^{'+}$);
      ▷ get samples in the $h_2^{'}$ by Equation 6.1.
      $x_i^{''-+}, x_i^{''--} \leftarrow$ getSamples($h_2^{'}, x_i^{'-}$);
      ▷ get samples in the $h_2^{'}$ by Equation 6.1.
    if $i <= t$ **then**
      $x_{i-1}^{''++} \leftarrow q_\theta(x_{i-1}^{''++}|x_i^{''++})$;
      $x_{i-1}^{''+-} \leftarrow q_\theta(x_{i-1}^{''+-}|x_i^{''+-})$;
      $x_{i-1}^{''-+} \leftarrow q_\theta(x_{i-1}^{''-+}|x_i^{''-+})$;
      $x_{i-1}^{''--} \leftarrow q_\theta(x_{i-1}^{''--}|x_i^{''--})$;
  return $x_0^{''++}, x_0^{''+-}, x_0^{''-+}, x_0^{''--}$

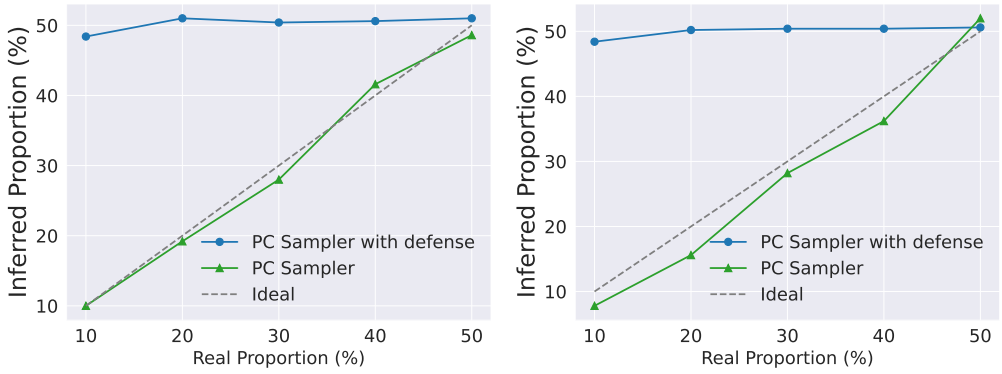
6.5.4 Experimental Setups

Experimental setups. We apply our method on two base samplers: one PC sampler for stochastic sampling and one DPM sampler for deterministic sampling. We directly use trained diffusion models from Section 6.3.3. In our defense, we use the library Sklearn to implement Linear SVM. We directly use trained property classifiers in Section 6.3.3 to predict scores S. We choose different diffusion steps for different samplers of diffusion models to manipulate. We summarize them in Table 6.4. As discussed in Section 6.3.3, 500 generated samples for stochastic sampling and 50,000 generated samples for deterministic sampling are used for computing FID values.

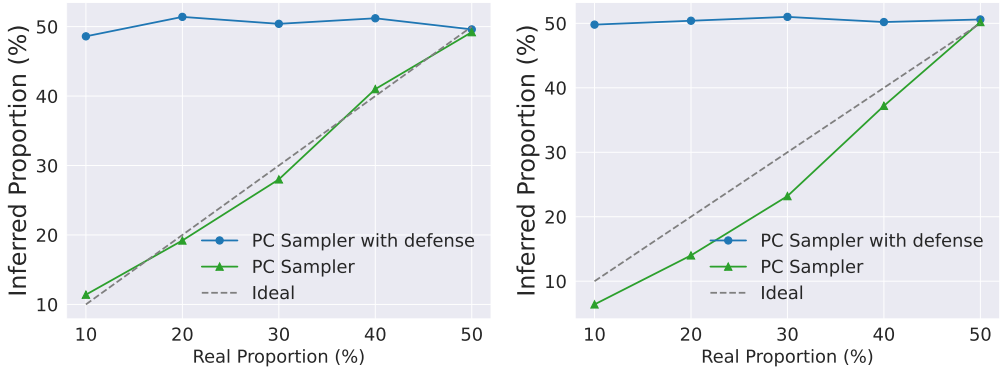
Hyperparameters of PriSampler. As introduced in Section 6.5.3, our method PriSampler has two hyperparameters: α and t. α controls the distance of the desired samples $x_t^{'}$ from an intermediate sample x_t. t is the diffusion step in which we manipulate an intermediate sample in the t step. Table 6.4 shows the hyperparameters α and t used for different samplers and diffusion models. For base samplers, the total number of sampling steps is 40 for the DPM sampler and 1,000 for the PC sampler. In terms of the base sampler — DPM sampler, we set its hyperparameter 'dpm_solver_method' as 'singlestep', and 'dpm_solver_order' as '3'. Therefore, for PriSampler applied to the DPM sampler, we set α and t as 50 and 6. Note, here, $t = 6$ is the index of diffusion steps rather than actual diffusion steps, because the step size of the DPM sampler is equal to 3, i.e. 'dpm_solver_order' = '3'. In terms of the base sampler — PC sampler, we set its hyperparameter 'predictor' as 'ReverseDiffusionPredictor', and 'corrector' as 'LangevinCorrector'. For PriSampler applied to the PC sampler, we choose different α and t for different diffusion models, as

Table 6.4: Hyperparameters (α, t) of `PriSampler` for different samplers and models.

Sampler	Model	CelebA-1k-10		CelebA-1k-20		CelebA-1k-30		CelebA-1k-40		CelebA-1k-50	
		α	t	α	t	α	t	α	t	α	t
PC	DDPM	150	699	150	699	150	699	140	699	140	699
	VPSDE	220	699	150	699	170	699	150	699	140	699
DPM	DDPM	50	6	50	6	50	6	50	6	50	6
	VPSDE	50	6	50	6	50	6	50	6	50	6
Sampler	Model	CelebA-50k-10		CelebA-50k-20		CelebA-50k-30		CelebA-50k-40		CelebA-50k-50	
		α	t	α	t	α	t	α	t	α	t
PC	SMLD	40	500	40	500	40	500	40	500	40	500
	VESDE	25	549	25	549	25	549	15	549	10	549



(a) DDPM. (b) SMLD.



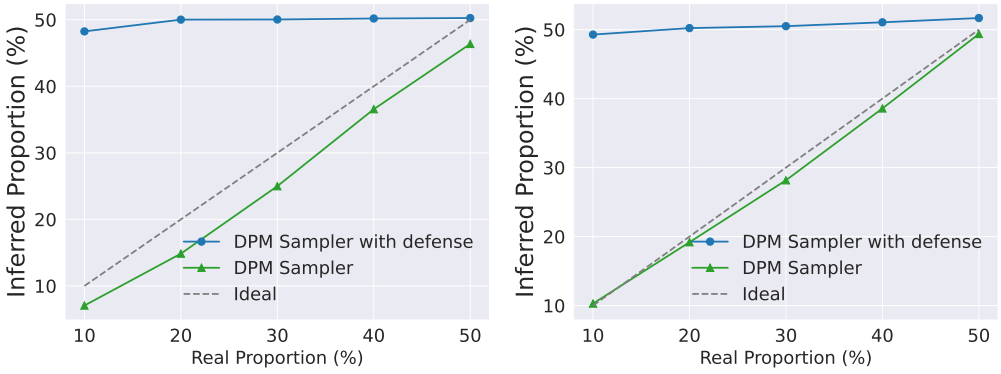
(c) VPSDE. (d) VESDE.

Figure 6.9: Defense performance on the PC sampler.

shown in Table 6.4. This is because the PC sampler is stochastic sampling where fresh noise will be added in the sampling process, which may affect the generated samples in the protected property space. Therefore, we adjust α and t to achieve the desired proportion.

6.5.5 Defense Results

Defense performance on different samplers. Figure 6.9 and Figure 6.10 present our defense performance to protect the sensitive property `male` for the PC sampler and the DPM sampler, respectively. Overall, our defense can achieve excellent performance. Even if the real proportion is 10%, our method can make adversaries get



(a) DDPM. (b) VPSDE.

Figure 6.10: Defense performance on the DPM sampler.

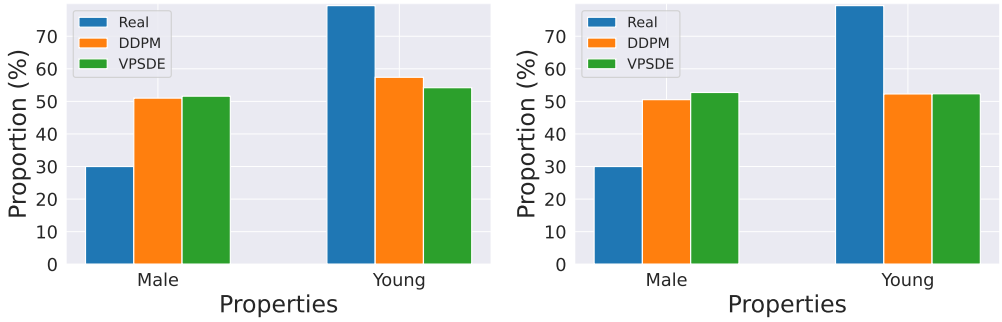
an inferred proportion of 50%.

Table 6.5: Defense performances on a single property. Desi. Prop.: Desired Proportion.

Model	Real Prop. (%)	Desi. Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID	Model	Real Prop. (%)	Desi. Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID
PC Sampler											
DDPM	10	50	48.40	1.60	56.35	VPSDE	10	50	48.60	1.40	58.72
	20	50	51.00	1.00	46.80		20	50	51.40	1.40	41.26
	30	50	50.40	0.40	40.00		30	50	50.40	0.40	38.89
	40	50	50.60	0.60	48.04		40	50	51.20	1.20	51.61
	50	50	51.00	1.00	49.58		50	50	49.60	0.40	42.82
SMLD	10	50	48.40	1.60	38.40	VESDE	10	50	49.80	0.20	61.32
	20	50	50.20	0.20	44.68		20	50	50.40	0.40	68.95
	30	50	50.40	0.40	45.52		30	50	51.00	1.00	57.47
	40	50	50.40	0.40	45.17		40	50	50.20	0.20	77.57
	50	50	50.60	0.60	46.95		50	50	50.60	0.60	49.75
DPM Sampler											
DDPM	10	50	48.25	1.75	44.70	VPSDE	10	50	49.28	0.72	52.44
	20	50	50.02	0.02	47.87		20	50	50.22	0.22	52.78
	30	50	50.04	0.04	44.41		30	50	50.50	0.50	50.66
	40	50	50.19	0.19	45.90		40	50	51.06	1.06	53.00
	50	50	50.26	0.26	47.68		50	50	51.68	1.68	45.33

Table 6.5 shows the corresponding quantitative results. Here, the absolute difference is the absolute difference between the inferred proportion $\hat{p}_{s_i}$ and desired proportion $\tilde{p}_{s_i}$, i.e. $|\hat{p}_{s_i} - \tilde{p}_{s_i}|$. The best defense performance can be seen at SMLD trained on CelebA-50k-20% for the PC sampler and DDPM trained on CelebA-1k-20% for the DPM sampler. Their absolute differences are 0.2% and 0.02%, respectively. The worst defense performance is 1.6% for the PC sampler for DDPM trained on CelebA-1k-10%, and 1.68% for the DPM sampler for VPSDE trained on CelebA-1k-10%. Table 6.7 summarizes the results for each sampler among different diffusion models. For the single property `male`, the average absolute difference is 0.75% for the PC sampler and 0.64% for the DPM sampler. It indicates that our defense method can control the error of an inferred proportion below 1.00%.

Defense performance on different diffusion models. Figure 6.9 and Figure 6.10



(a) PC sampler. (b) DPM sampler.

Figure 6.11: Defense performance for multiple properties.

present our defense performance on four types of diffusion models. Similarly, we can observe that the inferred proportions almost remain 50% for diffusion models trained on different datasets. It means our method can be effectively applied to these different diffusion models.

Defense performance on more than one property. Figure 6.11 shows the defense performance to protect two sensitive properties `male` and `young`. Here, we choose the models trained on CelebA-1k-30% as target models. That is, the real proportion of the property `male` is 30%. The corresponding real proportion of the private property `young` is 79.4%. Overall, we can observe that the inferred proportions of both properties are about 50%, no matter the larger proportion or the smaller proportion.

Table 6.6 shows the corresponding quantitative results. Table 6.7 summarizes the result for `male+young` properties on both samplers. We can see that the average absolute difference is 3.55% for the PC sampler and 1.97% for the DPM sampler. We also note that the absolution difference on two properties is larger than that on the single property. One of the reasons is that these properties are entangled together in diffusion models. As a result, it might lead the hyperplane not to completely separate these properties. We will take this as our future work to design disentangled generative algorithms for diffusion models from the perspective of the training process to improve our defense performance on multiple properties.

Table 6.6: Defense performances on multiple properties.

Model		Real Prop. (%)	Desi. Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)		Real Prop. (%)	Desi. Prop. (%)	Inferred Prop. (%)	Abs. Diff. (%)	FID
PC sampler											
DDPM	Male	30	50	51.00	1.00	Young	79.40	50	57.40	7.40	48.74
VPSDE	Male	30	50	51.60	1.60	Young	79.40	50	54.20	4.20	45.95
DPM sampler											
DDPM	Male	30	50	50.53	0.53	Young	79.40	50	52.28	2.28	40.25
VPSDE	Male	30	50	52.73	2.73	Young	79.40	50	52.33	2.33	42.29

Defense performance on different numbers of generated samples. Figure 6.12 shows the defense performance on different numbers of generated samples. Here, we choose the DDPM model trained on CelebA-1k-30% as the target model. The

Table 6.7: Summary of defense performances. Here, we report the average absolution difference (with standard deviation in parentheses) and the best and worst absolution difference.

Property	Sampler	Average (%)	Best (%)	Worst (%)
Male	PC	0.75 (0.48)	0.20	1.60
	DPM	0.64 (0.65)	0.02	1.75
Male+Young	PC	3.55 (2.92)	1.00	7.40
	DPM	1.97 (0.98)	0.53	2.73

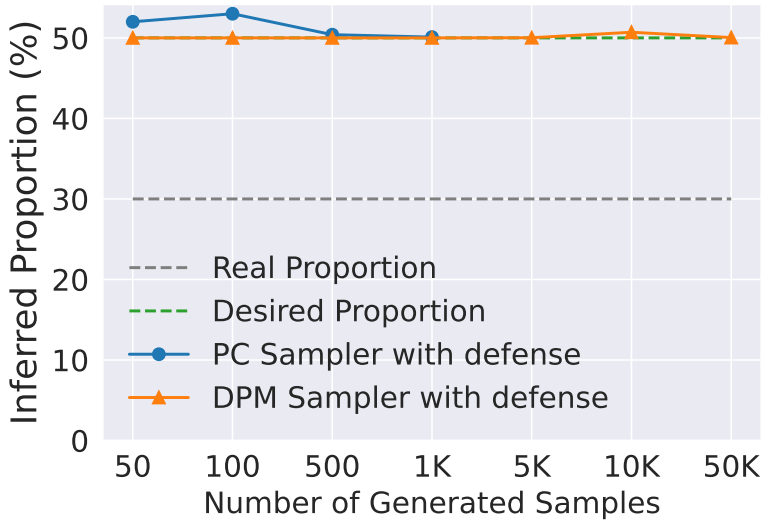


Figure 6.12: Defense performance on the number of generated samples.

sensitive property is `male`. We can clearly see that both types of samplers can provide good protection for the property `male` even if model owners only release only as few as 50 samples. Although the PC sampler shows a slight fluctuation in the phase of releasing a few samples, it is gradually stable after 500 samples. The DPM sampler extremely stabilizes no matter how many samples are released. One of the reason might be that random noise added during the sampling process for the PC sampler have some effects on generated samples because the PC sampler belongs to stochastic sampling while the DPM sampler belongs to deterministic sampling.

Defense performance on different diffusion steps. Figure 6.13 shows defense performance on different diffusion steps. Here, the target model is DDPM trained on CelebA-1k-30% and we use the PC sampler and the total number of sampling steps is 1,000, and the sensitive property is `male`. The blue line and the left axis show the inferred proportion while the red line and the right axis present the corresponding FID values. Generated samples in the 0 diffusion step are pure Gaussian noise while generated samples in the 999 step are realistic samples. Overall, we can see that defense performance and model utility in the latter stage of diffusion steps show better than that of the former stage. Generally, choosing late middle diffusion steps can obtain a good balance in defense performance, FID values, and the meaningfulness of generated images.

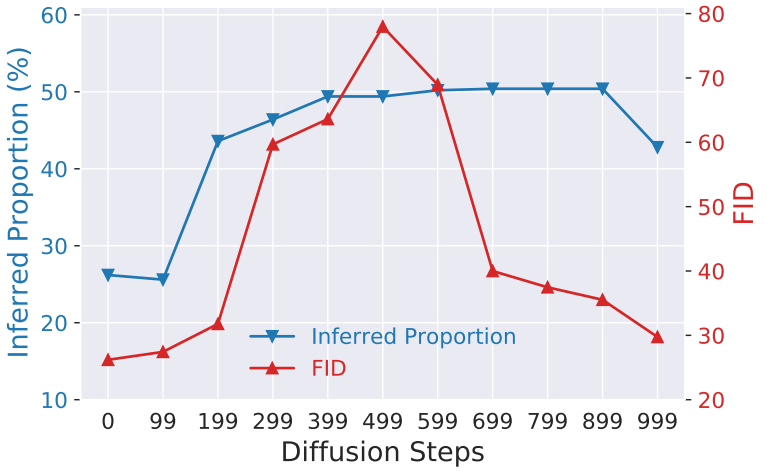


Figure 6.13: Defense performance on different diffusion steps.

6.5.6 Comparison with Differential Privacy

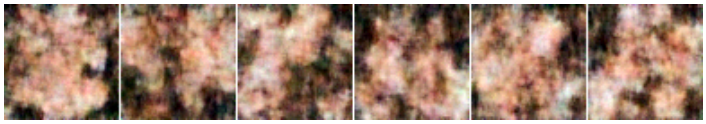
Differential privacy [ACG+16, Dwo08] is a common measure for defending against privacy attacks. In this subsection, we explore the feasibility of differential privacy to defend against property inference attacks. Furthermore, we make a comparison with our defense method PriSampler.

We use differentially private diffusion models (DPDMs) proposed by Dockhorn et al. [DCVK22], because they are the first to apply differentially private stochastic gradient descent (DPSGD) [ACG+16] to diffusion models and can generate meaningful images. We adopt their suggested hyperparameters to train DPDMs. We set the number of epochs and batch sizes as 100, and 128 respectively. The image size is fixed at 64, and we choose different sizes of training sets, i.e. CelebA-1k-30% and CelebA-50k-30%, and different privacy budgets ϵ, i.e. $\epsilon = 10$ and $\epsilon = 50$. We fix δ as 10^{-6} for all models. Here, we synthesize samples by stochastic sampling because DPDMs [DCVK22] analyze that it can obtain better FID values under differential privacy conditions.

Table 6.8 presents the comparison between our method and DPDM. Figure 6.14 visually shows synthetic samples. For the CelebA-1k-30% dataset, DPDM almost cannot generate meaningful images, which leads to an FID value of 446.35. In contrast, our method can achieve a 40.00 FID value and the inferred proportion is 50.50%. Figure 6.14(b) also shows the good quality of synthetic samples for our method PriSampler. For the CelebA-50k-30% dataset, we can clearly see that the generated samples from DPDM only have a vague shape of the human face. Even if we increase the privacy budget ϵ from 10 to 50, the synthetic human face samples still are distorted, although we can see that FID decreases from 121.56 to 103.64. Here, note that $\epsilon = 10$ is usually considered as low amounts of privacy. We also observe that the inferred proportion for DPDM under $\epsilon = 10$ is 44.00%, while that for DPDM under $\epsilon = 50$ is 24.20%. It indicates that DPDM under smaller privacy budgets can disguise the real proportion of certain properties to some extent. However, the quality of the generated samples is too vague. In contrast, our method can still synthesize meaningful samples with a balanced proportion.

Table 6.8: Comparison between PriSampler and DPDM. DDPM* means PriSampler is applied to the DDPM model. SMLD* means PriSampler is applied to the SMLD model.

	CelebA-1k-30%		CelebA-50k-30%		
	DPDM ($\epsilon = 10$)	DDPM*	DPDM ($\epsilon = 10$)	DPDM ($\epsilon = 50$)	SMLD*
FID	446.35	40.00	121.56	103.64	45.52
Inferred Prop.	100.00	50.40	44.00	24.20	50.40



(a) $\epsilon = 10$, DPDM trained on CelebA-1k-30%.

(b) PriSampler for DDPM trained on CelebA-1k-30%.

(c) $\epsilon = 10$, DPDM trained on CelebA-50k-30%.

(d) $\epsilon = 50$, DPDM trained on CelebA-50k-30%.

(e) PriSampler for SMLD trained on CelebA-50k-30%.

Figure 6.14: Visualization of synthetic samples under the defense DPDM and PriSampler.

6.6 Discussion

Our method PriSampler aims to navigate a sampler in the property space and is operated in the sampling process. Thus, it is a training-free method. Furthermore, it is a model-agnostic and can be used as a plug-in for a wide range of diffusion models. In this section, we discuss limitations and future work.

Model utility. Although our method PriSampler can guarantee the defense performance, i.e. achieving the desired proportions that model owners wish, it will sacrifice model utility to some extent. Nevertheless, we take the first step to protect diffusion models from property inference attacks. Furthermore, our defense method is still promising and competitive, compared to diffusion models trained with differential privacy.

Non-binary property. Our defense method PriSampler in this work mainly considers binary property which has two values. For instance, the value of the binary property `gender` has 0 (male) and 1 (female). When the property is non-binary, i.e. the value of a property has multiple categories or is continuous, our method PriSampler can also be applied by transforming this property as a binary property.

Entangled properties. We observe that when our defense method is applied to protect multiple sensitive properties, the defense performance, i.e. the average absolute difference, becomes slightly worse, compared to that applied to the single property. This might be because these properties are entangled together due to the training mechanism of existing diffusion models. As a result, it is difficult to find an ideal hyperplane to completely differentiate them. In the future, we intend to design diffusion models with disentangled properties, which aim to separate entangled properties as large as possible in the training process. In that way, our defense method can further improve the protection performance for diffusion models.

Membership inference. Membership inference and property inference are two main types of privacy attacks, but their attack goals are different. Membership inference involves the privacy of individual training sample of a training set while property inference involves the privacy of global properties of a training set. In the future, we plan to study the relationship between two types of privacy attacks and provide a holistic defense measure.

Attacks via weights. Our attack method only utilizes the synthetic data from a diffusion model to mount property inference attacks. Prior work on classification models [GWY+18] has proposed to infer the sensitive properties by the weights of a full-connected neural network. Therefore, we plan to investigate the feasibility of property inference attacks by utilizing the weights of diffusion models themselves.

6.7 Related Work

Diffusion models. Diffusion models [SDWMG15, HJA20] have recently drawn immense attention to academia and industry due to their high success in synthesizing realistic images. Subsequently, various methods [SE19, SSDK+21, SME21, KAAL22, LZB+22, LZB+22] are proposed to further improve the performance of diffusion models from the perspective of the training process, sampling mechanisms. Beyond image synthesis, they have been applied to a variety of novel applications, such as image restoration [SCC+22, WYZ22], super-resolution [HSC+22, SHC+23]. However, these works focus on improving the generative performance of diffusion models. In this chapter, considering the increasing popularity of diffusion models, we study diffusion models from the viewpoint of privacy.

Property inference attacks. Property inference attacks allow adversaries to infer global sensitive information of the training set from a machine learning model. They are firstly studied by Ateniese et al. [AMS+15] on simple machine learning

models, such as SVM and Hidden Markov Models. Since then, there are a more increasing number of works focusing on property inference in neural network models, such as fully-connected neural networks [GWY+18], convolution neural networks [SE22, MGC22, CAO+23], generative adversarial networks[ZCSZ22], graph neural networks [ZCB+22], and federated learning models [MSDCS19]. However, these works mainly focus on attacks, and their attack methods heavily rely on shadow models which require a large amount of computation. Furthermore, property inference attacks on emerging diffusion models have not yet been extensively studied. In this chapter, we take the first step to explore property inference attacks against various types of diffusion models under more realistic attack scenarios and affordable attack costs: the adversaries perform the inference only utilizing synthetic data. More importantly, we propose a set of effective defense measures to safeguard the sensitive properties of diffusion models.

There are several works studying privacy attacks against diffusion models through the lens of membership inference attacks [WYL+22, CHN+23, HP23a, MMY23, DKW+23, ZCGF23]. However, membership inference attacks aim to infer whether a sample was used for training a machine learning model and focus more on the privacy of the individual training sample of the training set. In contrast, this work endeavors to study property inference attacks which aim to infer the globally sensitive information of the training set used for diffusion models.

6.8 Conclusion

In this chapter, we have presented the first study about property inference of diffusion models. Under the property inference attack which only utilizes the synthetic data, we have investigated the property inference risks on four different types of diffusion models and two different types of sampling mechanisms (in total twenty diffusion models and three samplers). Our extensive empirical analysis has shown that various diffusion models and their samplers are vulnerable to property inference attacks. For instance, as few as 500 generated samples can precisely infer the real proportion of a property. More severely, better performance of diffusion models can lead to a more accurate estimation of property inference, which indicates that we should consider privacy concerns when consistently pursuing the generation performance of a diffusion model. We have also shown the property inference attack is still effective at inferring off-the-shelf pre-trained diffusion models in reality. We also developed a model-agnostic plug-in defense method PriSampler and demonstrated its effectiveness with various different types of samplers and diffusion models. Our PriSampler further shows significant performance in model utility and defense performance, when compared with diffusion models trained with differential privacy.

We have also identified several directions for future work, including developing attack methods via weights, designing diffusion models with disentangled properties, and constructing a holistic defense method by exploring the relationship between property inference and membership inference.

Part III

Applications of Generative Models

Chapter 7

Out-of-Distribution Attacks via Generative Adversarial Networks

In addition to studying model privacy and data privacy in generative models, it is essential to unlock the potential of generative models and innovatively apply them to investigate security risks in discriminative models. In this chapter, we propose a novel out-of-distribution (OOD) attack: leveraging any pre-trained generative adversarial networks (GANs), an adversary aims to fool a classification model and make the model misclassify a sample from GANs as a pre-specified target class. Specifically, we introduce a targeted attack framework through GANs for white-box and black-box scenarios. Our framework casts this problem as an optimization problem and a family of attack methods are developed. Extensive experimental results show that our methods can achieve competitive performance, even compared with several state-of-the-art adversarial example attacks. Finally, our methods can evade several widely-used and the latest OOD detection.

7.1 Introduction

Recent years have witnessed significant progress in machine learning (ML), ranging from computer vision [GPAM+14, HZRS16] to natural language processing [DCLT19, BMR+20]. The success of ML has also driven technology companies to deploy various ML-based applications in real-world scenarios, including safety-critical applications such as self-driving cars [FW15, BDTD+16]. However, ML models face various security threats in the open world [PCYJ17, CXC+19, FLL+22].

There has been a flurry of works revealing that ML models are vulnerable to adversarial example attacks [BCM+13, SZS+14, MMS+18, BRB18, CJW20]. Given a correctly classified example, an adversary can make an adversarial example by adding imperceptible perturbations, which causes the ML model to change its prediction result [SZS+14]. From the perspective of human vision, these imperceptible perturbations totally do not affect the category of the example, but the ML model has made a different decision. In other words, ML models are easy to be fooled by adversarial perturbations and are not robust in knowing what they know. In addition, another line of research studies demonstrates that for samples that are far from the training data or are completely unrecognizable to human beings (e.g. noises), ML

models also misclassify them as a particular class with high confidence. To put it another way, *ML models do not know when they do not know* [NYC15, HAB19, MH20].

Indeed, when ML models are deployed in the real world, any type of input sample could occur and these models face the risk fooled by adversarial samples. Considering the prevalence and availability of pre-trained GANs, this motivates us to think about a new potential threat: generated samples from a well-trained GAN might be utilized by adversaries to fool ML models.

In this chapter, we study a novel *out-of-distribution* (OOD) attack in which an adversary aims to craft a completely different sample (e.g. cartoon face) to deceive an ML model into recognizing a certain class that the ML model has learned (e.g. airplane). Concretely, leveraging any off-the-shelf pre-trained generative adversarial networks (GANs), we propose a novel targeted attack framework, which leads ML models to make a particular prediction for inputs from a GAN. We highlight that *the pre-trained GAN can be any unconditional GAN trained on any dataset*. For consistency with prior works [HG17, SBS+19], we refer to these generated examples as OOD examples and these examples have a different distribution of the training set of a victim ML model.

In general, ML models make arbitrary predictions for OOD examples because they indeed do not know OOD examples. Taking advantage of any GAN, our attack framework attempts to construct generated samples so that the victim ML model misclassifies them as a particular class that the adversary wishes. More broadly, our attack fools ML models into knowing what they actually do not know. Although a straightforward way is to utilize existing adversarial example attacks [SZS+14] where adversarial perturbations are added to OOD examples, our GAN-based attack methods are not restricted by the magnitude of perturbations. Most importantly, with the availability of various pre-trained GAN models, our attack framework provides a new angle to find diverse adversarial examples that state-of-the-art defense mechanisms do not consider.

Technically, we formulate our attack framework as an optimization problem: the attack framework aims to find a generated example from a pre-trained GAN model by minimizing the distance between its victim model's prediction and the pre-specified target class that the adversary defines. Under the white-box and black-box attack scenarios, gradient-based and non-gradient based optimization methods are proposed to obtain the generated sample. Extensive experimental evaluations demonstrate that our attack methods are highly effective, achieving over 97% average attack success rate on the white-box scenario and at least 62% average attack success rate on the black-box scenario (see Section 7.5.1).

Moreover, we compare our attack methods with the five state-of-the-art adversarial example attacks, i.e. C&W [CW17b] and PGD [MMS+18] in the white-box scenario, ZOO [CZS+17], DBA [BRB18] and HSJA [CJW20] in the black-box scenario. Extensive experimental evaluations show that the performance of our attack methods is still competitive (see Section 7.5.3). Here, we underline that the mechanism of our attack methods is totally different from that of adversarial example attacks, that is, the OOD adversarial example is generated from a GAN rather than an image with

perturbations, although both attacks have the same attack objective — fooling ML models. We further evaluate our attack methods by investigating whether OOD detection techniques can detect these OOD adversarial examples. Experimental results demonstrate that our attack methods can evade three state-of-the-art OOD detection techniques, i.e. ODIN [LLS18], OE [HMD19] and ATOM [CLW+21] (see Section 7.5.3). We also analyze the effects of different types of loss functions and optimization methods on attack performance (see Section 7.5.4).

Finally, we want to remark that our proposed methods not only can be applied as security attacks on ML models. Being a supplementary measure, they also can be utilized to conduct the white-box and black-box tests when deploying ML models in the real world.

7.2 Related Work

Adversarial example attacks. Depending on whether the adversary has access to the whole victim model, adversarial example attacks generally can be categorized into two types: white-box and black-box. White-box adversarial example attacks generally utilize gradient information of the victim model to perturb examples [BCM+13, SZS+14, GSS15, CW17b, MMS+18]. Goodfellow et al. propose a fast gradient sign method to generate adversarial examples under the ℓ_∞ norm. This attack method only involves a single-step gradient update [GSS15]. Furthermore, Madry et al. propose an iterative method — Projected Gradient Descent, to generate more powerful adversarial examples, and an adversarial training method is also proposed [MMS+18]. Unlike previous works, the C&W attack method utilizes new objective functions and converts the box constrained optimization problem into an unconstrained problem to construct adversarial examples [CW17b].

Because it is impossible to directly compute the gradients based on the victim model under the black-box scenario, various gradient estimation attack methods are proposed. The ZOO attack method chooses to estimate the gradients through monitoring the changes of prediction confidence [CZS+17]. Chen et al. utilize the binary information at the decision boundary to estimate the gradient direction [CJW20]. Instead of making an estimate of gradient information, Brendel et al. propose an iterative black-box attack method by rejection sampling [BRB18]. *Unlike our proposed methods where examples are generated from any pre-trained GAN, these attack methods construct adversarial examples by adding adversarial perturbations.*

There exist several works generating adversarial examples with GANs. Baluja et al. propose an adversarial transformation network that is trained to directly produce adversarial examples [BF18]. Xiao et al. propose a conditional GAN where it first generates a perturbation and an adversarial example is constructed by adding the perturbation into the original sample [XLZ+18]. Similarly, Song et al. propose an auxiliary classifier GAN to model the class-conditional distribution of data samples and utilize it to generate samples [SSKE18]. However, these approaches require a tailored GAN when constructing adversarial examples. Additionally, extra data is required not only for a substitute model training in the black-box scenario but also

for a tailored GAN training, whereas these data have to have the same distribution of the training set of the victim model and may not be easily obtained by the adversary. *In contrast, our work proposes a targeted attack framework for both white-box and black-box scenarios and any off-the-shelf pre-trained GAN can be directly leveraged. Even in the black-box scenario, our methods do not require training a substitute model.*

Out-of-distribution detection. There are many works attempting to enhance the robustness of ML models in the open world through out-of-distribution (OOD) detection [YZLL21, SLH+21, FLL+22]. Hendrycks et al. propose a baseline of OOD detection by utilizing probabilities from softmax distributions. This method is based on the insight: compared to OOD examples, correctly classified examples tend to have a higher prediction probability [HG17]. Liang et al. further improve the performance of OOD detection by utilizing temperature scaling and adding small perturbations to the inputs, which results in a larger separation between in-distribution and out-of-distribution samples [LLS18]. Hendrycks et al. improve detection performance by leveraging an auxiliary dataset of outliers to train anomaly detectors [HMD19]. Chen et al. propose to carefully select informative outliers from an auxiliary OOD dataset and utilize adversarial training to further enhance detection performance [CLW+21]. We note that almost all works evaluate their detection performance on OOD datasets collected from the real world. Although Chen et al. present their detection performance on adversarial examples, these examples are constructed by adding adversarial perturbations [CLW+21]. *Our GAN-based attack methods have a different mechanism in generating OOD examples, which may provide supplementary testing means to thoroughly evaluate the performance of OOD detection.*

Generative adversarial networks. Goodfellow et al. first propose generative adversarial networks (GANs) in 2014 [GPAM+14]. Since then, various GAN models are proposed to generate more realistic and diverse images [RMC16, ACB17, MKKY18, KALL18, BDS19, KLA+20]. Martin et al. propose to utilize the Wasserstein distance to stabilize the training process to further improve the quality of images [ACB17]. Karras et al. propose a growing strategy to stabilize the training process, which allows GAN models to produce more realistic and high-quality images [KALL18]. The StyleGAN introduces a novel hierarchical latent style layer, which allows GAN models to produce diverse styles of images [KLA19]. With the emergence of GANs, we are no longer limited to collecting data on our own or utilizing existing datasets when needing images. *Benefiting from this advantage, in this chapter, we aim to directly find adversarial OOD examples from any pre-trained GAN, especially utilizing these state-of-the-art GANs.*

7.3 Methodology

7.3.1 Background

A typical unconditional GAN consists of a generator G and a discriminator D. During the training process, the generator G learns to synthesize an image $x_g = G(z)$, where a latent code $z \in \mathbb{R}^n$ is drawn from a prior distribution p_z, such

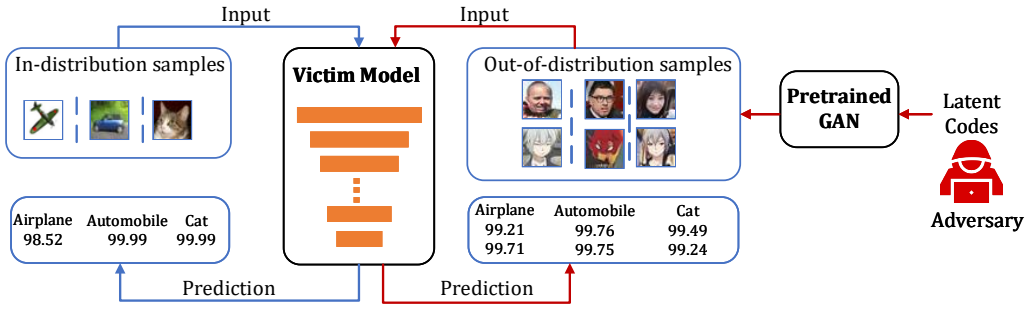


Figure 7.1: Attack overview. Our proposed attack constructs out-of-distribution samples by any pre-trained GAN to make a victim model classify them as certain classes that an adversary wishes. For instance, OOD samples, such as human faces or cartoon faces generated from GANs, are recognized as certain classes, such as airplanes, by the victim classifier trained on CIFAR-10.

as Gaussian distribution. The discriminator D is responsible for distinguishing between fake samples x_g generated by the G and real samples x_r from the training set X. Once the GAN model finishes training, only the generator G is used to produce a novel image through a latent code z, i.e. $x = G(z)$. We consider an m-class classifier $F(\cdot)$ as the victim model. It takes as input an image x and returns a full prediction $y = F(x)$. Here, the full prediction $y = F(x)$ refers to logits or probabilities. y_{max} refers to a predicted label.

In this chapter, we also call victim models without any defenses as *raw models*. In addition, if the distribution of examples is different from that of the training set of the victim model, we refer to these examples as *OOD examples*. For instance, an image of a horse is considered as an OOD example for a victim model trained on the cat/dog dataset. *Adversarial OOD examples* refer to OOD examples that are manipulated by an adversary, and they are classified into two types: restricted adversarial OOD examples and unrestricted adversarial OOD examples. *Restricted adversarial OOD examples* refer to OOD adversarial examples that are constructed based on norm-bounded perturbations, while *unrestricted adversarial OOD examples* refer to OOD adversarial examples from a pre-trained GAN model. When it is clear from the context, we interchangeably use OOD examples, adversarial OOD examples, restricted adversarial OOD examples, and unrestricted adversarial OOD examples.

7.3.2 Attack Overview

We provide a high-level description of our attack, as depicted in Figure 7.1. A victim model is trained on an in-distribution dataset. On the one hand, the victim model can correctly recognize in-distribution samples, such as airplane and cat, which is shown in the left part of Figure 7.1. On the other hand, human face images and cartoon face images that are completely different from these in-distribution samples can be all recognized as certain classes that the adversary wishes, as illustrated in the right of Figure 7.1. Our attack framework provides one approach to craft the generated samples that are misclassified as a pre-specified class by the victim model.

7.3.3 A Unified Attack Framework

Generally, attack goals can be categorized into two classes: untargeted and targeted. An untargeted attack aims to lead to a victim model's misclassification while a targeted attack aims to change a victim model's prediction to a pre-defined target class. In this chapter, we only consider the targeted attack for raw models, because any OOD example is regarded as a successful untargeted attack.

Specifically, the objective of our attack is to find a generated sample from a given GAN, which leads this sample to be classified as a pre-defined class by the victim model. Formally, we cast it as an optimization problem as follows:

$$z' = \arg\min_{z} \ \ell(F(G(z)), F(x_{ref})), \tag{7.1}$$

where $\ell(\cdot)$ is the loss function, x_{ref} is a reference sample from the target class t that the adversary wishes. By minimizing the distance of the victim model's outputs between the generated sample $F(G(z))$ and the reference sample $F(x_{ref})$, Equation 7.1 aims to find a latent code z' and causes the victim model to classify the generated image $x' = G(z')$ into the pre-specified class t.

Equation 7.1 is agnostic to any type of unconditional GAN. Thus, the generator G of any pre-trained GAN models can be utilized. In this work, we choose the cross-entropy loss and the StyleGAN [KLA19] as our loss function and the pre-trained GAN model to attack raw models. Therefore, Equation 7.1 can be reformulated as:

$$z' = \arg\min_{z} \ \ell_{CE}(F(G(z)), t) \tag{7.2}$$

Other types of loss functions can be utilized to further improve attack performance for models with defenses, which is detailed in Section 7.5.3. We leave the exploration of different GAN models as future work.

We consider two typical attack scenarios: white-box and black-box. In these attack scenarios, different optimization methods are proposed to solve the Equation 7.2.

White-box scenario. In this attack scenario, the adversary can obtain the whole model, including its weights and architecture. Therefore, the gradient information can be easily obtained for the adversary. Any gradient-based optimization methods can be used to solve the Equation 7.1. In our work, we apply the stochastic gradient descent (SGD) [RM51] for its computational efficacy. We discuss different types of optimizations in Section 7.5.4.

Black-box scenario. In this attack scenario, the adversary is only allowed to query the model and has access to the model's outputs. Gradient-based optimization methods cannot be applied due to the unavailability of the whole victim model. Here, we adopt Powell's conjugate direction method (Powell) [Pow64] to optimize Equation 7.1, because it does not require the gradient of the victim model. Specifically, Powell's method first creates a set of mutually conjugate directions for the latent code z and then finds the local minimum by line search along with these directions.

Algorithm 6: Attack procedure

Input: a victim model F, a target class t, a pre-trained GAN G, the maximum number of iterations K, the learning rate α, *flag*
Output: an unrestricted adversarial OOD example x

Initialization: draw a latent code z from the standard normal distribution
for $k = 0, 1, \ldots, K-1$ **do**
  if $F(G(z))_{max} == t$ **then**
    break ▷ Early stopping
  end if
  $\mathcal{L} = computeLoss(F(G(z)), t)$ ▷ Equation 7.2
  if *flag is white-box* **then**
    $z \leftarrow z - \alpha \nabla_z \mathcal{L}$ ▷ Stochastic gradient descent
  else if *flag is black-box* **then**
    $z \leftarrow$ Powell($\mathcal{L}$) ▷ Powell optimization
  end if
end for
return $x = G(z)$

The attack procedure for both the white-box and black-box scenarios is described in Algorithm 6.

7.4 Experiments

7.4.1 Datasets

We use five datasets in this chapter to conduct our experiments. These datasets can be further divided into three types: in-distribution, out-of-distribution, and auxiliary datasets. We refer to a dataset used to train a victim model as an in-distribution dataset, while an out-of-distribution dataset refers that its distribution is different from that of the training set of the victim model. Auxiliary datasets are also out-of-distribution datasets but they are commonly used in OOD detection to improve the robustness of machine learning models.

In-distribution datasets. We use CIFAR-10 [Kri09] and the German Traffic Sign Recognition Benchmark (GTSRB-43) [SSSI12] as in-distribution datasets to train victim models. CIFAR-10 has 10 classes and contains 50,000 training images and 10,000 test images, while GTSRB-43 has 43 classes of traffic signs and includes 39,209 training images and 12,630 test images.

Out-of-distribution datasets. We use FFHQ [KLA19] and iCartoonFace [ZZR+20] as out-of-distribution datasets. These datasets are used for GAN training. In addition, they are utilized to make adversarial examples when we compare our work with adversarial example attacks. FFHQ consists of 70,000 high-quality human face images and these images have a large amount of variation in the aspect of age, ethnicity, identity, and accessories. iCartoonFace contains 389,678 cartoon face images and these images are collected from 1,302 cartoon albums.

Auxiliary datasets. The ImageNet dataset [RDS+15] is used as the auxiliary dataset. It is widely used in various OOD detection methods to help improve the robustness of ML models. The ImageNet dataset consists of 1,281,167 diverse images and 1,000 classes.

7.4.2 Victim Models

We choose the WideResNet [ZK16] with depth 28 and widening factor of 4 and the DenseNet [HLVDMW17] with depth 100 and growth rate 12 as the architecture of victim models to be attacked, considering their excellent performance and widespread adoption in various application areas. They are also widely used in the research community of machine learning security [CW17a, TCBM20].

7.4.3 Evaluation Metrics

We use Attack Success Rate (ASR) to evaluate attack performance. The ASR is the ratio of success samples in all test samples. In our attack, an example is considered a success if it is recognized as a pre-specified class. With the aim to thoroughly evaluate attack methods and reveal the vulnerability of victim models, we further report average ASR, best ASR, and worst ASR from the perspective of classes of victim models.

The average ASR reports the mean of ASRs of all classes. The best ASR refers to the best ASR in all classes while the worst ASR reports the worst ASR in all classes. The best ASR indicates the attack success rate of the most vulnerable category in a victim model, which can be regarded as the most vulnerable point of the model. The worst ASR represents the probability of success for the least vulnerable class in a victim model. The worst ASR of 0 means that there exist classes that are harder to attack in the model. For computing efficiency, in our work, 10 test samples are used to compute the ASR of each class. Thus, we totally use 100 samples and 430 samples for CIFAR-10 and GTSRB-43, respectively. Note that in practice the attack is successful even if only one sample can fool the model.

7.4.4 Experimental Setups

In this chapter, we use the standard split of CIFAR-10 and GTSRB-43 to train victim models. As for OOD datasets, all samples in FFHQ are used for GAN training. Due to various image sizes in iCartoonFace, images whose size is equal to or larger than 128 are chosen. Consequently, we obtain 35,2459 images in total. Similarly, these images are used for GAN training. Samples that are utilized to make unrestricted adversarial OOD examples are randomly selected from the OOD datasets. All images in all datasets are resized to 32×32 and rescaled to [0, 1].

All victim models are trained with SGD optimizer and an initial learning rate of 0.1. For CIFAR-10, the number of training epochs is set to 100 and the learning rate is decayed by a factor of 0.1 at the 50th, 75th, and 90th epochs. For GTSRB-43, we set

Table 7.1: Performance of raw models on CIFAR-10 and GTSRB-43.

Dataset	Model	Accuracy
CIFAR-10	WideResNet	95.16%
	DenseNet	94.43%
GTSRB-43	WideResNet	96.46%
	DenseNet	96.08%

the number of training epochs to 20 and the learning rate is decayed by a factor of 0.1 at the 7th, 12th, and 17th epochs. For GAN models, StyleGAN is chosen due to its excellent performance on image generation and the suggested hyperparameters in its original publication are used for training [KLA19].

For our attack methods, in the white-box scenario, the learning rate of SGD and the maximum number of iterations is set as 0.1 and 5,000, respectively. Early stopping is allowed when OOD examples are found successfully. In the black-box scenario, the maximum number of queries is set as 25,000.

7.5 Evaluation

We first present our attack results for raw models. Next, we compare our methods with five state-of-the-art adversarial examples attacks. We further evaluate our attack performance on models with the protection of three representative defense measures. Finally, we explore diverse factors to better delineate the properties of our attack methods.

7.5.1 Attack Performance on Raw Models

Performance of raw models. Table 7.1 shows the performance of raw victim models trained on different datasets. All models achieve close to state-of-the-art performance. For instance, at least 94% accuracy and 96% accuracy can be seen on CIFAR-10 and GTSRB-43, respectively.

Results. Table 7.2 shows attack performance on raw models trained on various datasets for both white-box and black-box scenarios. Overall, all raw models on both scenarios are vulnerable to our proposed attack methods.

In the white-box scenario, the adversary can achieve an extraordinarily high average ASR among all victim models, ranging from 97.44% to 100.00%. For example, the attack method can achieve an average ASR of 100% on CIFAR-10, no matter which victim models are attacked. Even in terms of the worst ASR, the attack method still remains an attack success rate of 100% for CIFAR-10 and no less than 80% for GTSRB-43. This indicates that a victim model without any protection is easily fooled and our proposed method can always find a sample from the pre-trained GAN model and make the victim model recognized as a pre-specified class.

Table 7.2: Attack performance on raw models trained on various datasets. SD: standard deviation.

In-distribution Dataset	Victim Model	OOD Dataset	White-box			Black-box		
			Average ASR (SD) %	Best ASR %	Worst ASR %	Average ASR (SD) %	Best ASR %	Worst ASR %
CIFAR-10	WideResNet	FFHQ	100.00 (0.00)	100.00	100.00	90.00 (10.00)	100.00	70.00
		iCartoonFace	100.00 (0.00)	100.00	100.00	97.00 (4.58)	100.00	90.00
	DenseNet	FFHQ	100.00 (0.00)	100.00	100.00	83.00 (17.35)	100.00	50.00
		iCartoonFace	100.00 (0.00)	100.00	100.00	95.00 (6.71)	100.00	80.00
GTSRB-43	WideResNet	FFHQ	100.00 (0.00)	100.00	100.00	91.86 (13.16)	100.00	50.00
		iCartoonFace	99.07 (2.90)	100.00	90.00	89.07 (13.61)	100.00	40.00
	DenseNet	FFHQ	99.77 (1.51)	100.00	90.00	68.60 (27.50)	100.00	10.00
		iCartoonFace	97.44 (5.32)	100.00	80.00	62.09 (27.75)	100.00	10.00

In the black-box scenario, the attack method can still achieve above 83% average ASR on CIFAR-10 and 62% average ASR on GTSRB-43, although the performance is inferior to that of the white-box scenario. We observe that even in the black-box scenario, 100% best ASR can be seen on all raw models, which means that some classes of victim models are extremely vulnerable. In spite of restrictions of the black-box scenario where the adversary cannot have access to the model and only be allowed to obtain output information, the vulnerability of the victim model, i.e. these classes, could not be reinforced. At the same time, we also see that all worst ASR values in the black-box scenario show a decline, compared with that in the white-box scenario. For example, the attack method achieves 10% worst ASR on the victim model DenseNet trained on GTSRB-43. This might be because classes that are hard to be fooled become even harder to be fooled in the black-box scenario.

Note that OOD examples constructed by our methods are all from a pre-trained GAN model which produces totally different images from that of victim models. The principle of our attack method is based on the fact that ML models do not know when they do not know. The success of our attack methods in white-box and black-box scenarios reminds model providers that when deploying an ML model in the open world, it is necessary to guarantee the legitimacy of each input. Otherwise, the ML model could make unexpected predictions for illegal inputs and this characteristic can be abused by an adversary to mount a novel security attack.

7.5.2 Comparison with Adversarial Example Attacks

We compare our methods with state-of-the-art adversarial example attacks. Totally, five powerful adversarial example attack methods are chosen: C&W [CW17b] and PGD [MMS+18] that are used in the white-box scenario, and ZOO [CZS+17], DBA [BRB18] and HSJA [CJW20] that are used in the black-box scenario. Note that although there is a work studying OOD attack [SBS+19], this work only applies the PGD method to OOD dataset. Therefore, we do not explicitly compare this work and the attack method of this work can be considered as equivalent to the PGD method.

Table 7.3: Comparison with different attack methods on the white-box and black-box scenario.

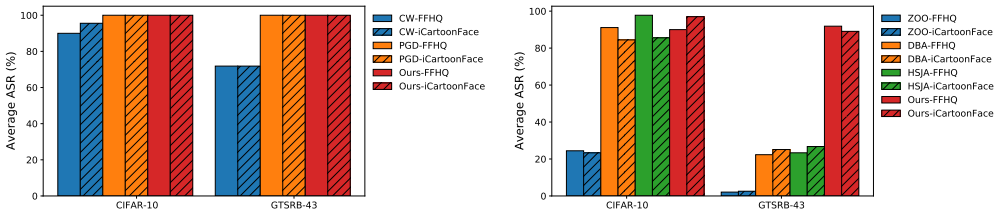
Scenarios	Method	OOD Dataset	CIFAR-10			GTSRB-43		
			Average ASR (SD) %	Best ASR %	Worst ASR %	Average ASR (SD) %	Best ASR %	Worst ASR %
White-box	CW	FFHQ	90.00 (10.48)	100.00	66.67	71.86 (21.05)	100.00	20.00
		iCartoonFace	95.56 (7.37)	100.00	77.78	71.86 (16.32)	100.00	30.00
	PGD	FFHQ	100.00 (0.00)	100.00	100.00	99.30 (3.34)	100.00	80.00
		iCartoonFace	100.00 (0.00)	100.00	100.00	95.58 (6.58)	100.00	70.00
	Ours	**FFHQ**	**100.00 (0.00)**	**100.00**	**100.00**	**100.00 (0.00)**	**100.00**	**100.00**
		iCartoonFace	**100.00 (0.00)**	**100.00**	**100.00**	**99.07 (2.90)**	**100.00**	**90.00**
Black-box	ZOO	FFHQ	24.44 (28.89)	88.89	0.00	2.09(4.60)	20.00	0.00
		iCartoonFace	23.33 (21.34)	55.56	0.00	2.56(20.00)	20.00	0.00
	DBA	FFHQ	91.11 (10.89)	100.00	66.67	22.33 (20.78)	80.00	0.00
		iCartoonFace	84.44 (14.23)	100.00	55.56	25.12(18.60)	70.00	0.00
	HSJA	FFHQ	97.78 (4.44)	100.00	88.89	25.35(19.09)	80.00	0.00
		iCartoonFace	85.56 (12.22)	100.00	55.56	26.74(16.24)	80.00	0.00
	Ours	**FFHQ**	**90.00 (10.00)**	**100.00**	**70.00**	**91.86 (13.16)**	**100.00**	**50.00**
		iCartoonFace	**97.00 (4.58)**	**100.00**	**90.00**	**89.07 (13.61)**	**100.00**	**40.00**

We implement the five algorithms by the open-source library — Adversarial Robustness Toolbox [ART18] and the suggested hyperparameters are used. WideResNet is chosen as the architecture of victim models. All restricted adversarial OOD examples are constructed under the ℓ_∞ distance. Following the tradition of prior work [CJW20], we set the maximum perturbation as $8/255$. An example is considered a success if it is recognized as a pre-specified class and the magnitude of perturbation does not exceed the maximum perturbation. In addition, the maximum number of queries in the black-box scenario is set as 25,000 per image for all attack methods.

Note that C&W and PGD are not used in the black-box scenario because both methods are required to train a substitute model which requires extra training data whose distribution has to be similar to that of the training set of the victim model. Instead, we choose ZOO, DBA, and HSJA in the black-box scenario, which all show promising attack performance. Here, we highlight that our proposed framework can be applied in both scenarios without any additional data.

Results. Table 7.3 and Figure 7.2 show results of our attack methods and adversarial example attack algorithms on both the white-box and black-box scenario. Overall, our methods achieve similar or superior attack performance on both scenarios, compared with the five state-of-the-art adversarial example attacks.

In the white-box scenario, our method can achieve a 100% average ASR on CIFAR-10. Similar performance can be seen for the PGD method on CIFAR-10 but the C&W method shows a slightly inferior attack performance. More than 99% average ASR on GTSRB-43 can be achieved by our method, while both the C&W and PGD attack method cannot achieve better performance where the average ASR of C&W and PGD is 71.86% and 95.85%, respectively. One possible reason is that due to the significant difference between in-distribution samples and out-of-distribution samples, ℓ-norm-based adversarial example attacks indeed require much larger magnitudes



(a) White-box scenario (b) Black-box scenario

Figure 7.2: Comparison with different attack methods.

of perturbations. We also notice that all attack methods show perfect performance with respect to best ASR, while the attack performance presents significant differences in the aspect of worst ASR. For example, the worst ASR on GTSRB-43 is 90.00% for our attack methods, whereas it is only 70.00% and 20.00% for the PGD method and the C&W method, respectively.

In the black-box scenario, our method further shows superior attack performance on both datasets. For example, ours can achieve over 90.00% average ASR on CIFAR-10 and 89.07% average ASR on GTSRB-43, while the method ZOO only gains no more than 24.44% and 2.56% average ASR on CIFAR-10 and GTSRB-43, respectively. Although DBA and HSJA can obtain similar attack performance in comparison with our method on CIFAR-10, the significant advantages of our method can be seen on GTSRB-43. For instance, the average ASR of our method on GTSRB-43 is at least higher 62% than that of DBA and HSJA where the best performance is at most 26.74% average ASR. This also reminds model owners that when evaluating the robustness of their models in the open world, it is not sufficient to directly apply current adversarial example attacks.

7.5.3 Attack Performance on Models with Defenses

We further investigate whether our proposed methods can evade state-of-the-art defense measures.

Existing works mainly focus on the detection of OOD samples to improve the robustness of models in the open world. To do this, these methods first compute a score for each sample based on the outputs of the model. The score can be regarded as the probability that this sample is an in-distribution example. Then, a sample is an in-distribution example if its score is larger than a pre-defined threshold τ. Specifically, given a victim model F, an OOD detector can be represented as:

$$Detector(x) = \begin{cases} 1 & F(x) > \tau \\ 0 & F(x) \leq \tau \end{cases} \tag{7.3}$$

Depending on whether or not the victim model F needs to be changed, OOD methods can be divided into two categories. The first category does not require any changes of the victim model F. In other words, this type of defense can be directly applied to a model without retraining and it detects OOD examples based on the confidence scores of models. We choose one classic and widely-used defense method ODIN [LLS18], considering its excellent performance in this category. The

second category requires changes of models, such as retraining models with novel loss functions, adding auxiliary datasets into the training process, and even introducing adversarial training. One classic defense method OE [HMD19] and the latest state-of-the-art defense method ATOM [CLW+21] are considered in this work. A brief introduction of each defense is given as follows.

• **ODIN.** This defense [LLS18] utilizes the model's outputs, i.e. confidence scores, to detect OOD examples. The main assumption is that neural networks tend to output higher confidence scores to in-distribution examples than OOD examples. The defense utilizes temperature scaling [HVD15] and adversarial perturbations [GSS15] to further enlarge the differences between in-distribution and OOD examples. In detail, for a given example x, a perturbed example $\tilde{x}$ is obtained by adding adversarial perturbations and the calibrated confidence score of $\tilde{x}$ can be calculated by temperature scaling. An example belongs to in-distribution if the score is greater than a pre-defined threshold τ. This defense can be considered as a post-processing technique and does not require model owners to retrain victim models.

In our experiments, we utilize the ImageNet dataset to choose hyperparameters. For WideResNet, we set the perturbation magnitude as 0 for both CIFAR-10 and GTSRB-43. For DenseNet, the perturbation magnitude is set as 0.0004 for CIFAR-10 and 0.001 for GTSRB-43, respectively. The temperature scaling is set as 1000 in all settings.

• **OE.** This defense [HMD19] makes use of auxiliary OOD datasets to train anomaly detectors that can generalize and detect unseen OOD examples. It introduces a new loss function that cannot only learn the original classification objective but also learn heuristics to detect whether a sample is an OOD example by the auxiliary OOD dataset. This approach requires retraining a model.

• **ATOM.** This defense [CLW+21] combines adversarial training and an auxiliary OOD dataset to collaboratively enhance the performance of OOD detection. More specifically, this approach first selects informative outliers from an auxiliary OOD dataset, and then these selected OOD samples and in-distribution samples are utilized to adversarially train the model. The projected gradient descent method [MMS+18] is adopted for adversarial training. This approach also requires retraining a model.

For all these defense methods, we follow the convention of the research community in OOD detection and the threshold τ is chosen when the true-positive rate (TPR) is 95% where the TPR refers to the ratio of in-distribution examples correctly classified as in-distribution examples. For OE and ATOM, the ImageNet dataset is used as the auxiliary OOD dataset for training. We train the models with the SGD optimizer and the suggested hyperparameters of their papers are used.

All hyperparameters of our attack methods on defense models are the same as those on raw models, except for loss functions. Specifically, we use loss function $f4$ for OE and cross-entropy loss function $f3$ for the victim model DenseNet trained on GTSRB-43 in the black-box scenario and ℓ_2 loss function $f2$ is used for the remaining victim models. The reason is that these loss functions show better attack

Table 7.4: Performance of victim models with defense measures. ↓ means smaller is better while ↑ means larger is better.

Method	Dataset	Model	FPR % ↓	Accuracy % ↑
ODIN	CIFAR-10	WideResNet	54.978	92.10
		DenseNet	46.024	91.12
	GTSRB-43	WideResNet	8.502	93.11
		DenseNet	11.400	92.86
OE	CIFAR-10	WideResNet	0.110	91.99
		DenseNet	0.226	91.38
	GTSRB-43	WideResNet	0.002	93.71
		DenseNet	0.014	93.23
ATOM	CIFAR-10	WideResNet	0.014	90.16
		DenseNet	0.038	89.05
	GTSRB-43	WideResNet	0.016	92.34
		DenseNet	0.096	91.43

performance and it is hard to achieve the best performance only by one loss function for various defense measures. We detail different loss functions in Section 7.5.4. Again, we only consider the targeted attack for models with defenses, because a successful targeted attack also indicates a successful untargeted attack. Note that the ASR in the defenses has a higher requirement: one successful attack refers to that one sample needs to first evade the OOD detection and is recognized as a pre-specified class that the adversary wishes.

Defense performance of models. Table 7.4 shows the performance of various defense measures. Here, the false-positive rate (FPR) refers to the ratio of OOD samples that are predicted as in-distribution samples. A lower FPR value indicates better detection performance. Accuracy refers to the ratio of samples that is predicted as an in-distribution sample and is recognized as a correct class. We can see that almost all models show outstanding detection performance and prediction performance, although the method ODIN on CIFAR-10 shows poor OOD detection performance.

Results. Table 7.5 shows the attack performance of our proposed methods under different defense measures. Overall, all defenses cannot prevent our attacks on the white-box and black-box scenarios, although these defenses can lower our attack success rate to some degree. The defense measures mainly concentrate on decreasing the worst ASR of our methods, while the best ASR is hardly reduced. The attack performance in the white-box scenario generally is better than that in the black-box scenario.

For the defense method ODIN, the average ASR in the white-box scenario is above 86.00% on CIFAR-10 and 69.38% on GTSRB-43 while in the black-box scenario the attack performance shows a decrease where the average ASR is more than 40.00% on CIFAR-10 and 9.53% on GTSRB-43. With regard to the best ASR, our attack performance remains 100.00% in the white-box scenario and more than 60.00% in the black-box scenario. We observe that the worst ASR value of 0.00% can be seen on several victim models in the black-box scenario, indicating that the ODIN

Table 7.5: Attack performance on various defense methods. SD refers to standard deviation.

Defense Method	In-distribution Dataset	Victim Model	OOD Dataset	White-box Average ASR (SD) %	White-box Best ASR %	White-box Worst ASR %	Black-box Average ASR (SD) %	Black-box Best ASR %	Black-box Worst ASR %
ODIN	CIFAR-10	WideResNet	FFHQ	96.00 (4.90)	100.00	90.00	45.00 (38.28)	100.00	0.00
			iCartoonFace	87.00 (14.87)	100.00	60.00	59.00 (24.27)	90.00	20.00
		DenseNet	FFHQ	93.00 (12.69)	100.00	60.00	40.00 (38.73)	100.00	0.00
			iCartoonFace	86.00 (12.81)	100.00	60.00	50.00 (20.00)	90.00	20.00
	GTSRB-43	WideResNet	FFHQ	93.72 (8.08)	100.00	70.00	37.21 (31.28)	100.00	0.00
			iCartoonFace	75.12 (18.97)	100.00	20.00	31.40 (25.30)	90.00	0.00
		DenseNet	FFHQ	87.91 (17.33)	100.00	30.00	13.72 (23.13)	80.00	0.00
			iCartoonFace	69.38 (22.11)	100.00	10.00	9.53 (13.63)	60.00	0.00
OE	CIFAR-10	WideResNet	FFHQ	44.00 (15.62)	60.00	10.00	35.00 (29.41)	100.00	0.00
			iCartoonFace	13.00 (11.87)	30.00	0.00	63.00 (19.52)	90.00	40.00
		DenseNet	FFHQ	56.00 (20.10)	90.00	20.00	44.00 (28.00)	100.00	10.00
			iCartoonFace	41.00 (15.78)	70.00	20.00	75.00 (17.46)	100.00	40.00
	GTSRB-43	WideResNet	FFHQ	73.26 (17.75)	100.00	30.00	7.44 (14.16)	70.00	0.00
			iCartoonFace	42.09 (13.90)	70.00	10.00	5.81 (11.05)	40.00	0.00
		DenseNet	FFHQ	53.64 (21.52)	100.00	10.00	4.19 (12.24)	70.00	0.00
			iCartoonFace	23.26 (18.26)	60.00	0.00	1.16 (3.21)	10.00	0.00
ATOM	CIFAR-10	WideResNet	FFHQ	86.00 (8.00)	100.00	70.00	23.00 (24.52)	80.00	0.00
			iCartoonFace	56.00 (14.97)	80.00	40.00	34.00 (22.00)	80.00	0.00
		DenseNet	FFHQ	78.00 (15.36)	100.00	50.00	11.00 (14.46)	50.00	0.00
			iCartoonFace	58.00 (15.36)	90.00	40.00	18.00 (13.27)	50.00	0.00
	GTSRB-43	WideResNet	FFHQ	59.30 (26.80)	100.00	10.00	0.93 (3.61)	20.00	0.00
			iCartoonFace	54.65 (21.50)	90.00	10.00	1.40 (5.10)	30.00	0.00
		DenseNet	FFHQ	62.79 (23.06)	100.00	0.00	0.23 (1.51)	10.00	0.00
			iCartoonFace	56.98 (22.26)	100.00	0.00	2.56 (4.87)	20.00	0.00

defense indeed improves the robustness of some victim models' classes to some extent.

For the defense method OE, the average ASR in the white-box scenario varies from 13.00% to 56.00% on CIFAR-10 and from 23.26% to 73.26% on GTSRB-43. In contrast, in the black-box scenario, the average ASR is above 35.00% on CIFAR-10 and 1.16% on GTSRB-43. We can also see that the best ASR is still high in both scenarios. Compared with the white-box scenario, the worst ASR has a lower value in the black-box scenario. We also observe that different GAN models have effects on attack performance. For instance, the GAN trained on FFHQ can achieve better performance than that on iCartoonFace in the white-box scenario.

For the defense method ATOM, our attack methods can achieve at least an average ASR of 54% for all victim models in the white-box scenario. As a comparison, the average ASR fluctuates from 11.00% to 34.00% on CIFAR-10 and from 0.23% to 2.56% on GTSRB-43. Although the worst ASR value is low, especially in the black-box scenario, our attack performance shows promising performance with respect of the best ASR. That is, some vulnerable classes of the victim model are still easily fooled. We should emphasize that usually the robustness level of a model largely depends on the most vulnerable point.

Our attack methods can successfully evade these defenses and achieve targeted attacks in both scenarios. Although a majority of worst ASR values is low and even 0, as shown in the worst ASR column in Table 7.5, the fact that these defense measures cannot decrease the best ASR means that the real threats of a model

cannot be alleviated. Our work also reminds model owners that when assessing their defense measures, the most vulnerable classes of a model should be focused on.

7.5.4 Analysis of Attack Performance

In this section, we analyze attack performance in terms of different types of loss functions and optimization methods. We also depict the attack process of our attack methods in both scenarios. In this section, we fix the victim models as the WideResNet model trained on CIFAR-10 and choose StyleGAN trained on FFHQ.

Effects of loss functions. Loss functions play an important role in generating adversarial OOD examples. As shown in Equation 7.1, our attack framework constructs an adversarial OOD example by minimizing a loss function. There are many different types of loss functions and in this work, we study the following loss function f:

$$\text{f1} = |F(G(z)) - F(x_{ref})| \tag{7.4}$$

$$\text{f2} = (F(G(z)) - F(x_{ref}))^2 \tag{7.5}$$

$$\text{f3} = \mathit{CrossEntropy}(F(G(z)), (F(x_{ref}))_{max}) \tag{7.6}$$

$$\text{f4} = \text{f2} + \lambda \cdot (-\max(\mathit{softmax}(F(G(z))))) \tag{7.7}$$

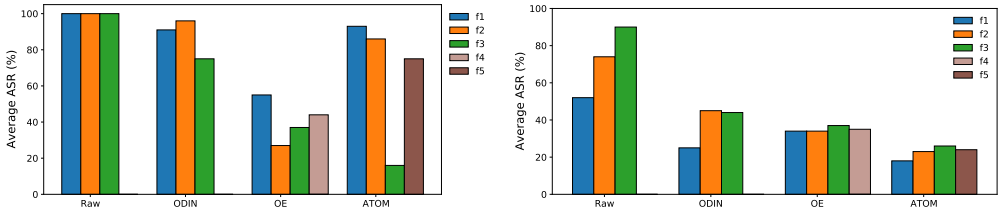
$$\text{f5} = \text{f2} + \lambda \cdot (\mathit{softmax}(F(G(z)))_{m+1}) \tag{7.8}$$

λ is a hyperparameter. f1, f2, f3 are ℓ_1 loss, ℓ_2 loss, and cross-entropy loss function, respectively. They are common and basic loss functions. Loss function f4 is designed for OE defense measures but can be widely applied to attack these defense techniques that detect OOD samples based on higher output scores. It adds a new item that aims to maximize the output scores of a sample, which makes the attack evade detection more efficiently. Loss function f5 is designed for ATOM defense measure and it adds a new item that aims to minimize the output scores of the OOD class of a sample. Similarly, this loss function can be applied to attack these defense techniques that add a new OOD class $m+1$ in ML models besides the normal number of classes m. The new class $m+1$ of this type of defense is specially used for OOD detection and a higher output score of this class means a higher OOD probability.

Table 7.6 and Figure 7.3 present attack performance across different loss functions in both scenarios. Overall, we can see that different loss functions indeed have different attack performances. For example, loss functions are hard to show the difference when attacking raw models in the white-box scenario. In contrast, the loss function f3 excels in raw models in the black-box scenario. When mounting attacks against victim models with defenses, it becomes somewhat more difficult to choose a loss function that is applied for all defenses, because the attack performance of these loss functions fluctuates. However, loss function f1, f2 and f3 can be regarded as good starting points for an attack. We also observe that f4 and f5 loss functions also show a good performance, comprehensively considering both attack scenarios.

Table 7.6: Attack performance in terms of different types of loss functions. SD: standard deviation.

Method	Loss function	White-box			Black-box		
		Average ASR(SD) %	Best ASR%	Worst ASR%	Average ASR(SD) %	Best ASR%	Worst ASR%
Raw	f1	100.00 (0.00)	100.00	100.00	52.00 (34.58)	100.00	0.00
	f2	100.00 (0.00)	100.00	100.00	74.00 (26.15)	100.00	20.00
	f3	100.00 (0.00)	100.00	100.00	90.00 (10.00)	100.00	70.00
ODIN	f1	91.00 (11.36)	100.00	70.00	25.00 (35.28)	100.00	0.00
	f2	96.00 (4.90)	100.00	90.00	45.00 (38.28)	100.00	0.00
	f3	75.00 (18.03)	100.00	50.00	44.00 (33.23)	100.00	0.00
OE	f1	55.00 (23.35)	90.00	20.00	34.00 (31.69)	100.00	0.00
	f2	27.00 (15.52)	60.00	10.00	34.00 (30.40)	100.00	0.00
	f3	37.00 (30.02)	100.00	0.00	37.00 (27.95)	100.00	10.00
	f4	44.00 (15.62)	60.00	10.00	35.00 (29.41)	100.00	0.00
ATOM	f1	93.00 (11.87)	100.00	60.00	18.00 (19.90)	60.00	0.00
	f2	86.00 (8.00)	100.00	70.00	23.00 (24.52)	80.00	0.00
	f3	16.00 (14.29)	40.00	0.00	26.00 (26.53)	80.00	0.00
	f5	75.00 (12.04)	90.00	60.00	24.00 (25.77)	90.00	0.00



(a) White-box scenario (b) Black-box scenario

Figure 7.3: Attack performance on various models in terms of different types of loss functions.

Effects of optimization methods. We study two types of optimization methods: SGD [RM51] and Adam [KB15]. Both optimization methods are used in the white-box scenario because they require computing gradients. We do not consider other optimization methods in the black-box scenario in addition to the Powell method. This is because existing black-box optimization methods are hard to be applied to the current attack scenario and a new black-box optimization needs to be proposed. We leave it for further work.

Figure 7.4 and Table 7.7 show attack performance on different types of optimization methods. We can observe that the Adam optimizer shows better attack performance on victim models with defense measures, compared with the SGD optimizer. For example, the Adam optimizer can increase the average ASR by 22.00% on OE and the attack performance on ATOM can be improved to 99.00%. For raw models, both optimizers achieve amazing performance and do not show a difference.

Optimization processes. Figure 7.5 illustrates the optimization processes when attacking the raw model trained on CIFAR-10. Cross-entropy loss is used in both

Table 7.7: Attack performance on various models in terms of different types of optimizers. SD refers to standard deviation.

Method	Optimizer	Average ASR (SD) %	Best ASR %	Worst ASR %
Raw	SGD	100.00 (0.00)	100.00	100.00
	Adam	100.00 (0.00)	100.00	100.00
ODIN	SGD	96.00 (4.90)	100.00	90.00
	Adam	99.00 (3.00)	100.00	90.00
OE	SGD	27.00 (15.52)	60.00	10.00
	Adam	49.00 (25.87)	90.00	10.00
ATOM	SGD	86.00 (8.00)	100.00	70.00
	Adam	99.00 (3.00)	100.00	90.00

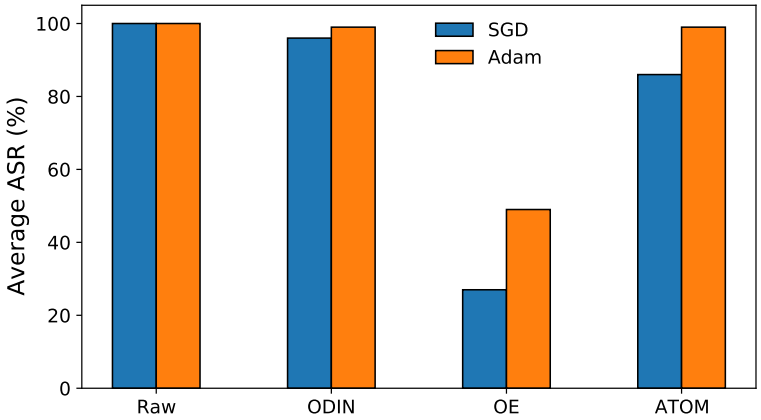
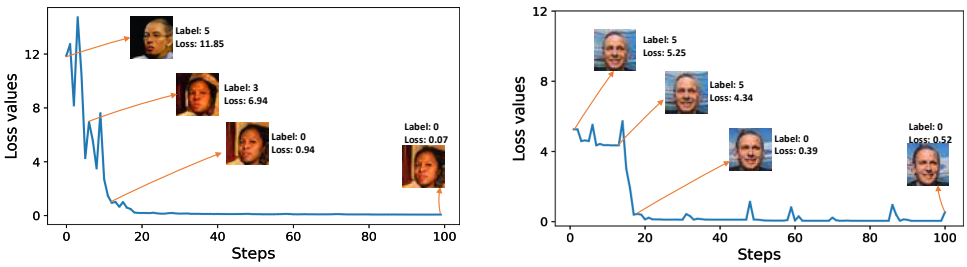


Figure 7.4: Attack performance on various models in terms of different types of optimizers.



(a) SGD optimization in the white-box.

(b) Powell optimization in the balck-box.

Figure 7.5: The optimization process of our attacks on both scenarios. The target label is 0.

scenarios. SGD optimizer and Powell optimizer is used in the white-box and black-box scenario, respectively. Here, the target label that the adversary wishes is labeled 0. We can observe that with the decreases in loss values, attack methods in the white-box and black-box scenarios can succeed to find target samples.

7.6 Discussion

Our attack methods craft adversarial OOD examples through any pre-trained GAN model to fool a victim classification model. In other words, our proposed framework builds a mapping where vulnerable points of a victim classification model, i.e. adversarial OOD examples, can be found in one GAN model. Here, we emphasize that this is different from perturbation-based adversarial example attacks in essence. Although adversarial example attacks can find adversarial OOD examples, these examples are discrete and isolated. In contrast, adversarial OOD examples constructed by our methods are from a GAN model. Although our novel attack methods can fool ML models and circumvent state-of-the-art OOD detection, it is still possible to mitigate our attacks. For instance, as illustrated in Section 7.5.3, one possible efficient method is to design OOD detection that focuses on the most vulnerable classes of a victim model.

7.7 Conclusion

Real-world ML models face various input examples, including OOD examples that have a different distribution from the training set. In this chapter, we have proposed a novel attack that causes ML models to misclassify an OOD example as a pre-specified class that the adversary wishes. By leveraging any off-the-shelf pre-trained GAN model, our framework attempts to craft the OOD example by minimizing the distance between the victim model's output and the pre-specified class. Based on different attack scenarios: white-box and black-box, different attack methods are proposed. We conduct extensive experiments on different victim models on different datasets. Our experimental results show that our attack methods achieve comparable performance on both scenarios, compared with five adversarial example attacks. Moreover, our evaluation also demonstrates that even for victim models deploying defense mechanisms, our attack methods can still achieve competitive performance on the white-box and black-box scenarios.

Our proposed attack methods can utilize any off-the-shelf unconditional GAN. Besides, our methods can also craft OOD examples in a unified framework, which can be applied to both the white-box and black-box scenarios. More importantly, our methods have a different attack mechanism from existing attack methods. Therefore, it is possible to consider our methods as a supplementary test tool to evaluate the robustness of real-world ML models. In future, we plan to develop defense measures to mitigate our attack. In addition, it is a promising direction to design an attack that aims to reduce the number of queries in the black-box scenario.

Chapter 8

Conclusion and Future Work

The demand for building generative AI systems for various fields, such as healthcare, artistic creation, and social media, is on the rise. Studying the privacy risks of generative AI systems can help to guarantee that they are utilized responsibly. Additionally, breakthroughs in generative AI can provide innovative perspectives to investigate trustworthy discriminative models.

8.1 Conclusion

In this dissertation, we discussed the privacy risks of generative models and the security risks of classification models. We have demonstrated a wide range of privacy attacks against generative models from the perspectives of model privacy and data privacy. In terms of model privacy, we have shown powerful model extraction attacks on GANs. We then have demonstrated a protection method, which can verify the ownership of GANs on both physical stealing and model extraction scenarios. In terms of data privacy, we have shown the membership inference risks of diffusion models, and revealed the relationship between leakage of training samples and generative mechanisms of diffusion models. We then have presented the property inference risks of diffusion models, and proposed a model-agnostic and plug-in defense method. In addition to studying privacy in generative models, we have shown that adversaries can perform more powerful evasion attacks by utilizing off-the-shelf GAN models.

8.2 Future Work

Although these works have revealed the privacy risks of generative models and novel security risks of classification models, there are several exciting directions for future research.

Model privacy on multimodal generative models. Our work on model privacy of GANs shows that as long as model owners allow adversaries to sufficiently query a victim GAN model, such as 50K queries, adversaries can nearly duplicate the victim GAN model without all the hard work that model owners did, such as collecting datasets, designing advanced architectures. Indeed, very recent work on generative

language models has shown that Sandford's researchers can steal the OpenAI's text-davinci-003 model (GPT-3.5) by querying the exposed interface [TGZ+23]. The resulting model Alpaca utilizes 52K instruction-following demonstrations generated from the text-davinci-003 model and can achieve qualitatively similar performance on self-instruct evaluation sets. Certainly, these attacks that compromise model privacy can raise significant apprehensions for model owners about their intellectual property. Worsely, these apprehensions are being further compounded because more and more multimodal and larger generative models have been developed, such as multimodal text-to-image generative models and text-to-audio generative models. Therefore, it is essential to investigate model privacy risks of multimodal generative models. More importantly, from the perspective of defense, studying ownership protection methods for these generative models is urgent.

Novel protection mechanisms on data privacy. Our work on membership inference of diffusion models reveals that training samples have much higher privacy risks in medium amounts of noise in the whole diffusion time steps, which indicates that the current noise mechanism used in diffusion models might need to be redesigned. Indeed, very recent work from the DeepMind team proposes a differentially private diffusion model to defend against privacy attacks, in which a modified timestep noise mechanism motivated by our findings is utilized to replace the original uniform distribution [GBG+23]. On the one hand, studying diffusion models with differential privacy and improving better quality of generated samples is still a promising direction. On the other hand, it is very valuable to explore novel protection mechanisms to mitigate model memorization. One possible example is to design novel algorithms that aim to erase memorized training samples via editing a well-trained model, which can avoid the issue of model training and enjoy computation-friendly advantages.

Trustworthy applications via generative models. Our work on OOD attacks via pre-trained GANs shows that it is useful to apply generative models to reveal the security risks of discriminative models. Currently, the robustness of discriminative models is still a big challenge when they are deployed in an open world. For example, discriminative models in autonomous driving systems might make inappropriate decisions for certain new samples. Given the success of generative models in synthesizing new and unseen samples, improving the robustness performance of discriminative models via the generation capability of generative models is an important direction.

Looking forward, we believe that progress in privacy in generative models not only benefits the responsible use of generative AI, but also deepens our understanding of generative techniques.

Bibliography

[AB17] Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[ABC+18] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 1615–1631. USENIX Association, 2018.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 214–223. PMLR, 2017.

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318. ACM, 2016.

[AHS85] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

[AMS+15] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[And82] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[AOD+19] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[ART18] ART. Adversarial robustness toolbox (art). `https://github.com/Trusted-AI/adversarial-robustness-toolbox`, 2018.

[AS66] Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

[ASTG19] Eirikur Agustsson, Alexander Sage, Radu Timofte, and Luc Van Gool. Optimal transport maps for distribution preserving operations on latent spaces of generative models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[BB99] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 12, 1999.

[BCM+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2013.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[BDTD+16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Zieba Karol. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[BF18] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2018.

[BKP+20] Yasaman Bahri, Jonathan Kadmon, Jeffrey Pennington, Sam S. Schoenholz, Jascha Sohl-Dickstein, and Surya Ganguli. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11(1):501–528, 2020.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin

Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[BNS+06] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 16–25. ACM, 2006.

[BRB18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[BZW+19] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a GAN cannot generate. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 4502–4511. IEEE, 2019.

[CAO+23] Harsh Chaudhari, John Abascal, Alina Oprea, Matthew Jagielski, Florian Tramèr, and Jonathan Ullman. Snap: Efficient extraction of private properties with poisoning. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 1935–1952. IEEE, 2023.

[CCG+20] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *Proceedings of USENIX Security Symposium (USENIX Security)*. USENIX Association, 2020.

[CCN+22] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[CCTCP21] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1964–1974. PMLR, 2021.

[CGZ+21] Kangjie Chen, Shangwei Guo, Tianwei Zhang, Xiaofei Xie, and Yang Liu. Stealing deep reinforcement learning models for fun and profit. In *Proceedings of ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 307–319, 2021.

[CHN+23] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

[CHZ22] Tianshuo Cong, Xinlei He, and Yang Zhang. SSLGuard: A watermarking scheme for self-supervised learning pre-trained encoders. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 579–593. ACM, 2022.

[CJG21] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 14–25, 2021.

[CJM20] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *Proceedings of Annual International Cryptology Conference (CRYPTO)*. Springer, 2020.

[CJW20] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 1277–1294. IEEE, 2020.

[CLE+19] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 267–284. USENIX Association, 2019.

[CLL+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[CLW+21] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. Atom: Robustifying out-of-distribution detection using outlier mining. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 12977 of *Lecture Notes in Computer Science*. Springer, 2021.

[Csi64] Imre Csiszár. Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten. *Magyer Tud. Akad. Mat. Kutato Int. Koezl.*, 8:85–108, 1964.

[CTW+21] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 2633–2650. USENIX Association, 2021.

[CW17a] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the ACM workshop on artificial intelligence and security (AISec)*, pages 3–14. ACM, 2017.

[CW17b] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[CWP+22] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. Copy, right? a testing framework for copyright protection of deep learning models. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 1013–1030. IEEE, 2022.

[CWX+21] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021.

[CXC+19] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2267–2281. ACM, 2019.

[CYZF20] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 343–362. ACM, 2020.

[CZS+17] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the ACM workshop on artificial intelligence and security (AISec)*, pages 15–26. ACM, 2017.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 4171–4186, 2019.

[DCVK22] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. Differentially private diffusion models. *arXiv preprint arXiv:2210.09929*, 2022.

[DDK+22] Adam Dziedzic, Haonan Duan, Muhammad Ahmad Kaleem, Nikita Dhawan, Jonas Guan, Yannis Cattan, Franziska Boenisch, and Nicolas Papernot. Dataset inference for self-supervised models. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2022.

[DJP+20] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[DKW+23] Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? *arXiv preprint arXiv:2302.01316*, 2023.

[DN21] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 8780–8794. Curran Associates, Inc., 2021.

[DP80] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[DSDB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[DTL21] Yuzhen Ding, Nupur Thakur, and Baoxin Li. Does a GAN leave distinct model-specific fingerprints. In *Proceedings of British Machine Vision Conference (BMVC)*, 2021.

[Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In *Proceedings of International Conference on Theory and Applications of Models of Computation (TAMC)*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008.

[DWW20] Xin Ding, Z Jane Wang, and William J Welch. Subsampling generative adversarial networks: Density ratio estimation in feature space with softplus loss. *IEEE Transactions on Signal Processing*, 68:1910–1922, 2020.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1322–1333. ACM, 2015.

[FLL+22] Zhen Fang, Yixuan Li, Jie Lu, Jiahua Dong, Bo Han, and Feng Liu. Is out-of-distribution detection learnable? In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2022.

[Fol20] Joachim Folz. Simplejpeg 1.4.0. `https://gitlab.com/jfolz/simplejpeg`, 2020.

[Fre99] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[FW15] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, 2015.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GBG+23] Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L Smith, Olivia Wiles, and Borja Balle. Differentially private diffusion models generate useful synthetic images. *arXiv preprint arXiv:2302.13861*, 2023.

[GCB+18] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[GGML15] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 881–889. PMLR, 2015.

[GLDGG19] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[GLF+23] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Sequence to sequence text generation with diffusion models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023.

[GMX+13] Ian Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2672–2680. Curran Associates, Inc., 2014.

[GSS15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[GWY+18] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings*

of ACM SIGSAC Conference on Computer and Communications Security (CCS), pages 619–633. ACM, 2018.

[HAB19] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 41–50. IEEE, 2019.

[HB17] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1501–1510. IEEE, 2017.

[HG17] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[HHB19] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. Monte carlo and reconstruction membership inference attacks against generative models. In *Proceedings on Privacy Enhancing Technologies*, pages 232–249. Sciendo, 2019.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[HLP+17] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5077–5086. IEEE, 2017.

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708. IEEE, 2017.

[HMD19] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[HMDDC19] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Membership inference attacks against generative models. In *Proceedings on Privacy Enhancing Technologies*, pages 133–152. Sciendo, 2019.

[HP21a] Hailong Hu and Jun Pang. Membership inference attacks against GANs by leveraging over-representation regions. In *Proceedings of*

ACM SIGSAC Conference on Computer and Communications Security (CCS), pages 2387—-2389. ACM, 2021.

[HP21b] Hailong Hu and Jun Pang. Stealing machine learning models: Attacks and countermeasures for generative adversarial networks. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, pages 1–16. ACM, 2021.

[HP22] Hailong Hu and Jun Pang. Fooling machine learning models: A novel out-of-distribution attack through generative adversarial networks. *Available at SSRN 4255820*, 2022.

[HP23a] Hailong HU and Jun PANG. Loss and likelihood based membership inference of diffusion models. In *Proceedings of the Information Security Conference (ISC)*, pages 121–141. Springer, 2023.

[HP23b] Hailong Hu and Jun Pang. Ownership protection of generative adversarial networks. *arXiv preprint arXiv:2306.05233*, 2023.

[HP23c] Hailong Hu and Jun Pang. PriSampler: Mitigating property inference of diffusion models. *arXiv preprint arXiv:2306.05208*, 2023.

[HRU+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 6626–6637. Curran Associates, Inc., 2017.

[HSC+22] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

[HSG+22] Jonathan Ho, Tim Salimans, Alexey A Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[Jag21] Matthew Jagielski. *Integrity and Privacy in Adversarial Machine Learning*. PhD thesis, Northeastern University, July 2021.

[JCB+20] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction

of neural networks. In *Proceedings of USENIX Security Symposium (USENIX Security)*. USENIX Association, 2020.

[JCCCP21] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 1937–1954. USENIX Association, 2021.

[JLG+15] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal, and Raheem A Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*. Internet Society, 2015.

[JSMA19] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. PRADA: protecting against dnn model stealing attacks. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[JSPHO21] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. Subpopulation data poisoning attacks. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 3104–3122. ACM, 2021.

[KAAL22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2022.

[Kag15] Kaggle.com. Diabetic retinopathy detection. `https://www.kaggle.com/c/diabetic-retinopathy-detection#references.`, 2015.

[KAH+22] Amirhossein Kazerouni, Ehsan Khodapanah Aghdam, Moein Heidari, Reza Azad, Mohsen Fayyaz, Ilker Hacihaliloglu, and Dorit Merhof. Diffusion models for medical image analysis: A comprehensive survey. *arXiv preprint arXiv:2211.07804*, 2022.

[KALL18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410. IEEE, 2019.

[KLA+20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119. IEEE, 2020.

[KNL+20] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *Proceedings of IEEE Security and Privacy Workshops (SPW)*, pages 69–75. IEEE, 2020.

[KPH+21] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[KTP+20] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of BERT-based APIs. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[KW14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LCC+19] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. COCO-GAN: generation by parts via conditional coordinating. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 4512–4521. IEEE, 2019.

[LCH+06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 1(0), 2006.

[LEMS19] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against model stealing attacks using deceptive perturbations. In *Proceedings of IEEE Security and Privacy Workshops (SPW)*, pages 43–49. IEEE, 2019.

[LF20] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 1605–1622. USENIX Association, 2020.

[LHM+19] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 10551–10560. IEEE, 2019.

[LHZG19] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, pages 126–137. ACM, 2019.

[LJW+20] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using GANs for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of ACM Internet Measurement Conference (IMC)*, pages 464–483. ACM, 2020.

[LLS18] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738. IEEE, 2015.

[LM05] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery in data mining (KDD)*, pages 641–647. ACM, 2005.

[LoC18] California State Legislature and Governor of California. California consumer privacy act of 2018. `https://oag.ca.gov/privacy/ccpa`, 2018.

[LRLZ22] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2022.

[LTG+22] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-LM improves controllable text generation. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 4328–4343. Curran Associates, Inc., 2022.

[LTH+17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic

single image super-resolution using a generative adversarial network. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017.

[LTR+19] Mario Lučić, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 4183–4192, 2019.

[LW16] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of European conference on computer vision (ECCV)*, pages 702–716. Springer, 2016.

[LWZZ19] Huiying Li, Emily Wenger, Ben Y Zhao, and Haitao Zheng. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*, 2019.

[LYK+20] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[LZ21] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 880–895. ACM, 2021.

[LZB+22] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2022.

[LZBZ22] Yiyong Liu, Zhengyu Zhao, Michael Backes, and Yang Zhang. Membership inference attacks by exploiting loss trajectory. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2085–2098, 2022.

[LZK21] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[MGC22] Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. Property inference from poisoning. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 1120–1137. IEEE, 2022.

[MGR+18] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In

Proceedings of European conference on computer vision (ECCV), pages 181–196. Springer, 2018.

[MGVP19] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do GANs leave artificial fingerprints? In *Proceedings of conference on multimedia information processing and retrieval (MIPR)*, pages 506–511. IEEE, 2019.

[MH20] Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[MMY23] Tomoya Matsumoto, Takayuki Miura, and Naoto Yanai. Membership inference attacks against diffusion models. *arXiv preprint arXiv:2302.03262*, 2023.

[MS20] Sasi Kumar Murakonda and Reza Shokri. ML Privacy Meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning. *arXiv preprint arXiv:2007.09339*, 2020.

[MSB12] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blin" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

[MSDCS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[MSDH19] Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *Proceedings of Conference on Fairness, Accountability, and Transparency*, pages 1–9. ACM, 2019.

[Mül97] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[MYP20] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[NBC+08] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, J Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. *LEET*, 8(1-9):16–17, 2008.

[NDR+21] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[NWC+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.

[OAFS18] Seong Joon Oh, Max Augustin, Mario Fritz, and Bernt Schiele. Towards reverse-engineering black-box neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[OCN+21] Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3630–3639. IEEE, 2021.

[OOS17] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 2642–2651, 2017.

[OSF19] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4954–4963. IEEE, 2019.

[PBL10] Eckhard Platen and Nicola Bruti-Liberati. *Numerical solution of stochastic differential equations with jumps in finance*, volume 64. Springer Science & Business Media, 2010.

[PCYJ17] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *proceedings of Symposium on Operating Systems Principles (SOSP)*, pages 1–18. ACM, 2017.

[PGS+20] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. ActiveThief: Model extraction using active learning and unannotated public data. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 865–872. AAAI, 2020.

[PLWZ19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2337–2346. IEEE, 2019.

[PMG+17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of ACM on Asia conference on computer and communications security (ASIACCS)*, pages 506–519. ACM, 2017.

[PMG+18] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. In *Proceedings of the VLDB Endowment*, pages 1071–1083. VLDB Endowment, 2018.

[PMSW18] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. SoK: Security and privacy in machine learning. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[PotEU16] European Parliament and Council of the European Union. Art. 35 gdpr: Data protection impact assessment. `https://gdpr-info.eu/art-35-gdpr/`, 2016.

[Pow64] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[PTD+22] Walter HL Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F Da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M Jorge Cardoso. Brain imaging generation with latent diffusion models. In *MICCAI Workshop on Deep Generative Models*, pages 117–126. Springer, 2022.

[RBL+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695. IEEE, 2022.

[RDN+22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[RM15] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1530–1538. PMLR, 2015.

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016.

[RW18] Eitan Richardson and Yair Weiss. On GANs and GMMs. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 5847–5858. Curran Associates, Inc., 2018.

[SBS+19] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *Proceedings of the ACM workshop on artificial intelligence and security (AISec)*, pages 105–116. ACM, 2019.

[SCC+22] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 1–10. ACM, 2022.

[Sch02] E Eugene Schultz. A framework for understanding and predicting insider attacks. *Computers & Security*, 21(6):526–531, 2002.

[SDWMG15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 2256–2265. PMLR, 2015.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc., 2019.

[SE22] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *Proceedings on Privacy Enhancing Technologies*, 4:528–551, 2022.

[SGTZ20] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of GANs for semantic face editing. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9243–9252. IEEE, 2020.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2234–2242. Curran Associates, Inc., 2016.

[SHC+23] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2023.

[SLH+21] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

[SM20] Shuang Song and David Marn. Introducing a new privacy testing library in tensorflow. `https://blog.tensorflow.org/2020/06/introducing-new-privacy-testing-library.html`, 2020.

[SME21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[Son21] Yang Song. Score-based generative modeling through stochastic differential equations. `https://github.com/yang-song/score_sde_pytorch`, 2021.

[Son22] Yang Song. *Learning to generate data by estimating gradients of the data distribution*. PhD thesis, Stanford University, August 2022.

[SSDK+21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[SSG22a] Axel Sauer, Katja Schwarz, and Andreas Geiger. StyleGAN-XL: Scaling StyleGAN to large diverse datasets. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 1–10. ACM, 2022.

[SSG+22b] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*, 2022.

[SSKE18] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 8322–8333. Curran Associates, Inc., 2018.

[SSSI12] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[SSZ19] Reza Shokri, Martin Strobel, and Yair Zick. Privacy risks of explaining machine learning models. *arXiv preprint arXiv:1907.00164*, 2019.

[SZ21] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in GANs. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1532–1540. IEEE, 2021.

[SZH+19] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*. Internet Society, 2019.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[TCBM20] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1633–1645. Curran Associates, Inc., 2020.

[TGZ+23] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. `https://crfm.stanford.edu/2023/03/13/alpaca.html`, 2023.

[THF+19] Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-hastings generative adversarial networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 6345–6353, 2019.

[Tie94] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, pages 1701–1728, 1994.

[TSSJ+22] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2779–2792. ACM, 2022.

[TVDJ19] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 8250–8260. Curran Associates, Inc., 2019.

[TYF20] Tatsuya Takemura, Naoto Yanai, and Toru Fujiwara. Model extraction attacks against recurrent neural networks. *arXiv preprint arXiv:2002.00123*, 2020.

[TZJ+16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 601–618. USENIX Association, 2016.

[UNSS17] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR)*, pages 269–277. ACM, 2017.

[vdWSN+14] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[VPB+15] N Venkatanath, D Praneeth, Maruthi Chandrasekhar Bh, Sumohana S Channappayya, and Swarup S Medasani. Blind image quality evaluation using perception based features. In *Proceedings of National Conference on Communications*, pages 1–6. IEEE, 2015.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[WG18] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 36–52. IEEE, 2018.

[WWZ+20] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8695–8704. IEEE, 2020.

[WYL+22] Yixin Wu, Ning Yu, Zheng Li, Michael Backes, and Yang Zhang. Membership inference attacks against text-to-image generation models. *arXiv preprint arXiv:2210.00968*, 2022.

[WYZ22] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022.

[XLZ+18] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3905–3911, 2018.

[XMSM20] Pan Xudong, Zhang Mi, Ji Shouling, and Yang Min. Privacy risks of general-purpose language models. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 1471–1488. IEEE, 2020.

[XSA+18] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. TextureGAN: Controlling deep image synthesis with texture patches. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8456–8465. IEEE, 2018.

[XZY+22] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[YDF19] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to GANs: Learning and analyzing GAN fingerprints. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 7556–7566. IEEE, 2019.

[YMMS22] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 3093–3106, 2022.

[YSAF21] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 14448–14457. IEEE, 2021.

[YSC+22] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry S. Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2022.

[YSS+21] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Gosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya

Mironov. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.

[YSZ+15] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[YZLL21] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[YZS+22] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

[ZC23] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023.

[ZCB+22] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *Proceedings of USENIX Security Symposium (USENIX Security)*, pages 4543–4560. USENIX Association, 2022.

[ZCGF23] Derui Zhu, Dingfan Chen, Jens Grossklags, and Mario Fritz. Data forensics in diffusion models: A systematic analysis of membership privacy. *arXiv preprint arXiv:2302.07801*, 2023.

[ZCSZ22] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against GANs. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*. Internet Society, 2022.

[ZGJ+18] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the Asia Conference on Computer and Communications Security (ASIACCS)*, pages 159–172. ACM, 2018.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of British Machine Vision Conference (BMVC)*, 2016.

[ZKSE16] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European conference on computer vision (ECCV)*, pages 597–613. Springer, 2016.

[ZLDQ19] Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. RankSRGAN: Generative adversarial networks with ranker for image super-resolution. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3096–3105. IEEE, 2019.

[ZSZZ20] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain GAN inversion for real image editing. In *Proceedings of European conference on computer vision (ECCV)*, pages 592–608. Springer, 2020.

[ZXL+17] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 5907–5915. IEEE, 2017.

[ZZR+20] Yi Zheng, Yifan Zhao, Mengyuan Ren, He Yan, Xiangju Lu, Junhui Liu, and Jia Li. Cartoon face recognition: A benchmark dataset. In *Proceedings of ACM International Conference on Multimedia (MM)*, pages 2264–2272. ACM, 2020.