# Evaluation of Deep Image Generation Through the Lens of Privacy and Utility

Ryan Webster

Normandie Université

UNIVERSITÉ CAEN NORMANDIE

# THÈSE

**Pour obtenir le diplôme de doctorat**

Spécialité **INFORMATIQUE**

Préparée au sein de l'**Université de Caen Normandie**

## Evaluation of Deep Image Generation Through the Lens of Privacy and Utility

Présentée et soutenue par

**RYAN WEBSTER**

**Thèse soutenue le 10/07/2023**

devant le jury composé de :

| | | |
|---|---|---|
| MME JULIE DIGNE | Chargé de recherche HDR - UNIVERSITE LYON 1 CLAUDE BERNARD | Rapporteur du jury |
| M. VINCENT LEPETIT | Professeur des universités - ENPC PARIS | Rapporteur du jury |
| M. FREDERIC JURIE | Professeur des universités - Université de Caen Normandie | Membre du jury |
| M. HERVÉ JÉGOU | Chercheur - META | Membre du jury |
| M. OLIVIER LEZORAY | Professeur des universités - Université de Caen Normandie | Membre du jury |
| M. LOIC SIMON | Maître de conférences - Université de Caen Normandie | Membre du jury Co-encadrant |
| M. NICOLAS THOME | Professeur des universités - UNIVERSITE PARIS 4 PARIS-SORBONNE | Président du jury |
| M. JULIEN RABIN | Maître de conférences HDR - Université de Caen Normandie | Directeur de thèse |

Thèse dirigée par **JULIEN RABIN** (GREYC ALGORITHMIQUE)

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

*Dedicated to my parents.*

# Abstract

Modern image generators, such as stable diffusion or Midjourney, have become large scale, complex and general systems. As the wide spread application and use of these systems grow, so do their potential problem areas. In this thesis we investigate how generative models can leak information about their training data and the problems that poses to both the systems and the users. Systems such as Midjourney are trained with data gathered from the web and protected content can appear during generation without attribution. As generative models also have widespread application in the medical domain, it's imperative for the utility of the generative model to not generate data with strict privacy protections. We present the automatic evaluation of generative models, with a focus on these issues. We first present several statistical measures that can measure the image quality of deep generators, the diversity of generated images and finally measure their ability to overfit training samples. For the rest of the thesis, we study the problem of membership inference. We investigate a diverse set of factors that lead to vulnerability to membership attacks. On the other hand, we also observe many training setups which lead to robustness and empirical privacy. We present several new membership attacks that made improvements over the state of the art. Finally, we present a state of the art data extraction attack, capable of reconstructing training images from the most widely used generation systems.

# Résumé

Les générateurs d'images modernes, tels que Stable Diffusion ou Midjourney, sont devenus des systèmes à grande échelle, complexes et généraux. À mesure que l'application et l'utilisation de ces systèmes se généralisent, leurs éventuels problèmes se multiplient. Dans cette thèse, nous étudions comment les modèles génératifs peuvent fuiter des informations sur leurs données d'entraînement et les problèmes que cela pose à la fois aux systèmes et aux utilisateurs. Des systèmes comme Midjourney sont entrainés avec des données collectées sur le web et des contenus protégés peuvent apparaître pendant la génération sans notification d'attribution. Comme les modèles génératifs ont également une application répandue dans le domaine médical, il est impératif pour l'utilité du modèle génératif de ne pas générer de données sous protection stricte de la vie privée. Nous présentons l'évaluation automatique des modèles génératifs, avec un accent sur ces problèmes. Nous présentons d'abord plusieurs mesures statistiques qui peuvent mesurer la qualité des images produites par de tels générateurs profonds, leur diversité et enfin mesurer leur capacité à surapprendre les échantillons d'entraînement. Pour le reste de la thèse, nous étudions le problème de l'inférence d'appartenance. Nous étudions un ensemble divers de facteurs qui conduisent à la vulnérabilité aux attaques d'appartenance. D'un autre côté, nous observons également de nombreuses configurations d'entraînement qui assurent empiriquement la robustesse et la confidentialité. Nous présentons plusieurs nouvelles attaques d'appartenance permettant des améliorations par rapport à l'état de l'art. Enfin, nous présentons une attaque de pointe pour l'extraction de données, capable de reconstruire des images d'entraînement à partir des systèmes de génération les plus largement utilisés.

# Acknowledgments

This was my first experience living abroad and one of the hardest challenges during my thesis was adapting to life in France. However, this was also a great advantage too. I meet so many wonderful people here from so many different backgrounds that I would have never meet elsewhere. Learning a new language and feeling outside my comfort zone was the norm for me and I gained a new sense of sympathy for others making the same endeavor. I would like to thank Zeina and Shivang who were amongst the first friends I meet here, who helped me adapt to life in France. I would like to thank Matthieu, Maksim, Dara and Tony for their friendship and support throughout my thesis. Tony, you're a big inspiration to me and have a knack for lifting others up. Maksim, I wouldn't have made it through the isolation during the pandemic without our shows, workouts and discussions. Thanks Dara for helping me learn French and for sharing great music. I would also like to thank my friends back home: Sam, Tony, Jake and Rolf. I felt your support throughout and coming home and seeing you guys would always be a high point of my year. I would like to thank my many friends in lab: Guillaume, Nathan, Benjamin, Darshan and many others for making life in lab enjoyable. Thank you to my surrogate advisor Olivier, you helped me immensely with my CSI meetings and other stuff for which I'm greatful for. I would like to thank the unwavering support of my parents, Todd and Connie and my brother Matt. You help me navigate all aspects of life and your support has been most important of all. Finally, I would like to thank my colleagues Julien, Loïc and Frederic. You guys have been patient with me in supporting my many research directions. Julien and Loïc are also great running buddies, humorous and down to earth, which made me feel comfortable doing research with you.

I would like to thank various agencies that have funded my research. This thesis was funded by the Centre national de la recherche scientifique (CNRS). Work was conducted at the GREYC laboratory in Normandie (UMR 6072) in association with the University of Caen Normandie (Unicaen). Financial support was also provided by the French State, managed by the French National Research Agency (ANR-17-CE39-0006).

# Contents

Chapter I

# Introduction

I found myself first captivated by research after coming across an algorithm which could recreate new images given an example [137]. This method, dating to the early 2000's, can only handle simple textural patterns but I already saw huge potential to aid the creative process. Generating data from examples is known more generally as generative modeling and over the course of my thesis I witnessed this field grow from an academic subject to a ubiquitous and indispensable tool. Systems such as ChatGPT can generate code or write essays and can aid immensely when learning new things. The system Midjourney helps democratize the artistic process by generating realistic imagery from natural language. The millions of active users of said systems are a testament to their usefulness and generality.

The seminal work in Portilla and Simoncelli [137] works by capturing how humans perceive images. This work can be dated back to the work of Bela Julesz at Bell labs in the 1970s. Julesz used computers to automatically recreate images using statistics so human observers could not tell the difference between images and their synthesized versions. This manuscript entails this process of automatic synthesis of images, or *image generation*, which we define as follows:

***Image Generation***

> Image generation is the automatic creation of *realistic* yet *novel* digital images by computer algorithms.

By automatic, we mean a user with little technical knowledge, in terms of computer science or artistry, can easily generate the image they want. By realistic, we mean a human observer thinks the image appears to be naturally created; i.e. it is a real photograph or painting created by an artist. Novelty is a harder and more subtle term to define and the automated discovery of images which lack novelty is a primary theme in this thesis. In general terms, it means an exact copy of the image does not exist within the set of images the learning algorithm used to generate the image. This work deals primarily with *deep neural networks*; these algorithms are presented data and by design try to emulate the patterns present as closely as possible.

Perhaps more than any other field in the realm of computer science, image generation has seen the most impressive advances and more recently, widespread application within society. Indeed, with the advent of widely used image generation systems like MidJourney [116], even a non savvy user can generate high resolution photographs or artwork, just by providing a text description. As with any transformative technology, as the use cases grow so do the problem areas. Namely, if a generative model generates an *exact copy* of a real image from its training set, it may infringe upon the privacy or artistic rights of the owner of the image. In the final chapter of this thesis, and as our most recent contribution, we show that said system actually does this in practice, albeit extremely rarely. Thus, to provide context and motivation

to this exposition, we begin with a non-technical introduction to generative models and their place in society in the following section. In particular, we focus on the currently unsolved issue of copyright and privacy in generative models as the primary motivating example for this manuscript.

## 1.1 Generative Models in Society

Within the field of computer science, image generation is a relatively new subject, with the first successful methods dating to the late 90s. Starting from models that learned from just a single image and generating simple images like texture, generative models have rapidly evolved into very large scale and general systems. We give a brief summary of their evolution.

**The 90s: Texture Synthesis** Starting from the seminal work in [58], this method computed primitive statistics of image features that are not learned from the image, but mathematical representations. The model in [137] extended this approach to more complex and large scale textures, such as tree bark or flowers. In the 00's, the so called copy paste methods were predominant [42, 88]. These methods do not learn, but rather permute small regions from the input image to create a "novel" output. All of these methods saw limited application in industry, mainly for the generation of texture in computer graphics applications, as they were rather limited in the domain of images they could synthesize realistically. Furthermore, they require significant user skill in terms of manipulating the algorithm for the desired outcome.

**The 2010's: Realistic Generation with Deep Networks** A major breakthrough in image generation was with the use of deep neural networks and in particular Generative Adversarial Networks (GANs) [52]. In contrast to the above methods, these methods *learn* representations of their images, which at first were datasets of thousands of images. These methods are notable as they could generate images of complex objects, such as human faces or scenes, albeit at low resolution and blurry. In the latter half of the decade, these models massively improved, to the point where human observers could no longer tell the difference between real images and those generated by these systems. Notably, this also marks some of the first serious and potentially negative ramifications image generation can have on society. For instance, with realistic generation comes the possibility of proliferating fake depictions of events, i.e. deepfakes, which are a still largely unsolved problem [27]. This decade marks more serious general discussion of AI's potential negative impact on society, and how to mitigate it, known as "AI Safety" or "AI Alignment" [5]. This is a vast and complex topic out of the scope of this manuscript, but we refer the reader to AnthropicAI's guide on AI Safety for a modern presentation [7].

**2020s: Text to Image Generation with Diffusion Models** In just the last few years, systems which are in line with our definition of image generation have become a reality. As a testament to that, publicly available image generation systems like MidJourney [116] or Stable Diffusion [163], have garnished millions of active users, most of which have no technical knowledge and create images just via a description. These models use a technique known as diffusion models (DDPMs), which are easier to train than GANs and scale to larger datasets [37]. Also notable, is that they can handle *multimodal* input; that is they can be trained with images and text simultaneously, and thus have some understanding of language. This is imperative to their widespread use, as users do not need technical knowledge, and can express their intent for generation in natural language. These models are typically much larger and trained on billions-scale data, which is typically automatically scraped from the internet. Despite their massive training sets, they are also capable of copying their data, which is a topic we'll delve into momentarily. We first point the reader to Table. 1.1, which provides an overview of the evolution of generative models and their capabilities. We also note Fig. 1.2, which demonstrates visually the stunning progress of image generation in the last few years.

| Method | Year | Quality | Ease of Use | Generality | Data | Copyright | Energy Cost |
|---|---|---|---|---|---|---|---|
| Texture Synthesis [137] | 2000 | Medium | Low | Low | One Image | None | Negligible |
| Copy-Paste [88] | 2005 | Medium | Low | Low | One Image | Medium | Negligible |
| GAN [52] | 2014 | Low | Low | Medium | Thousands | Low | ≈$10 |
| ProgGAN [78] | 2017 | Medium | Medium | Medium | $\approx 100K$ | Low | ≈ $1K |
| StyleGAN [83] | 2019 | High | Low | Medium | $\approx 100K$ | Low | ≈$1K |
| DDPM [37] | 2021 | High | Low | High | $\approx 1M$ | Low | ≈$10K |
| Latent Diffusion [144] | 2021 | High | High | High | $\approx 400M$ | High | ≈$1M |
| Stable Diffusion v2 [163] | 2022 | High | High | High | $\approx 5B$ | High | ≈$6M |
| Deep Image Floyd [157] | 2023 | Very High | Very High | Very High | $\approx 5B$ | High | >$6M |

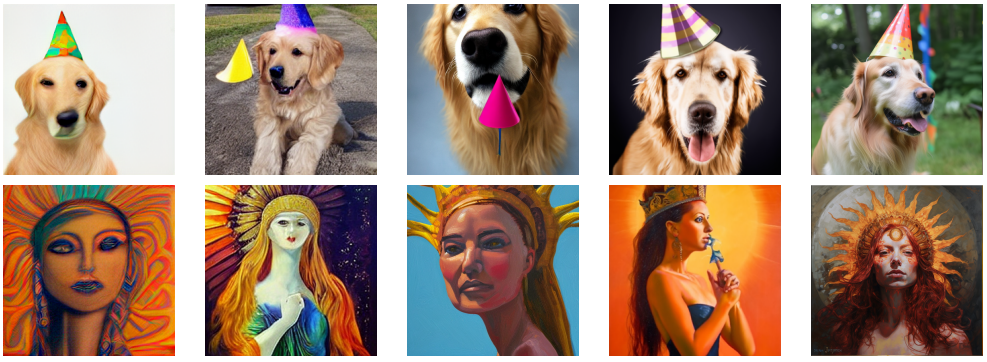
Table 1.1: Evolution of image generation over the last two decades. Image generation systems have evolved into large scale and general systems. As they've become easier to use and more widely applicable, more problem areas have arised. The last three columns refer to the number of data points needed, the copyright risk the model poses, and the energy consumption required for training. Now, these systems can have a negative impact on society. As these models have become extremely large scale, they require large distributed cloud computing. For the last several models, we estimated the cost based on current cost of cloud computing, and the training time published by the model. For the copyright risk, we give a general assessment, inspired by the discussion in Sec. 1.1.1, the experiments in this manuscript and related work. For the data points required, we give the amount used at publication (in general though, earlier models do not scale up).

### 1.1.1 Copyright in Generative Models: A Case Study

In just the last year, generative models have seen an explosion in both user interest and billions of dollars of investment money [179]. Companies such as OpenAI, StabilityAI and Midjourney have received substantial investments and spent millions of dollars training their image generation systems. OpenAI and Midjourney offer paid monthly subscriptions or offer a per image quote on their generations [116] and thus make real money from generation as part of their business model. Artists use these generation systems and generate revenue of their own, as A.I. art has become a staple of the contemporary art scene [6]. In the 2020's, sales on A.I. generated art on the NFT platform [127] grew exponentially, as image generation tools became more accessible to artists. It's no surprise that A.I. art is automating away some of the digital art creation process and even stealing the thunder of real artists. For instance, an image generated from Midjourney won a major art competition at a state fair last year [145]. Unsurprisingly, some artists are outraged and rightly so; like most diffusion models, Midjourney likely trained its model from the likes of automatically web scraped images (we are almost certain of this, see the final Chapter 7), but does not give attribution to its sources. It is no surprise then that there have been a number of class action lawsuits versus various generation systems, for instance, Getty Images has filed a class action lawsuit against StabilityAI (the company behind Stable Diffusion), under suspicion a large amount of their images were used [125]. To determine whether this case holds water, we give a brief overview of copyright law in the U.S.

**Copyright in the U.S.** Copyright law in the U.S. hinges on the idea of *fair use*, which is a set of four rather vaguely defined guidelines determining if a copyright protected work has been used fairly for creative expression [53]. It underlines four major guidelines which constitute fair use, entailing how the work was used, whether the use generates revenue, how much of the work was used and so on. Importantly, it states that if the use is *transformative*, then it is likely considered fair; uses which alter the meaning or express something different from what was present in the original work are likely considered fair. Within the art world, even keeping the work largely intact, so long as it conveys a new meaning, is a common practice known as appropriation [166]. In practice, copyright cases are determined on a case by case basis,

and the deliberate vagueness of fair use can often be a strong point in protecting creative expression [46].



| Aug. 2021: DALLE-Mini [43] | Dec. 2021: Latent Diffusion [144] | Dec. 2022: Stable Diffusion v2.0 [163] | Deep Image Floyd: April 2023 [157] | Midjourney (latest): April 2023 [116] |
|---|---|---|---|---|

Table 1.2: Fast progress of text to image models in the 2020's. In just a few years, text to image methods have evolved into large scale, general purpose, high quality and widely used image generation systems. These images are all synthesized by text prompts: for the top row "Photograph of a golden retriever with a party hat on," and bottom, "The queen of the sun, oil on canvas." Whilst there are many applications, there are societal problems as well, see Sec. 1.1.1 and Table 1.1.

**Copyright in Generative Models** Within the context of machine learning, enforcing copyright law has been largely unsuccessful [46]. This is because most machine learning applications like retrieval or classification only express high level or meta trends within the data. This is considered a non-expressive representation, which is an important distinction commonly used to determine fair use today [123]. Generative models on the other hand can potentially be considered to use their training data expressively [46].

**When is Text-to-Image Fair Use?** As was mentioned, text-to-image generative models require billions scale datasets, such as LAION-2B [151], which were created by automatically scraping the internet. Incidentally, a large portion of these datasets are copyrighted and contain large subsets of art and creative work. Typically, this is in the form of a Creative Commons license, most of which require attribution, which the most popular text-to-mage systems do not provide with their generations.

Users do have some amount of creative expression when interacting with such systems. Typically, this is an iterative process; a user has a desired prompt and if the image is not satisfactory, they then modify it and try again, a process known as "prompt engineering". When it comes to stylizing images, users often provide the name of the artist whose style they wish to emulate. Figure 1.1(b) shows such an example of a commonly used style from several artists, and a Midjourney generation with their name included in the prompt.

**Enforcement** Even if one had strong suspicion their works were used, there is the issue of how to actually enforce your copyright case. Take for instance, the case of Getty Images versus StabilityAI [125]. In this case, it is fairly easy to show that many of Getty Images copyrighted works appear in the dataset used to train Stable Diffusion, as the dataset is public (for instance by using a semantic retrieval, presented in chapter 7). However, in terms of fair use in the U.S., it would also be necessary to prove the "substantiality" on the work used and whether generations are transformative. In the Getty Images case, they claim their logo appears in generations, which is likely not substantial enough. For systems like Midjourney, even the training set is unknown and proving your work was used is known as *membership inference*.

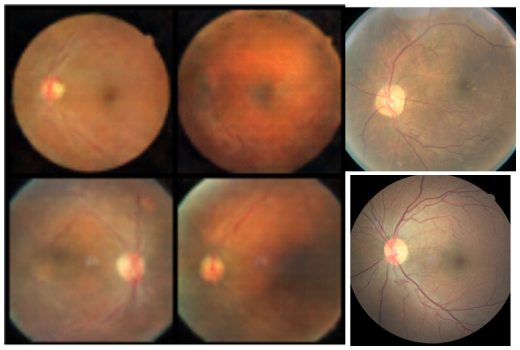
(a) Appropriation: Original on the left and modified version on the right.

(b) Midjourney can emulate an artists works (Van Gogh on left and Edvward Munch on right).

(c) An extraction attack versus a Diffusion Model. Exact copies like this pose copyright issues.

(d) Part of the image has varied in the generation, but it is not transformative.

Figure 1.1: Copying phenomena in generative models. In the top left, the "American Gothic" picture has been largely re-used: this is fair use as it is transforms the meaning the image conveys. In the top right, we demonstrate Midjourney can replicate an artists style (left vincent van gogh and right The Scream by Edvard Munch). Most of the time, users input a living artists name and get generations similar to their content. It's unclear whether this is fair use, as the artist's content was scraped without permission and the generation may not be transformative. In the bottom left, we show a malfunction of a Diffusion Model, which exactly copies inputs. In the last chapter we explore efficient and reproducible ways to study this phenomena. On the bottom right, we show another malfunction where the image is modified, but not substantially.



(a) Membership inference versus medical images [55]. Left two columns are generated, classified to be in the training set. Right column is real images.

(b) Membership inference versus identities, explored in Chapter 6. Left column is generated, right two columns real.

Figure 1.2: Membership inference versus GAN generators. Here, we highlight two applications where privacy may be important. On the left are medical images generated by a GAN generator, which have been inferred to be in the training set. Such attacks hinder the utility of generative models use in practice, for instance in the medical domain where it's imperative data stay private.

**Membership Inference** Membership inference is the process of determining which samples were used for training a model [155, 24].

There are several settings: the *white-box* setting assumes the party performing inference (referred to as the attacker), has the model and some of its training set, but does not know exactly which samples were used. This setting is akin to the one in the GettyImages case [125]. The *black-box* setting assumes the attacker does not have the model and some of its training data, and may only query the model's outputs. This would be the case for users of Midjourney. Membership attacks typically exploit systematic bias in a model's output. They can demonstrate that an output is highly biased for a suspected training sample, in a way that is not merely up to chance.

We study membership attacks starting from Chapter 4 and for the remainder of the thesis. However, since publication, many works have improved both attack efficacy and defenses. Several approaches have made breakthroughs in Differential Privacy (DP) [41, 169, 186], which can provide mathematical guarantees that no information leaks from training. For instance, the very recent work in [51] is amongst the first method to generate useful sample with DP using diffusion models, for use on downstream vision tasks. We study a similar problem in Chapter 4, but with empirical verification rather than DP. Still, these methods have much lower quality than non-DP training and still provide "trivial" privacy bounds; in other words as DP only provides an upper bound on the information that leaks. In practice these upper bounds are too weak and many acceptable DP parameters are still empirically vulnerable to membership inference [71]. Unlike membership attacks, extraction attacks verse generative models like DDPMs are a more cutting edge area of research.

**Extraction Attacks** Extraction are a stronger form of membership inference wherein training samples are reconstructed from model outputs [26, 24]. For instance, for text-to-image models, only the prompt is provided and the generation process begins with random noise. For the vast majority of prompts, the outputs will all be slightly different and unlike any image in the training set (i.e. fulfilling the novelty requirement of our definition of image generation). In recent work [24, 178], and the final chapter of this work, it is shown that certain prompts can generate pixel-for-pixel copied images in the training set. This does not happen by chance, and is proof this image was used for training. See Fig. 1.1, bottom row for a visual reference.

**Privacy Perspective** The problems discussed thus far with copyright largely also apply to privacy in machine learning. That is, generated models vulnerable to membership inference pose can often pose copyright and privacy concerns simultaneously. Generative models are often trained on medical data [172, 185]. In this setting, due the tight legal restrictions on the use of medical data, it is required that any machine learning model doesn't leak information on its training set. Nonetheless, generative models have been widely applied to medical images in particular, because they are able to learn useful representations without supervision (i.e. human annotation of images). Generated images may even provide privacy, and thus allow for the release and aggregation of medical data. We explore in Chapter 4 and Chapter 5 that GAN images appear to have some empirical privacy properties in the proper training scenarios. Fig. 1.2 shows several examples of generated images and inferred membership for several domains where privacy may be important.

### 1.1.2 Technical Advances

In this section, we'll hone in on a more detailed view of advances that lead to high quality generation systems. Whilst some problems have essentially been solved (for instance, realistic generation), there are still some unsolved technical problems and shortcomings for today's models that we'll highlight, before exposing the outline of the thesis.

**Advances in Training** GANs are trained adversarially, that is there are two networks with opposing objectives: the discriminator network tries to discern generated images from training data and the other

tries to fool this network by generating realistic datum. Without any constraints, this problem is inherently unstable and early GANs were notoriously hard to train. The first successful methods, such as deep convolutional GAN (DCGAN) [139], would often diverge on complex datasets. Thus, the first major advances in image generation in GANs were modifications of the training objective, rather than the architecture. Methods such as Wassertein GAN (WGAN) [8] or Mescheder et al.(MESCH) [112], offer theoretical insight which suggest to constrain the discriminator network during training. Indeed, when unconstrained, the discriminator often "wins," early in training and hinders the ability of the generator to learn. In (MESCH), the discriminator is directly penalized and stabilizes training enough to handle complex datasets such as ImageNet.

**Architectural Advances** In contrast to training, architectural advancements pertain to the design of the network itself. Several prominent advances led to an improvement in quality. Models such as StlyeGAN, ProGAN and Mescheder implemented residual layers, which allowed them to scale to larger size. Early models used convolutions, which only use local information in each layer. Later models, such as StyleGAN or BigGAN, incorporated global information in each layer beyond just convolutions. Likewise, DDPM models use attention layers, which also incorporate global information. Both of these techniques typically lead to better control during generation.

**Evaluation Metrics** Whilst visual inspection can easily tell a low quality model from a high quality one, assessing overall performance on high quality and large generators requires automated analysis. Progress in generative models has been partially advanced by such metrics, which help objectively compare quality between models. Early metrics (studied in Chapter 2) could measure quality in line with human perception. Later, metrics were developed to assess diversity of generated samples as well as their quality. Several metrics today are used to measure quality on generation tasks not seen during training. For text-to-image systems, quality is measured through generating from text prompts not seen during training. Finally, membership inference attacks can also be seen as a type of automated assessment.

**Training with Low Data** Another challenge addressed later in the 2010s was generation with few samples. This setting is important as users often want to generate specific datum not in standard benchmarks. Several approaches were proposed around 2019, which augment training data specifically for GAN training, such as DiffAugment or StyleGAN-ADA [195, 80]. These methods also employ transfer learning, which is re-using a model, or slightly modifying it to train on new data. These techniques are studied in Chapter 5, but with constraints on the number of parameters learned. More in this vein, for DDPMs, methods such as Textual Inversion [49] are capable of training on just a few samples, whilst only making a small modification to a pre-trained DDPM model. Also successful are the LoRA methods [61], which is the most popular way to train specific stable diffusion checkpoints [32]. These methods synthesize with the help of a small "low rank" network. In other words, the original model is unchanged, which is quite similar to our goal in Chapter 5. For GANs, hyper networks serve a similar purpose, in only modifying a small part of the network for synthesis with few samples [4].

### 1.1.3 Outstanding Challenges

In light of the progress and challenges we've underlined thus far, we summarize with three outstanding challenges, which are both relevant to this thesis and could be expanded for future work:

(C1) Efficient and general membership attacks against large scale generative models.

(C2) Image generation systems trained to be robust against membership attacks, rather than just for quality alone.

(C3) Learning generative models with few parameters and few samples.

For (C1), we emphasize the challenge is with large scale systems and with efficiency. Indeed, several attacks presented in the beginning of the thesis are not well adapted for the billions scale datasets presented in the last chapter. When considering a large volume of training samples, membership attacks need to be both general and extremely efficient. We note that whilst this seems like a negative goal, it's necessary for verifying the safety of deployed systems. It can go hand in hand with the second objective, for instance by providing empirical verification for systems trained with privacy guarantees. Challenge (C3) would significantly aid current generation systems for creative content. This is also linked to (C2) and would be beneficial to both companies and users. For content creators, systems like Midjourney could redistribute revenue to the correct creators whilst mitigating copyright risks.

### 1.1.4 Outline

The chapters in this thesis, which are based on accepted or submitted papers reproduced largely as is. Thus, they are made to be self contained and contain related work therein. At the outset of each chapter, we give its context and relations within the manuscript and also with respect to work that has been published since. There are connections and themes throughout, which we highlight now. In Chapter 2, we introduce evaluation of generative model quality. In Chapter 3 we diagnose overfitting in GANs using latent recovery and measure a model's bias towards generating training samples. This can be used to create a membership attack, which we then explore in Chapter 4. In this chapter, we make progress towards (C1) by investigating what circumstances lead to vulnerabilities to membership inference. In Chapter 5, we explore the applications of parameter efficient learning, transfer learning and generation with low data, making progress on (C3). Furthermore, we show that learning with few parameters also makes progress on (C2) by providing a trade-off between quality and privacy. In Chapter 6 we revisit membership attacks and investigate how identities can be inferred from generated images by designing a new attack paradigm. Finally, in Chapter 7, we design a large scale extraction attack against real world diffusion models and make contributions to (C1).

### List of publications

[a] L. Simon, R. Webster, and J. Rabin. Revisiting precision recall definition for generative modeling. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5799–5808, 2019.

[b] R. Webster, J. Rabin, L. Simon, and F. Jurie. Detecting Overfitting of Deep Generative Networks via Latent Recovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[c] R. Webster, J. Rabin, L. Simon, and F. Jurie. Generating Private Data Surrogates for Vision Related Tasks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 263-269, 2021. doi:10.1109/ICPR48806.2021.9413067

[d] R. Webster, J. Rabin, L. Simon, and F. Jurie. Width-Wise Parameter Sharing for Multi-Domain Gan Learning. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 4163-4167, 2022. doi:10.1109/ICIP46576.2022.9897423

#### Under Review

[a] R. Webster, J. Rabin, L. Simon, and F. Jurie. This Person (Probably) Exists. Identity Membership Attacks Against GAN Generated Faces. *arXiv preprint arXiv:2107.06018*, 2021.

[b] R. Webster, J. Rabin, L. Simon, and F. Jurie. On the De-duplication of LAION-2B. *arXiv preprint arXiv:2303.12733*, 2023.

[c] R. Webster. A Reproducible Extraction of Training Images from Diffusion Models. *arXiv preprint arXiv:2305.08694*, 2023.

Chapter 2

# Evaluating Quality with Precision Recall Curves

This chapter fits chronologically first within the work conducted during this manuscript. We also felt it the most appropriate to expose first, as the majority of this thesis deals with evaluating generative models, we introduce concepts used throughout, for instance the idea of viewing globally the entire set of generated images as a distribution, and evaluating closeness of two distributions as a proxy for model quality. At the time of writing, few tools were available for assessing failure modes of generated distributions. The predominant tool at the time, which has continued to be widely used today for its simplicity, is the Fréchet Inception Distance [59], which we define at the outset of the chapter. The FID summarizes the closeness of two distributions of images. Normally, these distributions are a real distribution of images and a generated one. A low enough FID often corresponds to the inability of human observers to tell the difference between real and fake; the generated distribution is thus called realistic, or plausible. Rather than providing a scalar for generative quality like the FID, PR curves distinguish so called mode-collapse (poor recall) and bad quality (poor precision). We first generalize their formulation to arbitrary measures, hence removing any restriction to finite support. We also expose a bridge between PR curves and type I and type II error rates of likelihood ratio classifiers on the task of discriminating between samples of the two distributions. Building upon this new perspective, we propose a novel algorithm to approximate precision-recall curves, that shares some interesting methodological properties with the hypothesis testing technique from [101]. We demonstrate the interest of the proposed formulation over the original approach on controlled multi-modal datasets.

As we'll study more in Chapter 5, generative models trained with constraints (in this case, with few parameters and data points), have specific trade-offs between their constraints and realism. For instance, we'll see that GANs often favor realism; images will appear realistic, but many images appearing in the real distribution will never be generated. In Sec. 5.4, we see this happen in practice. FID will not differentiate this failure mode, and thus provides a less clear global picture of model quality. In the next chapter, we'll delve into other shortcomings of the FID, specifically that is not well adapted to models that memorize training data.

**Continued Work** Many works have since built upon PR curves in generative models since this work was published. For instance, in [89], PR curves are constructed with a nearest neighbor classifier, rather than a classifier trained with a neural network. In many cases, these leads to a more accurate and flexible way to compute PR curves. For large scale datasets like LAION-2B, efficient libraries for nearest neighbor computation exist [74], and thus PR curves can be in theory calculated for large scale generative models. We explore using approximate nearest (ANNs) neighbors in Chapter 7 and methods which use nearest neighbors for PR curves can likely be scaled to larger datasets [89]. Whilst out of the scope of this manuscript, we have found many close links between the PR curves presented here and a variety of statistical measures in the literature [159]. For instance, it can be shown that the PR curves in this chapter,

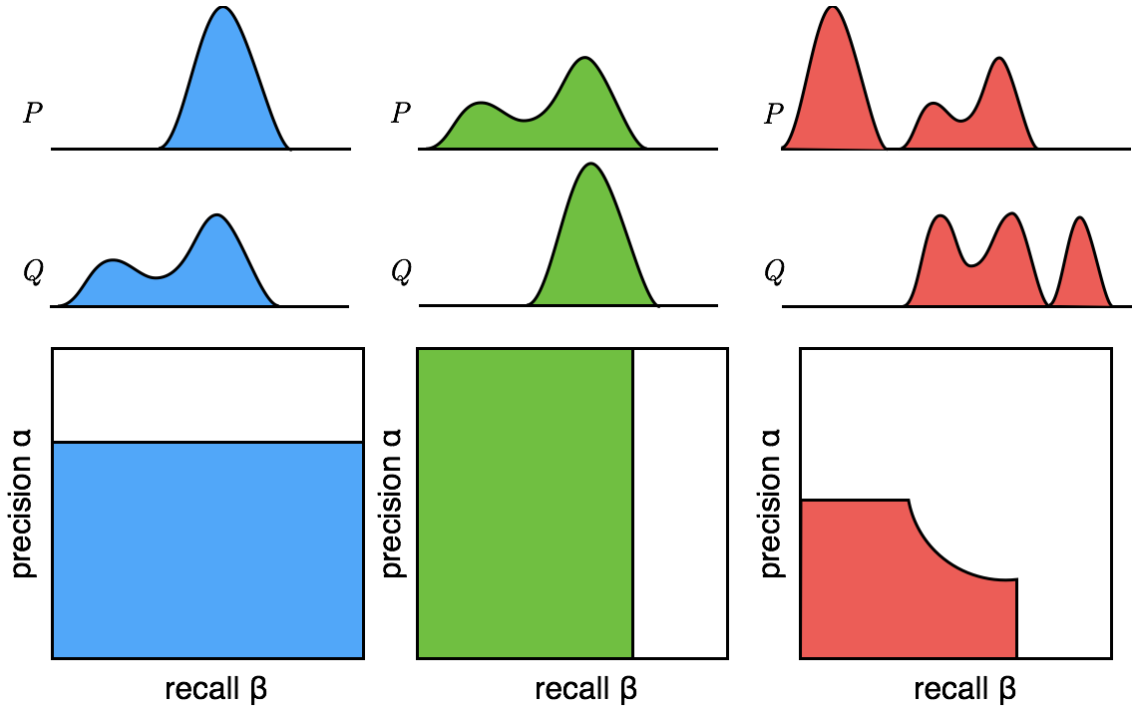



Figure 2.1: Illustration of precision-recall curves for multi-modal continuous distributions $P$ and $Q$. Left: mode invention (precision is only partial but full recall). Middle: mode dropping (partial recall) but do not produce outliers (full precision but partial recall). Right: mode dropping / invention plus mode reweighting.

can be converted to the Lorenz curves found in [173].

## 2.1 Introduction

This chapter addresses the question of the evaluation of generative models, such as Generative Adversarial Networks (GAN) [52] or Variational Auto-Encoders [87], that have attracted a lot of attention in the last years. These approaches aim at training a model to generate new samples from an unknown target distribution $P$, for which one has only access to a (sufficiently large) sample set $X_i \sim P$. While this class of methods have given state-of-the-art results in many applications (see *e.g.* [21] for image generation, [65] for inpainting, *etc*), there is still a need for evaluation techniques that can automatically assess and compare the quality of generated samples $Y_i \sim Q$ from different models with the target distribution $P$, for which the likelihood $P(Y_i)$ is unknown. Most of the time, such a comparison is just reduced to a simple visual inspection of the samples $Y_i$, but very recently several techniques have been proposed to address this problem that boils down to the comparison of two empirical distributions in high dimension. While generative models have seen successful applications far beyond just image data (such as speech enhancement [134], text to image synthesis [142] or text translation [90]), we will focus on image generation, as is popular in the recent literature.

**Related work** When it comes to evaluating generative models of images, visual inspection, that is, observing how "realistic" the images appear, remains the most important decider of the model's success. Indeed, state-of-the-art methods, such a Progressive GANs [79] on face images or BigGAN [21] trained conditionally on ImageNet classes, include large grids of generated samples wherein the success of the method over previous approaches is visually obvious. Nonetheless, automatic evaluation of such models is extremely important, for example when conducting large scale empirical comparisons [102], in cases

where model failure is more subtle than simply poor image quality (*e.g.* mode collapse) such as in [148], or presumably in domains in which humans are less attuned to discern quality of samples.

Attempts to provide automatic assessment of image quality can be traced back to the first GAN methods [140], where the authors assessed quality of generated samples with a nearest neighbor classifier. In [149], the so-called *Inception Score* was introduced, which analyzes the entropy of image classes at the output of the Inception Network [164], which reflects if samples cover all classes and each clearly belongs to a particular class. In [114] and in the subsequent chapter, test set samples (*i.e.* those unseen during training), are recovered via optimization. Successful generators are better at recovering all images from the training distribution, which in a controlled setting can be viewed as a notion of recall [102, 148].

**Fréchet Inception Distance** As we mentioned at the outset of the chapter, the Fréchet Inception Distance (FID) is a widely used metric that provides a scalar value assessing model quality. The FID models the real and generated distributions as multivariate gaussians in Inception V3 feature space [164]. Letting $\mu_P, \Sigma_P$ and $\mu_Q, \Sigma_Q$ denote the mean and covariances of the real and generated distributions, the FID is the 2-Wasserstein distance between these two distributions. For multivariate distributions, this can be written in closed form as

$$\text{FID}(P, Q) = ||\mu_P - \mu_Q||^2 + \text{Tr}(\Sigma_P + \Sigma_Q - 2(\Sigma_P \Sigma_Q)^{\frac{1}{2}}) \tag{2.1}$$

The FID has been widely adopted because of its consistency with human inspection and sensitivity to small changes in the real distribution (*e.g.* slight blurring or small artifacts in generated images). A few recent approaches involve training a binary classifier to separate fake (*i.e.* generated) samples $Y_i$ from real data samples $X_i$. In [101], a score is defined from a two-sample statistical test of the hypothesis $P = Q$. Finally, in [69], classifiers trained with various divergences (normally used as objectives for discriminators during GAN training) are used to define a metric between $Q$ and $P$. Surprisingly, successful models such as WGAN [9] have the smallest distance even on those metrics which were not used for training (*e.g.* a WGAN trained with the Wasserstein-1 distance evaluated with a least squares discriminator).

Unfortunately as pointed out by [148], the popular FID only provides a scalar value that cannot distinguish a model $Q$ failing to cover all of $P$ (referred to henceforth as *low recall*) from a model $Q$ which has poor sample quality (referred to as low precision). For example, when modeling a distribution of face images, a $Q$ containing only male faces with high quality versus a $Q$ containing both genders with blurry faces may have equal FID. Following the lead of [148], we will consider another category where one wants not only to assess if the samples are of good quality (high precision) but also to measure if the generated distribution $Q$ captures the variability of the target one (high recall). The reader may refer to Figure 2.1 to gain a crude understanding of the intended purpose of precision and recall. [148] proposed an elegant definition of precision and recall for discrete distributions. They challenge their definition on image generation by discretizing the probability distributions $P$ and $Q$ over Inception features via K-means clustering. Note that a similar notion was proposed by the authors of PACGAN [95] under the name of mode collapse region (denoted as $MCR(P, Q)$). Their motivation was to develop a theoretical tool to analyze how using multi-element samples in the discriminator can mitigate mode dropping.

**Contributions and outline** The chapter is organized as follows. First, Section 2.2 recalls usual notations and some definitions from measure theory. Then, we expose the main contributions of this chapter:

- A first limit of [148] is the restriction to discrete probability distributions (*i.e.* considering that samples live in a finite state space $\Omega$). In Section 2.3, this assumption is dispensed by defining Precision-Recall curves from *arbitrary* probability distributions for which some properties are then given;
- In the original work of [148] the Precision-Recall curves approach was opposed to the hypothesis testing techniques from [101]; we demonstrate in Section 2.4 that precision and recall are actually

linear combinations of type I (the false positive rate) and type II (the false negative rate) errors of optimal likelihood ratio classifiers, and give as well some upper-bound guarantee for the estimation of Precision-Recall curves with non-optimal classifiers; Besides, our formulation also exhibits a relationship with the MCR notion proposed by [95] which turns out to be the ROC curves (1−type I versus type II errors) for optimal classifiers;

- Section 2.5 details the proposed algorithm to estimate Precision-Recall curves more accurately; the clustering optimization step used in the original method is now simply replaced by the training of a classifier which learns to separate samples from the two datasets;
- The experimental Section 2.6 demonstrates the advantage of the proposed formulation in a controlled setting using labelled datasets (CIFAR10 and ImageNet categories), and then shows its practical interest for evaluating state-of-the art generative image models.

## 2.2 Notions from standard measure theory

We start these notes by recalling some standard notations, definitions, and results of measure theory. For the remainder, $(\Omega, \mathcal{A})$ represents a common measurable space, and we will denote $\mathcal{M}(\Omega)$ the set of signed measures, $\mathcal{M}^+(\Omega)$ the set of positive measures and $\mathcal{M}_p(\Omega)$ the set of probability distributions over that measurable space.

**Definition 1.** *Let $\mu, \nu$ two signed measures. We denote by*

- $\text{supp}(\mu)$*, the support of $\mu$ (technically, such a notion is defined up to null sets);*
- $\frac{d\mu}{d\nu}$*, the Radon-Nykodim derivative of $\mu$ w.r.t. $\nu$;*
- $|\mu|$*, the total variation measure of $\mu$;*
- $\mu \wedge \nu = \min(\mu, \nu) := \frac{1}{2}(\mu + \nu - |\mu - \nu|)$ *(a.k.a the measure of largest common mass between $\mu$ and $\nu$ [135]).*

The extended half real-line is denoted by $\overline{\mathbb{R}^+} = \mathbb{R}^+ \cup \{\infty\}$.

**Theorem 1** (Hahn decomposition)**.** *Let $\mu \in \mathcal{M}(\Omega)$. Then there exists an essentially unique partition $\Omega = \Omega_\mu^+ \sqcup \Omega_\mu^-$ (*i.e. *where $\Omega_\mu^+ \cap \Omega_\mu^- = \varnothing$) such that $\forall A \in \mathcal{A}$:*

$$\begin{aligned} A \subset \Omega_\mu^+ &\Rightarrow \mu(A) \geq 0 \\ A \subset \Omega_\mu^- &\Rightarrow \mu(A) \leq 0 \end{aligned} .$$

**Corollary 1.** *Let $\mu, \nu \in \mathcal{M}^+(\Omega)$. Then, $\forall A \in \mathcal{A}$, we have:*

$$(\mu \wedge \nu)(A) = \mu(A \cap \Omega_{\mu-\nu}^-) + \nu(A \cap \Omega_{\mu-\nu}^+).$$

## 2.3 Precision-Recall set and curve

We follow [148] for the definition of the Precision-Recall (PR) set that we extent to any arbitrary pair of probability distributions $P$ and $Q$, up to two additional minor changes. First, we have tried to adapt their definition in a shorter form. Second, we include the left and lower boundaries in the PR set.

**Definition 2.** *Let $P, Q$ two distributions from $\mathcal{M}_p(\Omega)$. We refer to the Precision-Recall set $PRD(P, Q)$ as the set of Precision-Recall pairs $(\alpha, \beta) \in \mathbb{R}^+ \times \mathbb{R}^+$ such that*

$$\exists \mu \in \mathcal{M}_p(\Omega), P \geq \beta\mu, Q \geq \alpha\mu \ . \tag{2.2}$$

The *precision* value $\alpha$ is related to the proportion of the generated distribution $Q$ that match the true data $P$, while conversely the *recall* value $\beta$ is the amount of the distribution $P$ that can be reconstructed from $Q$. Therefore, in the context of generative models, one would like to have admissible precision-recall pairs that are as close to $(1,1)$ as possible. One can then easily show the following properties:

**Theorem 2.** *Let $P, Q$ two distributions from $\mathcal{M}_p(\Omega)$. Then,*

1. $(0,0) \in PRD(P,Q) \subset [0,1] \times [0,1]$;
2. $P = Q \Leftrightarrow (1,1) \in PRD(P,Q)$;
3. $(\alpha,\beta) \in PRD(P,Q)$ *and* $\alpha' \leq \alpha, \beta' \leq \beta$ *implies that* $(\alpha',\beta') \in PRD(P,Q)$.

Because of the lack of natural order on $[0,1] \times [0,1]$, no point of $\mathrm{PRD}(P,Q)$ is strictly better than all the others. Yet, the singular importance of $(1,1)$ should draw our attention to the Pareto front of $\mathrm{PRD}(P,Q)$ defined as follows.

**Definition 3.** *The precision recall-curve $\partial PRD(P,Q)$ is the set of $(\alpha,\beta) \in PRD(P,Q)$ such that*

$$\forall (\alpha',\beta') \in PRD(P,Q), \alpha \geq \alpha' \text{ or } \beta \geq \beta'.$$

In fact, this frontier is a curve for which [148] have exposed a parameterization. We generalize their result here (dropping any restriction to discrete probabilities).

**Theorem 3.** *Let $P, Q$ two distributions from $\mathcal{M}_p(\Omega)$ and $(\alpha,\beta)$ positive. Then, denoting*

$$\forall \lambda \in \overline{\mathbb{R}^+}, \begin{cases} \alpha_\lambda := ((\lambda P) \wedge Q)(\Omega) \\ \beta_\lambda := \left(P \wedge \frac{1}{\lambda} Q\right)(\Omega) \end{cases} \tag{2.3}$$

1. $(\alpha,\beta) \in PRD(P,Q)$ *iff* $\alpha \leq \alpha_\lambda$ *and* $\beta \leq \beta_\lambda$ *where* $\lambda := \frac{\alpha}{\beta} \in \overline{\mathbb{R}^+}$.
2. *As a result, the PR curve can be parameterized as:*

$$\partial PRD(P,Q) = \{(\alpha_\lambda, \beta_\lambda) / \lambda \in \overline{\mathbb{R}^+}\}. \tag{2.4}$$

*Proof.* The second point derives easily from the first which we demonstrate now. Let $(\alpha,\beta)$ positive and $\lambda := \frac{\alpha}{\beta}$. By definition $(\alpha,\beta) \in \mathrm{PRD}(P,Q)$ iff $\exists\, \mu \in \mathcal{M}_p(\Omega)$

$$P \geq \beta\mu = \frac{\alpha}{\lambda}\mu \text{ and } Q \geq \alpha\mu$$

iff

$$\mu \leq \frac{1}{\alpha}(\lambda P \wedge Q) = \frac{1}{\beta}(P \wedge \frac{Q}{\lambda})$$

which yields the expected criteria given that $\mu(\Omega) = 1$. □

## 2.4 Link with binary classification

Let us consider samples $(X_i, Y_i) \sim P \times Q$ and as many Bernoulli variables $U_i \sim \mathcal{B}_{\frac{1}{2}}$. And let $Z_i = U_i X_i + (1-U_i) Y_i$. Then $Z_i \sim \mathbb{P}_Z$ follows a mixture of $P$ and $Q$, namely $\mathbb{P}_Z = \frac{1}{2}(P+Q)$. Then, let us consider the binary classification task where from $Z_i$, one should decide whether $U_i = 1$ (often referred to as the null hypothesis). We show that the precision-recall curve can be reinterpreted as mixed error rates of binary classifiers obtained as likelihood ratio tests (hence the most powerful classifiers according to the celebrated Neyman-Pearson lemma).

**Theorem 4.** *Let $\lambda \geq 0$. Let $Z = UX + (1-U)Y$ where $(X, Y, U) \sim P \times Q \times \mathcal{B}_{\frac{1}{2}}$. Defining the likelihood ratio classifier $\tilde{U}$ as the following indicator function*

$$\tilde{U}(Z) := \mathbb{1}_{\lambda \frac{dP}{d\mathbb{P}_Z}(Z) \geq \frac{dQ}{d\mathbb{P}_Z}(Z)}\,, \tag{2.5}$$

$$\textit{then,} \quad \alpha_\lambda = \lambda \mathbb{P}(\tilde{U} = 0 | U = 1) + \mathbb{P}(\tilde{U} = 1 | U = 0)\,.$$

*Proof.* Note that we can reformulate $\tilde{U}$ as $\tilde{U}(Z) = \mathbb{1}_{\Omega^+_{\lambda P - Q}}(Z)$. Then,

$$\begin{aligned}\mathbb{P}(\tilde{U} = 1 | U = 0) &= \int_\Omega \mathbb{1}_{\Omega^+_{\lambda P - Q}}(z) d\mathbb{P}_Z(z | U = 0) \\ &= \int_\Omega \mathbb{1}_{\Omega^+_{\lambda P - Q}}(z) dQ(z) = Q(\Omega^+_{\lambda P - Q})\end{aligned}\,.$$

Now, using $\mathbb{1}_{\tilde{U}=0} = \mathbb{1}_{\Omega^-_{\lambda P - Q}}$, we have similarly $\mathbb{P}(\tilde{U} = 0 | U = 1) = P(\Omega^-_{\lambda P - Q})$. Combining the two errors, we get

$$\lambda \mathbb{P}(\tilde{U} = 0 | U = 1) + \mathbb{P}(\tilde{U} = 1 | U = 0) = (\lambda P \wedge Q)(\Omega) = \alpha_\lambda$$

where we have used Corollary 1. □

The previous protocol demonstrates that points on the PR curve are actually a linear combination of type I error rate (probability of rejection of the true null hypothesis $\mathbb{P}(\tilde{U} = 0 | U = 1)$) with type II error rate ($\mathbb{P}(\tilde{U} = 1 | U = 0)$). It also shows that if one is able to compute the likelihood ratio classifier, then one could virtually obtain the precision-recall curve $\partial\mathrm{PRD}(P, Q)$. Unfortunately, in practice the likelihoods are unknown. To alleviate this set-back, one can argue like [111] that optimizing standard classification losses is *in fine* equivalent to minimize a Bregman divergence to the likelihood ratio. Besides, we are going to show that using Eq. (2.4) with any other classifier always yields an over-estimation of $\alpha_\lambda$ and $\beta_\lambda$. To do so, we will need the following lemma, which is merely a quantitative version of the Neyman-Pearson Lemma.

**Lemma 1.** *Let $\tilde{U}(Z)$ be the likelihood ratio classifier defined in Eq. (2.5), associated with the ratio $\lambda$. Then, any classifier $U'(Z)$ with a lower type II error, that is such that*

$$\mathbb{P}(U' = 1 | U = 0) \leq P(\tilde{U} = 1 | U = 0)\,,$$

*undergoes an increase of the type I error such that*

$$\begin{cases} \alpha'_\lambda := \lambda P(U' = 0 | U = 1) + P(U' = 1 | U = 0) & \geq \alpha_\lambda \\ \beta'_\lambda := P(U' = 0 | U = 1) + \frac{1}{\lambda} P(U' = 1 | U = 0) & \geq \beta_\lambda \end{cases}$$

*Proof.* The proof is similar to the classical proof of the Neyman-Pearson lemma (see Appendix **??**). □

**Theorem 5.** *Let $\tilde{U}$ the likelihood ratio classifier from Eq. (2.5) associated with the ratio $\lambda$, and let $U'$ be any other classifier. Using precision-recall pair $(\alpha'_\lambda, \beta'_\lambda)$ defined in Lemma 1, we have that*

$$\alpha'_\lambda \geq \alpha_\lambda \textit{ and } \beta'_\lambda \geq \beta_\lambda$$

*Proof.* The proof uses Lemma 1 and its symmetric version (obtained by swapping the role of type-I and type-II errors). Three cases may arise:

1. If $\mathbb{P}(U' = 0 | U = 1) \geq \mathbb{P}(\tilde{U} = 0 | U = 1)$ and $\mathbb{P}(U' = 1 | U = 0) \geq \mathbb{P}(\tilde{U} = 1 | U - 0)$ then the conclusion of the theorem is trivially true;
2. If $\mathbb{P}(U' = 1 | U = 0) \leq \mathbb{P}(\tilde{U} = 1 | U = 0)$, then the conclusion is ensured by Lemma 1;
3. If $\mathbb{P}(U' = 0 | U = 1) \leq \mathbb{P}(\tilde{U} = 0 | U = 1)$, then one should use the symmetric version of Lemma 1.

□

## 2.5 Algorithm

Based on the above analysis, we propose the Algorithm 1 to estimate (via the function `estimatePRCurve`) the Precision-Recall curve of two probability distributions known through their respective sample sets.

**Inputs:** Dataset of target/source sample pairs:
$\mathcal{D} = \{(X_i, Y_i) \sim P \times Q \text{ i.i.d}/i \in \{1, \dots, N\}\}$,
Parameterization of the PR curve: $\Lambda = \{\lambda_1, \dots, \lambda_L\}$
**Output:** $\partial\text{PRD}_\Lambda \simeq \{(\alpha_\lambda, \beta_\lambda)/\lambda \in \Lambda\}$
**Algorithm** `estimatePRCurve(`$\mathcal{D}$, $\Lambda$`)`

$\mathcal{D}^{train}, \mathcal{D}^{test}$ =`createTrainTest(`$\mathcal{D}$`)`
$f$ =`learnClassifier(`$\mathcal{D}^{train}$`)`
$\partial\text{PRD}_\Lambda$ =`estimatePRD(`$f$, $\mathcal{D}^{test}$, $\Lambda$`)`
**return** $\partial\text{PRD}_\Lambda$

**Procedure** `createTrainTest(`$\mathcal{D}$`)`

$\mathcal{D}^{train} = \varnothing, \mathcal{D}^{test} = \varnothing$
**for** $i \in \{1, \dots, N\}$ **do**
  $U_i \sim \mathcal{B}_{\frac{1}{2}}$
  $Z_i^{train} = U_i X_i + (1 - U_i) Y_i$
  $Z_i^{test} = (1 - U_i) X_i + U_i Y_i$
  $\mathcal{D}^{train} \leftarrow \mathcal{D}^{train} \cup \{(Z_i^{train}, U_i)\}$
  $\mathcal{D}^{test} \leftarrow \mathcal{D}^{test} \cup \{(Z_i^{test}, 1 - U_i)\}$
**end**
**return** $\mathcal{D}^{train}, \mathcal{D}^{test}$

**Procedure** `estimatePRD(`$f$, $\mathcal{D}^{test}$, $\Lambda$`)`

$fVals = \{f(z)/(z, u) \in \mathcal{D}^{test}\}$
$errRates = \varnothing$
$N_j = \left|\{(z, u) \in \mathcal{D}^{test}/u = j\}\right|$, for $j \in \{0, 1\}$
**for** $t \in fVals$ **do**
  $\text{fpr} = \frac{1}{N_1} \left|\{(z, u) \in \mathcal{D}^{test}/f(z) < t, u = 1\}\right|$
  $\text{fnr} = \frac{1}{N_0} \left|\{(z, u) \in \mathcal{D}^{test}/f(z) \geq t, u = 0\}\right|$
  $errRates \leftarrow errRates \cup \{(fpr, fnr)\}$
**end**
$\partial\text{PRD}_\Lambda = \varnothing$
**for** $\lambda \in \Lambda$ **do**
  $\alpha_\lambda = \min(\{\lambda\text{fpr} + \text{fnr}/(\text{fpr}, \text{fnr}) \in errRates\})$
  $\partial\text{PRD}_\Lambda \leftarrow \partial\text{PRD}_\Lambda \cup \{(\alpha_\lambda, \frac{\alpha_\lambda}{\lambda})\}$
**end**
**return** $\partial\text{PRD}_\Lambda$

**Algorithm 1:** Classification-based estimation of the Precision-Recall curve.

**Binary Classification** We know from Theorem 4 that the Precision-Recall curve can be exactly inferred from the likelihood ratio classifier denoted as $\tilde{U}$. However, as explained earlier, since both the generated and target distributions ($Q$ and $P$ respectively) are unknown, one could not compute in practice this optimal classifier. Instead, we propose to *train* a binary classifier $U'$. Recall that from Theorem 5

the estimated PR curve, being computed with a sub-optimal classifier, lies therefore above the optimal one. We only assume in the following that the classifier –denoted to as $f$ in the algorithm description– which is returned by the function `learnClassifier` after the training, ranges in a continuous interval (e.g. $[0, 1]$), so that the binary classifier $U'$ is actually obtained by thresholding: $U'(Z) = \mathbb{1}_{f(Z) \geq t}$.

As a result, since the classifier needs some training data, the $N$ sample pairs $\mathcal{D} = \{(X_i, Y_i), 1 \leq i \leq N, X_i \sim P, Y_i \sim Q\}$ in the input dataset are first split into two sets $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ (function `createTrainTest`). For each image pair $(X_i, Y_i)$, a Bernoulli random variable $U_i$ with probability $\frac{1}{2}$ is drawn to decide whether a true sample $X_i$ (when $U_i = 1$) or a fake one $Y_i$ (when $U_i = 0$) is used for the training set $\mathcal{D}^{\text{train}}$. The other sample is then collected in the test set $\mathcal{D}^{\text{test}}$ to compute the PR curve.

**Precision and Recall estimation** Recall from Theorem 3 that the PR curve $\partial\text{PRD} = \{(\alpha_\lambda, \beta_\lambda), \lambda \in \overline{\mathbb{R}^+}\}$ is parametrized by the ratio $\lambda = \frac{\alpha}{\beta}$ between precision $\alpha$ and recall $\beta$. We denote by $\partial\text{PRD}_\Lambda$ the approximated PR curve when this parameter takes values in the set $\Lambda$.

Given a test dataset $\mathcal{D}^{\text{test}}$, the function `estimatePRD` computes the PR values $(\alpha_\lambda, \beta_\lambda)$ from the *false positive rate* fpr and the *false negative rate* fnr of the trained classifier $f$:

- fpr corresponds to the empirical type I error rate, that is here (arbitrarily) the proportion of real samples $z = X_i$ (for which $u = 1$) that are misclassified as generated samples (*i.e.* when $f(z) < t$);
- conversely, fnr is the empirical type II error rate, that is the proportion of generated samples $z = Y_i$ (for which $u = 0$) that are misclassified as real samples (*i.e.* $f(z) \geq t$);

Now, this raises the question of setting the threshold $t$ that defines the binary classifier $U'(z) = \mathbb{1}_{f(z) > t}$. Since Theorem 5 states that the computed precision and recall values $(\alpha_\lambda, \beta_\lambda)$ are actually upper-bound estimates, we use the minimum of these estimates when spanning the threshold value in the range of $f$. Note that it is sufficient to consider the finite set $fVals$ of classification scores over $\mathcal{D}^{test}$.

**Comparison with ROC curves** Using a ROC curve (for Receiver Operating Characteristic) to evaluate a binary classifier is very common in machine learning. Let us recall that it is the curve of the true positive rate ($1 - \text{fnr}$) against the false positive rate (fpr) obtained for different classification thresholds. Considering again the likelihood ratio test classifiers for all possible ratios would then provide the Pareto optimal ROC curve and could be used to assess if $P$ and $Q$ are similar or not. It turns out that the the frontier of the Mode Collapse Region proposed by [95] provides exactly this optimal ROC curve. For the recall, this notion is originally defined as follows:

$$MCR(P, Q) = \{(\epsilon, \delta) / 0 \leq \epsilon < \delta \leq 1, \\ \exists A \in \mathcal{A}, P(A) \geq \delta, Q(A) \leq \epsilon\}$$

From this definition, one can see that the MCR exhibits mode dropping by analyzing if part of the mass of $P$ is absent from $Q$. The notion differs from PRD at least in two ways. First MCR is not symmetric in $P$ and $Q$. Then it uses the mass of a subset $A$ instead of an auxiliary measure $\mu$ to characterize the shared / unshared mass between $P$ and $Q$. Despite those differences, the two notions serve a similar purpose. Given their respective interpretation as optimal type I vs type II errors, they mostly differ in terms of visual characterization of mode dropping.

## 2.6 Experiments

In this section we demonstrate that Algorithm 1 is consistent with the expected notion of precision and recall on controlled datasets such as CIFAR-10 and Imagenet. The results even compare favorably to [148]

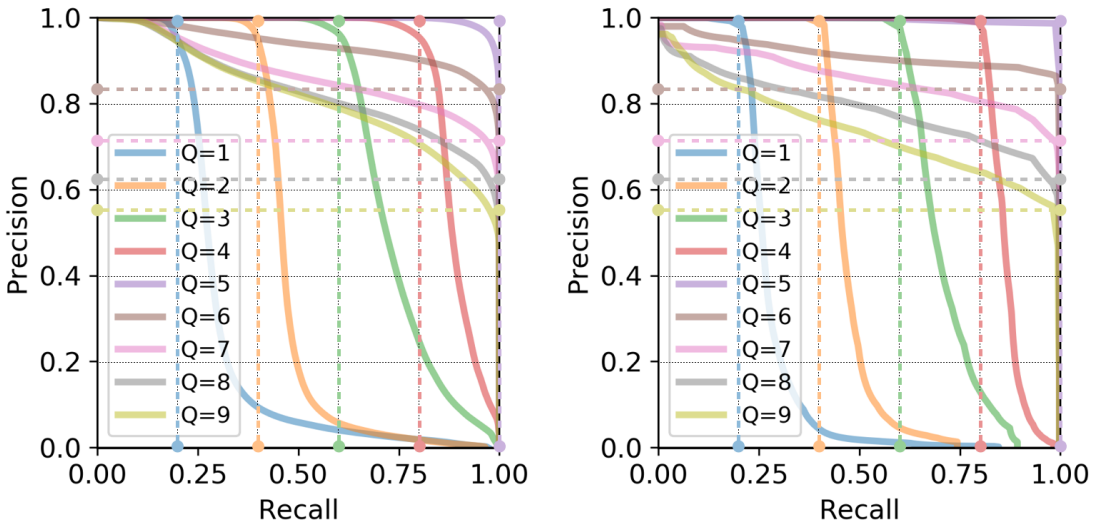



Figure 2.2: Precision-recall curves for $P$ made of the five first classes of CIFAR-10 versus $Q$ made of $q \in \{1, \dots, 9\}$ first classes. Left estimate from [148] and right our implementation.

for such datasets. The situation is more complex when one distribution is made of generated samples, because the expected gold-standard precision-recall curve cannot be predicted in a trivial way.

In all our experiments, we compute the precision-recall curve between the distribution of features of the Inception Network [164] (or some other network when specified) instead of using raw images (this choice will be discussed later on). In simple words, it means that we first extract inception features before training / evaluating the classifier. The classifier itself is an ensemble of 10 linear classifiers. The consensus between the linear classifiers is computed by evaluating the median of their predictions. Besides, each linear classifier is trained independently with the ADAM algorithm. We progressively decrease the learning rate starting from $10^{-3}$ for 50 epochs and use a fixed weight decay of 0.1. Any sophisticated classification method could be used to achieve our goal (deeper neural network, non-linear SVM, *etc*), but this simplistic ensemble network turned out to be sufficient in practice. Observe that this training procedure is replacing the pre-processing (K-means clustering) in the original approach of [148], which relies also on inception features so that both methods share a similar time complexity.

Figure 2.2 reproduces an experiment proposed by [148]. It presents the estimated precision-recall curves on distributions made from CIFAR-10 samples. The reference distribution $P$ is always the same and it gathers samples from the first 5 classes. On the other hand, $Q$ is composed of the first $q$ classes. When $q \leq 5$ we should expect a rectangular curve with a maximum precision of 1 and maximum recall of $q/5$ (as illustrated in middle of Fig. 2.1). Similarly, when $q > 5$ the expected curve is also rectangular one, but this time the maximum precision is $5/q$ and the maximum recall is 1 (Fig. 2.1, left). These expected theoretical curves are shown in dash. The original implementation from [148] is shown on the left and ours on the right. It is clear that both methods capture the intended behaviour of precision and recall. Besides, two subtle differences can be observed. First, as implied by Theorem 5 our implementation is always overestimating the theoretical curve (up to the variance due to finite samples). On the contrary, the clustering approach does not provide similar guarantee (as observed experimentally). Second, our implementation is slightly more accurate around the horizontal and vertical transitions.

One particular difficulty with the clustering approach lies in choosing the number of clusters. While the original choice of 20 is reasonable for simple distributions, it can fail to capture the complexity of strongly multi-modal distributions. To highlight this phenomenon, we present in Figure 2.3 another controlled experiment with Imagenet samples. In this case, $P$ and $Q$ are both composed of samples from 80 classes, with a fixed ratio $\rho$ of common classes. In this case, the expected curves can be predicted

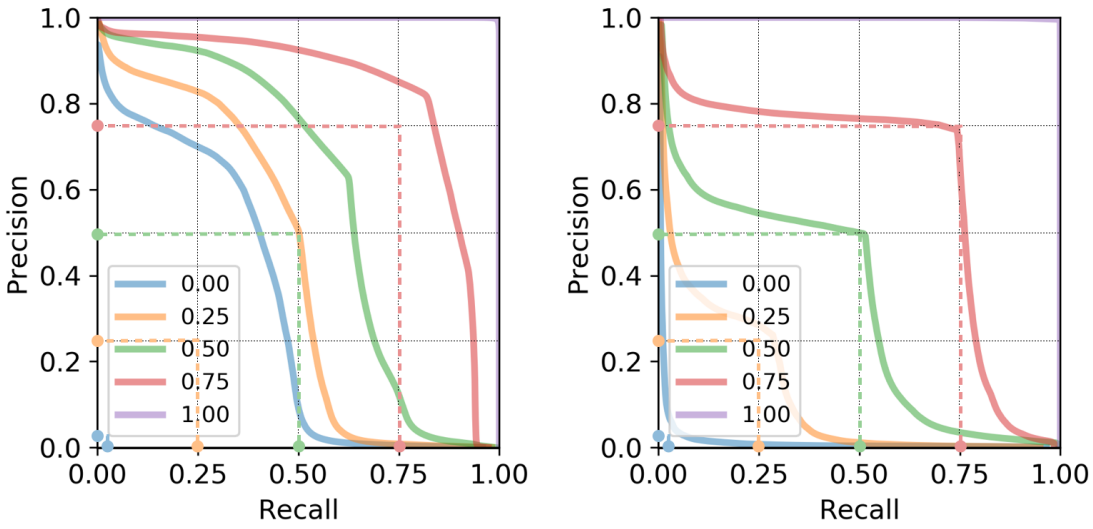



Figure 2.3: PR curves for $P$ and $Q$ made of 80 classes from ImageNet. The ratio of common classes varies from 0 to 100%. Left: from [148]. Right: our implementation.

(see dash curves). They correspond to rectangular curves with both maximal precision and recall equal to $\rho$. As can be seen on the experimental curves, the clustering approach is prone to mixing the two datasets in the same clusters. It therefore produces histograms that share a much heavier mass than the non discretized distributions, resulting in PR curves that depart strongly from the expected ones. Of course, such a drawback could be partially fixed by adapting the number of clusters. However even then the clustering approach may fail, as is demonstrated in Figure 2.4. In this experiment, the distribution $P$ is obtained approximately 60% female faces and 40% male faces from the CelebA dataset, while $Q$ is composed of female only. The theoretical curve is a sharp transition arising at recall 0.6. This is well captured by our estimate (right curve) while varying the number of clusters always leads to an oversmooth estimate, with either under estimated precision or over-estimated recall.

**Experiments on Generated Images** Figure 2.5 illustrates the proposed approach to three GANs trained on the Celeba-HQ [79] dataset. We highlight the two recent approaches of progressive GANs in [79] and the 0 centered gradient penalty ResNets found in [113] as they produce realistic images. For comparison, we also include DCGAN [140]. For analysis with Algorithm 1, we choose the first $N = 1000$ images of Celeba-HQ, and generate as many images with each GAN. For training the classifiers, we split each set (real and fake images) into 900 training images and 100 test images. In light of the previous experiments, we choose to train our architecture on top of vision relevant features. Because we are dealing with faces, we choose the convolutional part of the VGG-Face network [131]. One advantage on using VGG-Face is that artifacts present in generated images, such as unrealistic backgrounds, are mitigated by the VGG-face network, so that classification can focus on the realism of facial features. Of course, small artifacts can be present in even high quality generators and a perfect classifier could "cheat" by only using seeing such artifacts. Fig. 2.5, shows the computed PR curves for the three generators. Intuitively, networks with high precision should generate realistic images consistently. Progressive GANs achieve a maximum precision of 1.0, and overall high precision, which is visually consistent. DCGAN is producing unrealistic images which is reflected by it's overall low precision. In some sense, recall reflects the diversity of the generated images with respect to the dataset and it is interesting to note all networks achieved higher recall than precision. Finally, for the sake of comparison with the FID [59], the networks in Fig. 2.5 achieved FIDs of 25.23, 27.61 and 67.84 respectively from left to right (lower is better).

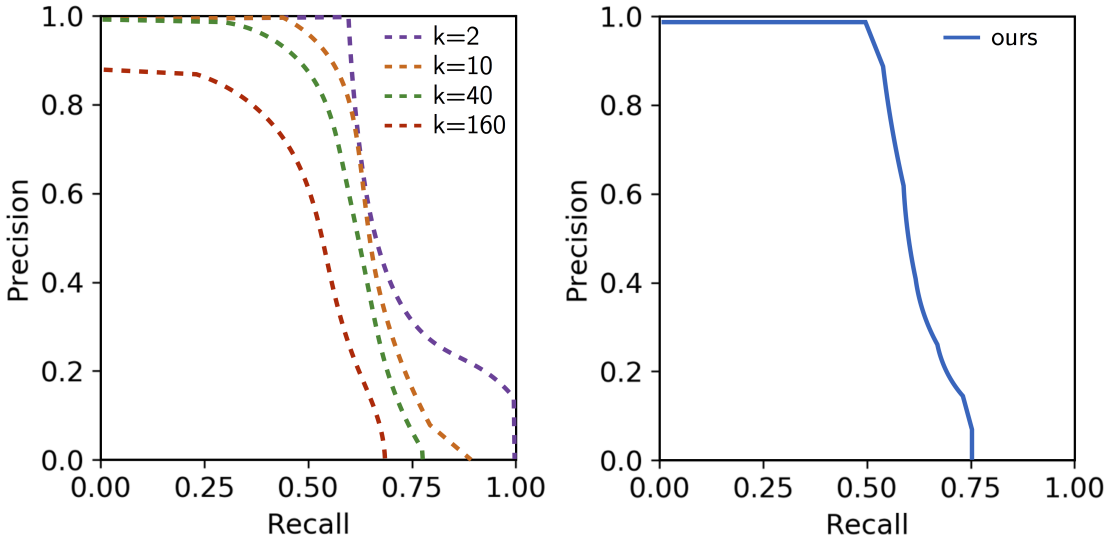



Figure 2.4: PR curves when $P$ is composed of faces from CelebA (60% females) and $Q$ is composed of females only.

Next, we analyze BigGAN [21] on ImageNet for our classification approach and the clustering approach presented in [148]. Both approaches use inception features as before. We take 80 images from the first 40 classes of ImageNet, and then 20 images from the first 40 classes for test images. We use 20 clusters for the K-means approach and a single linear layer for the classification approach. Fig. 2.6 highlights a large difference between the approaches; the clustering approach overestimates the similarities between the distributions and the classification approach easily separates the two distributions. As was demonstrated in Fig. 2.3, there are more classes than clusters, which could explain why images from both distributions may fall into the same clusters, in which [148] will fail to discern the two distributions. It is interesting to note that the classifier easily separates the distributions despite the inception features being sparse for image samples. One can observe a lack of intra-class diversity in the BigGAN samples, which may be how the classifier discerns the samples. We leave further investigation of this discrepancy for future work.

## 2.7 Discussion and future work

In this paper, we have revisited a recent definition of precision-recall curve for comparing two distributions. Besides extending precision and recall to arbitrary distributions, we have exhibited a dual perspective on such notions. In this new view, precision-recall curves are seen through the prism of binary classification. Our central result states that the Pareto optimal precision-recall pairs can be obtained as linear combinations of type I and type II errors of likelihood ratio classifiers. Last, we have provided a novel algorithm to evaluate the precision-recall curves from random samples drawn within the two involved distributions.

**Discussion** One achievement of our formulation is that one can directly define the precision-recall curves of distributions defined on continuous manifolds. In particular, our definition could be applied directly in the image domain, instead of first embedding the distribution in a feature space. From the strict computation perspective, there should not be any daunting obstacle in the way, as soon as we can have access to enough data to train a good classifier. This is usually the case for generative models, since the standard datasets are quite massive.

However, it is not obvious whether classifiers trained on raw-data provide useful notions of PR-

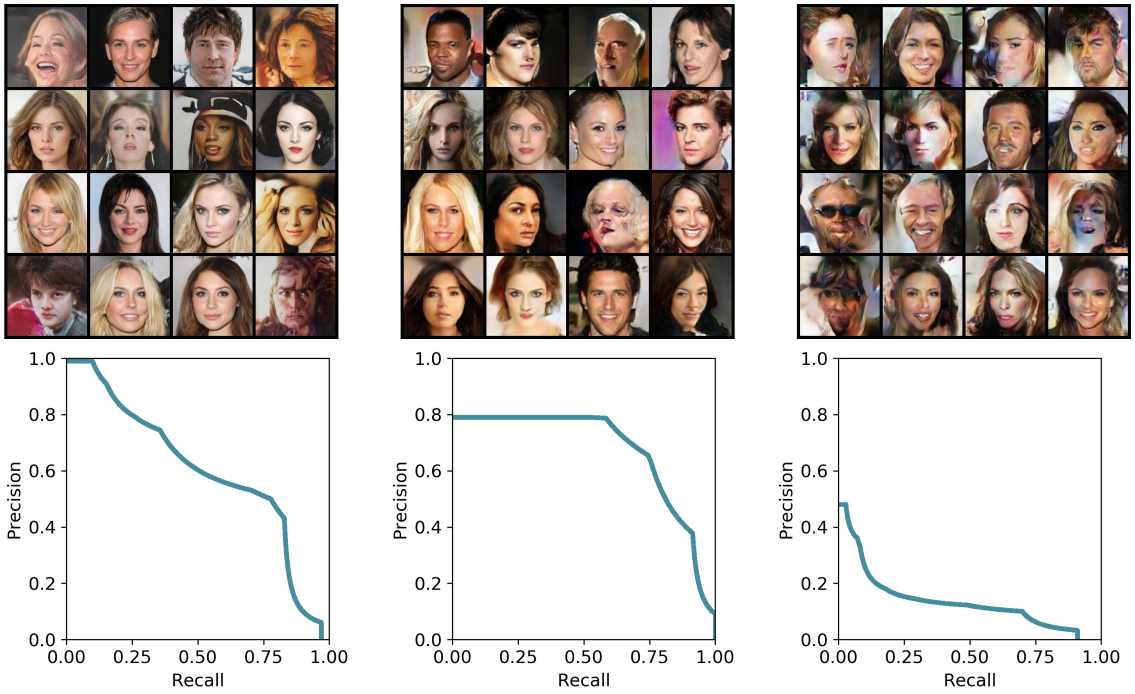



Figure 2.5: Precision-recall curves and generated images for various popular GANs on CelebA-HQ dataset [79]. From left to right: PGGAN [79], ResNet [113], and DCGAN [140].

curves. Indeed, given the current state of affairs of generative modeling, we think that the raw image curves may be less useful. Indeed, until now, even the best generative models produce artifacts (blurriness, structured noise, etc.). As such, the theoretical distributions (real and generated) are mutually singular. So, their theoretical precision-recall curve should be always trivial (*i.e.* reduced to the origin). It is hence a necessary evil to embed the distributions into a feature space as it allows a classifier to focus its attention on statistical disparities that are meaningful for the task at hand. For instance, when evaluating a face generator, it makes sense to use features that are representative of facial attributes. Nonetheless, future work should investigate a wider variety of pre-trained features as well as classifiers trained on raw data to determine which method is most suitable for computing PR curves.

**Perspectives** This work offers some interesting perspectives that we would like to investigate in the future. First, as opposed to the usual GAN training procedure where a scalar divergence is used to assess the similarity between generated and target distributions, one could use the proposed precision and recall definitions to control the quality of the generator while preventing mode-collapse. For instance, the discriminator could use the role of the classifier, as it has been done in [149].

Another interesting aspect is that like most existing divergences comparing probability distributions, the proposed approach is based on likelihood ratios that only compare samples having the same values. More flexible ways do exist to compare distributions, based for instance on optimal transport, such as the Wasserstein distance (e.g 1-Wasserstein GAN [8]) and could be adapted to keep the notion of trade-off between quality and diversity.

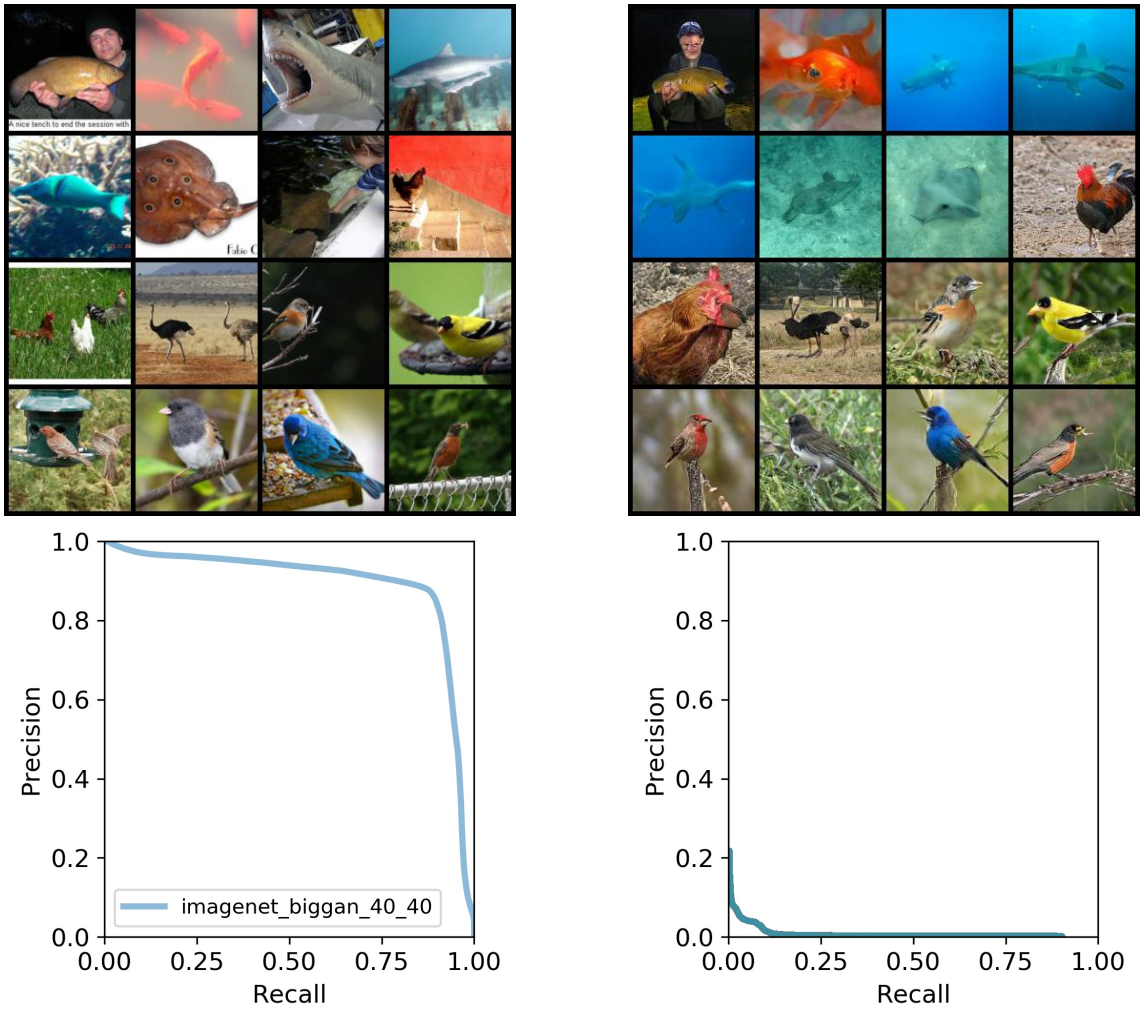



Figure 2.6: Evaluating generated samples on ImageNet. First row: Samples from various categories of ImageNet (on the left), and generated samples for the same categories from BigGAN [21] (on the right). Second Row: PR curves computed with the clustering approach of [148] and ours.

## 2.8 Proof of Lemma 1

Let $\varepsilon \geq 0$ such that $\mathbb{P}(U' = 1|U = 0) = \mathbb{P}(\tilde{U} = 1|U = 0) - \varepsilon$. First, we decompose $\beta'_\lambda$ into 4 terms

$$\begin{aligned}
\beta'_\lambda =& \mathbb{P}(U' = 0|U = 1) + \tfrac{1}{\lambda}\mathbb{P}(U' = 1|U = 0) \\
=& \mathbb{P}(U' = 0, \tilde{U} = 0|U = 1) + \mathbb{P}(U' = 0, \tilde{U} = 1|U = 1) \\
&+ \tfrac{1}{\lambda}\mathbb{P}(U' = 1|U = 0) \\
=& \mathbb{P}(\tilde{U} = 0|U = 1) - \mathbb{P}(\tilde{U} = 0, U' = 1|U = 1) \\
&+ \mathbb{P}(U' = 0, \tilde{U} = 1|U = 1) + \tfrac{1}{\lambda}\mathbb{P}(U' = 1|U = 0).
\end{aligned}$$

Considering separately each of the previous terms, we have

$$\begin{aligned}
A =& \mathbb{P}(\tilde{U}=0|U=1)\,,\\
-B =& \mathbb{P}(U'=1,\tilde{U}=0|U=1) = \int \mathbb{1}_{U'=1}\mathbb{1}_{\tilde{U}=0}dP\\
\leq& \int \mathbb{1}_{U'=1}\mathbb{1}_{\tilde{U}=0}\tfrac{1}{\lambda}dQ = \tfrac{1}{\lambda}\mathbb{P}(U'=1,\tilde{U}=0|U-0)\\
=& \tfrac{1}{\lambda}(\mathbb{P}(U'=1|U=0) - \mathbb{P}(U'=1,\tilde{U}=1|U=0))\\
=& \tfrac{1}{\lambda}\Big(\mathbb{P}(\tilde{U}=1|U=0) - \varepsilon\\
& \quad - (\mathbb{P}(\tilde{U}=1|U=0) - \mathbb{P}(U'=0,\tilde{U}=1|U=0))\Big)\\
=& \tfrac{1}{\lambda}(\mathbb{P}(U'=0,\tilde{U}=1|U=0) - \varepsilon).
\end{aligned}$$

Finally

$$B \geq -\tfrac{1}{\lambda}(\mathbb{P}(U'=0,\tilde{U}=1|U=0) - \varepsilon).$$

Similarly

$$\begin{aligned}
C =& \mathbb{P}(U'=0,\tilde{U}=1|U=1) = \int \mathbb{1}_{U'=0}\mathbb{1}_{\tilde{U}=1}dP\\
\geq& \int \mathbb{1}_{U'=0}\mathbb{1}_{\tilde{U}=1}\tfrac{1}{\lambda}dQ = \tfrac{1}{\lambda}\mathbb{P}(U'=0,\tilde{U}=1|U=0)\,.
\end{aligned}$$

Using both inequalities for $B$ and $C$, one gets

$$B + C \geq \frac{\varepsilon}{\lambda}.$$

Last,

$$D = \tfrac{1}{\lambda}\mathbb{P}(U'=1|U=0) = \tfrac{1}{\lambda}(\mathbb{P}(\tilde{U}=1|U=0) - \varepsilon).$$

Putting everything together, namely $\beta'_\lambda = A + B + C + D$, yields

$$\beta'_\lambda \geq \mathbb{P}(\tilde{U}=0|U=1) + \tfrac{1}{\lambda}\mathbb{P}(\tilde{U}=1|U=0) = \beta_\lambda.$$

Using (by a slight abuse of notation) $\alpha_\lambda = \lambda\beta_\lambda$ and $\alpha'_\lambda = \lambda\beta_\lambda \geq \alpha_\lambda$ concludes the proof. □

Chapter 3

# Overfitting in Generative Networks

In the previous chapter, we presented the evaluation of several generative models in terms of the closeness of real and generated distributions via their precision recall curves. However, the analysis lacked an important aspect, whether or not these models *memorize* images they saw during training. Indeed, the presented metrics will report good scores even for a model that outputs its training images. In this chapter, we develop a tool to assess whether a model is biased to generate training samples, as compared with test samples coming from the same distribution. When this work was published, it was common to include visualizations of training set nearest neighbors, to suggest generated images are not simply memorized. We take a different approach through directly reconstructing images by solving an optimization problem and find generator inputs. Ultimately, we find that certain generators can overfit samples when too few points are used. In the extreme case, the distribution of errors of test and train samples are disjoint; thus an observer can perfectly observe which training samples were used. Using this methodology, we show that pure GAN models appear to generalize well, in contrast with those using hybrid adversarial losses, which are amongst the most widely applied generative methods. We also show that standard GAN evaluation metrics fail to capture memorization for some deep generators.

In the next chapter, we take a privacy perspective of our findings by constructing a membership attack. During the time this work was conducted, it was not feasible to train GANs on few data points. Indeed, this chapter only involves GANs trained with sufficiently many data, where we don't detect overfitting. In the next chapter, we use new tools to train GANs on a few data points. In this setting, it is possible to perform membership inference versus GANs. Thus, we present a remedy for this issue in the Chapter 5, by sharing weights between generators and limiting the amount of trainable parameters. Yet another possible perspective is presented in Chapter 6. Here we show two possible strategies for robustness versus membership attacks: one being removing dataset bias and two early stopping during GAN training.

**Continued Work** Since publication, many methods have expanded upon this work. At the time of publication, we found relatively simple methods were successful to recover generator inputs, however, new generator architectures are more complex and harder to invert. For instance, Image2Stylegan [3, 2] adds terms to the optimization objective to handle the multiple inputs of the StyleGAN network (see Chapter 5 for architecture details), in contrast to optimizing only the output as is done in this chapter. In [170, 35], inversion is performed layer-by-layer; doing so this way simplifies each optimization and provides a warm start for the next layer. Finally, in Encoder4Editing [170], rather than just optimizing for the input, they optimize a separate network to encode into input space. We explored this avenue in parallel around this time, and we present a similar encoder based inversion in the context of membership attacks in Chapter 4.

Finally, there are the works which try to diagnose overfitting in GAN generated images. For instance, in [108], the authors design a non-parametric three sample test to investigate what they call "data copying". It works by trying to classify generated and real samples, similar to the two sample test we provided

in Chapter 2. Finally, there are several methods which study GAN data copying in the low data setting [45]. As we were unable to stabilize GAN training for few data points in this work, the authors in [45] perform a more complete study of GAN overfitting in this case.

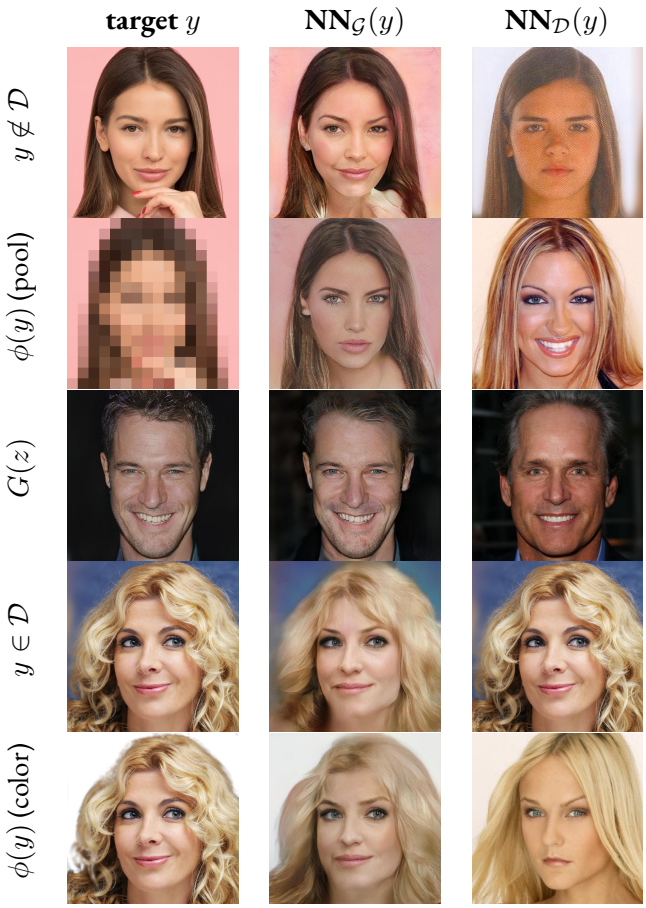



Figure 3.1: Rather than inspecting the most similar images $\text{NN}_{\mathcal{D}}(y)$ in the training dataset $\mathcal{D}$ for sampled generated images $G(z)$ (row 3), we consider finding the most similar image in the manifold $\text{NN}_{\mathcal{G}}(y)$ of generated images (column 2). As seen in the last two rows, $\text{NN}_{\mathcal{G}}(y)$ is more meaningful under some transformations. Analysis of the discrepancy between reconstructions of the train set $\mathcal{D}$ and reconstructions outside $\mathcal{D}$ makes it possible to detect overfitting for some generators.

## 3.1 Introduction and Related Work

In just a few short years, image generation with deep networks has gone from niche to a center piece of machine learning. This was largely initiated by Generative Adversarial Networks (GANs) [52] and since then incredible progress has been made, from deep convolutional GAN (DCGAN) [139] producing artifacted faces, to progressive GANS (PGGAN) [78] producing faces which are virtually indistinguishable from real ones even to human observers and at high resolution (see Fig. 3.1). While a large amount of research has proposed new generative models, less research has been devoted to the evaluation of such models. Furthermore, evaluating overfitting of deep generators has been performed via intuitive visual demonstrations, such as training set nearest neighbor search and latent space interpolation [21, 78]. Fig. 3.1 (last column) illustrates the nearest neighbor (NN) test, where $\text{NN}_{\mathcal{D}}(y)$ is the training dataset NN of a few images $y$. While $\text{NN}_{\mathcal{D}}(y)$ with the Euclidean distance is a common heuristic (last column in Fig. 3.1), here we show it is not robust to an image transformation.

In contrast, we suggest to rely on the opposite methodology by optimizing the latent code $z \sim \mathcal{Z}$ to find the nearest neighbors $\text{NN}_{\mathcal{G}}(y)$ in the manifold of generated faces $\mathcal{G} = \{G(z)\}_{z \sim Z}$ of images from the training ($y \in \mathcal{D}$) and a validation set ($y \notin \mathcal{D}$). Not only is this approach more robust, it provides us with reconstructions errors which can be analyzed for different sets of images. Using this framework that we refer to as *latent recovery*, we propose the following contributions:

- A demonstration of successful latent recovery across a variety of generators. In Section 3.2 we introduce our optimization procedure and show it is meaningful even if the target image is corrupted.

- Section 3.3 introduces a novel method to numerically estimate overfitting in deep generators via statistics of recovery errors on test and train sets. Overfitting is undetectable for GANs, which is corroborated visually in Fig. 3.3 and statistically in Table 3.1. Overfitting is however detectable in hybrid adversarial losses similar to CycleGAN [197], and easily detectable in non-adversarial generators such as GLO [17]. Finally, we show that standard evaluation metrics do not detect overfitting in some models.

### 3.1.1 Related Work

Adversarial losses have seen successful applications in a variety of settings beyond just image generation: unpaired image to image translation in CycleGAN [197], face attribute modification in StarGAN [29] and various image inpainting techniques [65, 184] to name a few. This progress has created a huge need to evaluate generated image quality, which to some degree has not been fully answered [20].

**GAN Evaluation Metrics** The Fréchet Inception Distance (FID), introduced in the first chapter . 2.1, is the most commonly used metric for GAN evaluation. In the large scale GAN study [103], FID was used to compare a huge variety of GANs, wherein it was shown auxiliary factors such as hyperparameter tuning can obfuscate true differences between GANs. In [147], notions of precision and recall are introduced for generated images, to help characterize model failure rather than providing a scalar in image quality. While these works have helped to compare GAN image quality, they do not address overfitting of the training set.

**Overfitting in Generative Networks** For image classification, a model is said to overfit when it performs significantly better on training examples compared to test examples, and said to generalize otherwise. While the exact reasons why deep nets generalize even when over parametrized is an open question, they are certainly not immune to overfitting. In the extreme case, Zhang et al. [188] demonstrated random labels can be perfectly memorized even on the large scale ImageNet database.

Despite this, very little work has gone into defining overfitting for generative models. In [10], the authors defined generalization for GANs in a largely theoretical setting. The formulation was used to suggest a new GAN training protocol rather than provide an evaluation technique. In [11], the support of a GAN generator, in terms of the number of face identities it could produce, was estimated using the birthday paradox heuristic. While crude, it suggested the support of faces could be quite large with respect to the size of the training set. The very recent work of [54] attempts to numerically estimate the notion of overfitting with a Neural Net Distance (NND). That is, they train a neural net to differentiate generated samples from real samples and similar to [68], use the resulting divergence as a measure of quality. Importantly, they are able to show a slight overfitting for some GANs and show that this divergence penalizes a generator trivially memorizing the train set. Unfortunately, this approach requires a massive test set in order to train the NND, which is unrealistic considering successful GANs already require massive train sets, not to mention the NND itself needs to be trained. Furthermore, the NND may favor GANs if the divergence they use resembles or is identical to the GAN under evaluation.

Finally, a new class of generative models has recently been proposed which involves invertible generators [38, 86]. These generators are attractive as they are mathematically well motivated and admit exact log-likelihood estimations, via taking the determinant of each layer jacobian. [120] examined the log likelihoods for such models and showed that while some GAN models generalized nicely to validation samples, out of distribution samples, such as those taken from completely different datasets, yield higher likelihoods. [168] also examined log-likelihood in the generative setting, and both works ultimately cautioned against the use of log-likelihood for generative evaluation.

**Memorization and Privacy** Beyond these aspects of memorization and practical evaluation of generators lies the important and debated issue of privacy: How to ensure that the data used for training cannot leak by some reverse engineering, such as reconstruction from features [104, 47] ? Because GANs have seen such widespread application, it is imperative that we have better evaluation tools to assess how much these networks have overfit the training data. For example, if a user is using a neural net to inpaint faces as in [94] or to perform super-resolution enhancing [34], it seems necessary to ensure verbatim copies of training images do not appear, due to privacy or even copyright concerns. Indeed, several attacks against machine learning systems have been exposed in the literature [129]. For instance, authors in [47] designed an inversion attack to coarsely recover faces used during the training of a white box facial recognition neural network. More recently, [155] performed a successful membership attack, which is the ability to discern training examples from a model, in a purely black box setting. Very recently [57] explored the potential of membership attacks for GANs and exploited the tendency of the discriminator to overfit the training set.

## 3.2 Reconstruction by Latent Code Recovery

This section proposes a methodology for reconstructing the most similar images to target images with an existing generator. Inversion of deep representations has been already addressed in the literature. [104] used a simple optimization procedure to maximize an output class of a VGG-like network. In the seminal works of [124, 165], a similar inversion of deep nets unveiled adversarial examples. In [103], *generative* networks are inverted to study recall, which is the ability of the network to reproduce all images in the dataset and finally [115] used latent recovery of a GAN generator to evaluate its quality.

Other works tackle recovering latent codes directly by training an encoder network to send images back from image space to latent space, such as the BEGAN model [15] or Adversarially Learned Inference (AGI) [40]. In Generative Latent Optimization (GLO) [17], a generative model is trained along with a fixed-size set of latent codes, so that they are known explicitly when training finishes.

In this chapter, we will proceed by recovering latent codes via optimization, following [115, 19, 96, 103]. In contrast with [115], we will ultimately be concerned comparing image recovery between train and validation sets.

### 3.2.1 Latent Code Recovery with Euclidean Loss

We explore recovery with a euclidean loss and find it is effective at recovering latent codes for a variety of GAN methods. Here, we consider the following *latent recovery* optimization problem

$$z^\star(y) \in \arg\min_z \|\phi(G(z)) - \phi(y)\|_2^2 \qquad (\mathrm{NN}_\mathcal{G})$$

where $G$ is a deep generative network, $z$ is the input latent vector and $y$ is the target image. Using a solution $z^*$ of Problem ($\mathrm{NN}_\mathcal{G}$), we denote by $\mathrm{NN}_\mathcal{G}(y) = G(z^*)$ the Nearest Neighbor recovery of a given image $y$ in the set of generated images, as opposed to the usual NN search in a dataset $\mathcal{D}$: $\mathrm{NN}_\mathcal{D}(y) = \arg\min_{x\in\mathcal{D}} \|x - y\|$. In this work, we consider mostly $\phi$ as the identity, but other operators are discussed in the next paragraph and for applications such as super resolution in Section 5.5. Fig. 3.1 illustrates the difference between the two NN searches on a few examples.

**Experimental Validation** In every experiment, we employ LBFGS and noted it converges roughly 10x faster than SGD (successful recovery requiring approximately 50 iterations as opposed to 500 in [19, 96]). Although Eq. ($\text{NN}_\mathcal{G}$) is highly non-convex, the proposed latent recovery optimization works well, as shown in Fig. 3.1 and Fig. 3.3. In particular for generated images $y = G(z)$, where $\text{NN}_\mathcal{G}(y) = z$, a global minimum (verbatim copy) is consistently achieved (see third row of Fig. 3.1). Every network analyzed in this document appeared to be able to verbatim recover generated images, an observation also noted by [96] and exemplified by the tight distribution of errors near zero in Fig. 3.4. Note that we also considered the widely used perceptual loss [73] by taking $\phi$ to be VGG-19 features, with either no improvement or even degradation of visual results (see supplementary). Furthermore, we did not see any difference in the statistical results of Section 3.3 for perceptual losses.

### 3.2.2 Latent Code Recovery Under Distortion

It should be noted that Eq. ($\text{NN}_\mathcal{G}$) by itself may not be meaningful for some generators. For example, if the generator is invertible, errors will zero regardless of the target image. To verify that Eq. ($\text{NN}_\mathcal{G}$) is meaningful, we want to make sure the error is lower for images inside the considered manifold and large for those outside. To do this, we follow [59] (used there to motivate the FID) wherein we test Eq. ($\text{NN}_\mathcal{G}$) response to various distortions. We choose $\phi$ to be one of the three distortions that are illustrated in Fig. 3.2:

- **Smooth Vector Field Warp (Fig. 3.2a)** Following [59, 191] we warp training images by bilinear interpolation with a smooth 2D vector field $V_{\sigma_d} = V * g$, which is obtained from the Gaussian smoothing $g$ of a Gaussian random vector field $V(x, y) \sim \mathcal{N}(0,\ \sigma_d^2)$;
- **Corruption Noise Patches (Fig. 3.2b)** As [94], we corrupt training images by replacing patches of various sizes with fixed Gaussian noise with variance $\sigma_d^2$;
- **Additive Noise (Fig. 3.2c)** We add noise to each training image with $X_n = X + W_d$, where $W_d$ is sampled from a Gaussian distribution $W_d \sim \mathcal{N}(0,\ \sigma_d^2)$.

**Experimental Validation** Fig. 3.2 demonstrates a few facts about latent recovery. By inspection of recovered images, it appears robust enough to recover faces semantically similar to the ground truth even if the image has been heavily distorted. It also demonstrates the precision of the network, for example the three networks highlighted will reject images only slightly outside the manifold. In Table 3.1, we can see that not all networks share the same specificity. For example, the GLO networks can recover distorted images with similar MRE's to training images, which means the networks are less precise. This is coupled with a lower FID of the network, for example see Fig. 3.5.

## 3.3 Using Latent Recovery to Assess Overfitting

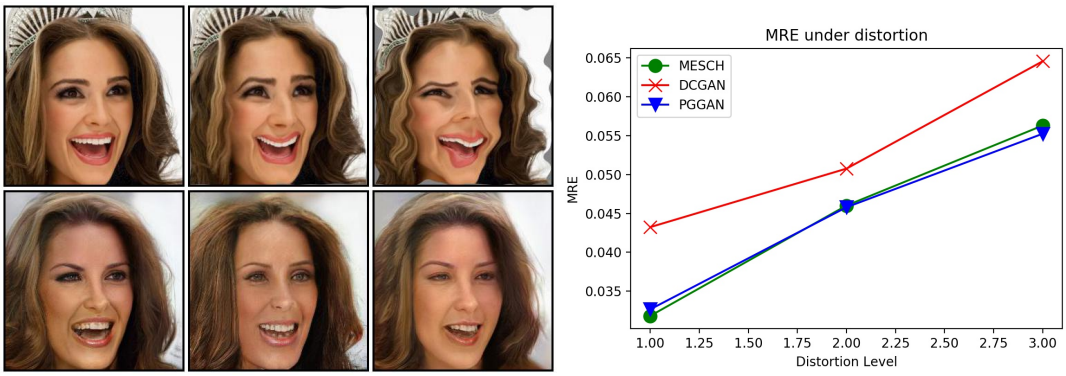
In this section, we train a variety of generative models with a training and validation split. Then, we analyze the difference between image recovery using Eq. ($\text{NN}_\mathcal{G}$), between training and validation images.

### 3.3.1 Training Protocols

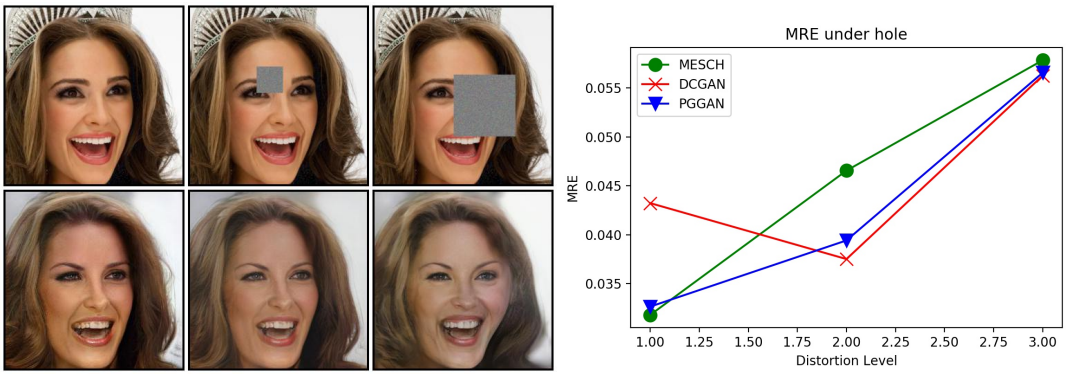
We summarize the details of each generative model below, in terms of their training procedure and purpose within this work. **GAN** Generative Adversarial Networks (GAN) involve a stochastic training procedure which simultaneously trains a discriminator and a generator. The original GAN [52] optimization problem writes

$$\max_D \min_G \mathbb{E}_{z\sim\mathcal{N}(0,1),x\sim p_{data}}[\mathcal{L}_{adv}(D, G, z, x)] \tag{3.1}$$

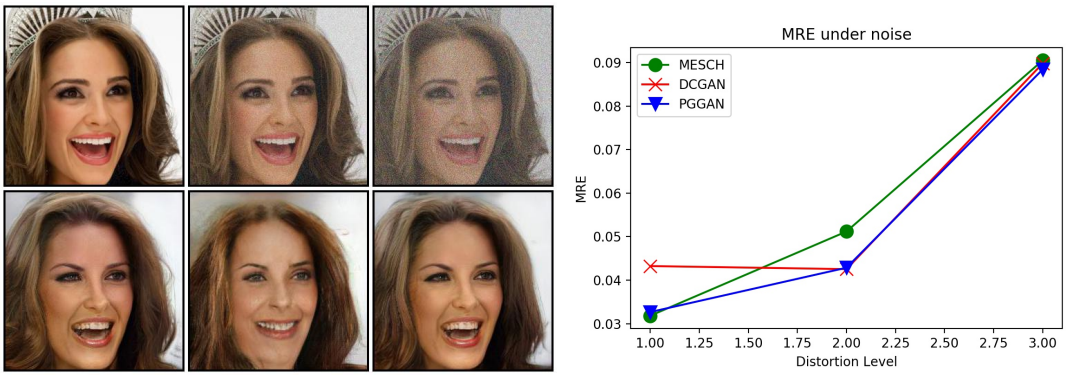
where $\mathcal{L}_{adv}(D, G, z, x) = \log(D(x)) + \log(1 - D(G(z)))$. We examine three prominent GANs in the literature. First is DCGAN [139], as it is one of the most widely used GAN architectures and with




(a) This is a deformation by smooth diffeomorphism (warping) . Top row is real and bottom row are recovered.




(b) An ansupervised inpainting (face completion) task.




(c) An additive white noise .

Figure 3.2: Median recovery error (MRE, see Eq. (3.4)) for 1800 test images on various GAN generators (PGGAN [78], MESCH [112] and DCGAN [139]) under various distortions $\phi$ in latent recovery optimization ($\text{NN}_G$) (see text for details).

still decent performance across a variety of datasets [103]. Then we study two state-of-the-art GANs for high resolution generation; progressive growing of GANs [78], which we refer to as PGGAN and the zero centered gradient penalty Resnet presented in [112], which we refer to as MESCH. We train these three GANs on CelebA-HQ with a training split of the first 26k images and the first 70k images of LSUN bedroom and tower. We chose these splits to preserve the quality of each method, as GAN quality significantly degrades with small dataset sizes.

**Generative Latent Optimization (GLO)** The recently introduced Generative Latent Optimization

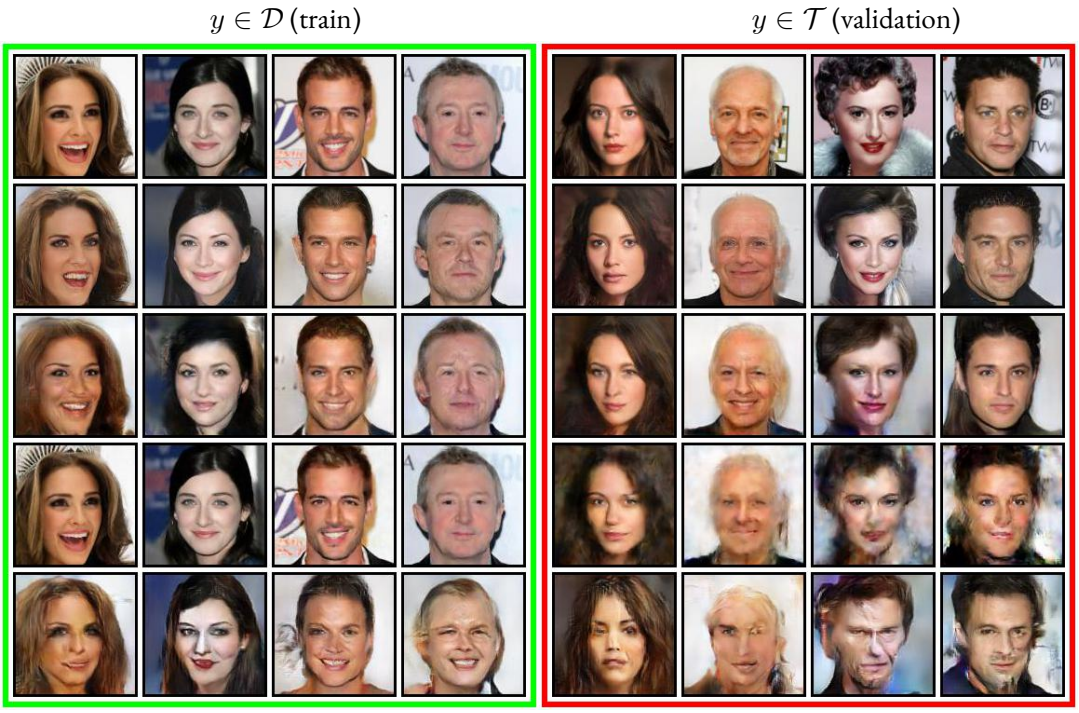



Figure 3.3: Latent recovery of training images from $\mathcal{D}$ (left, green frame) and test images from $\mathcal{T}$ (right, red frame) for $128 \times 128$ images of Celeba-HQ [78]. From top row to bottom are first **target** images, and then recovery from Progressive GANs [78] (**PGGAN**), o-GP resnet GAN [112] (**MESCH**), a **GLO** network [17], and finally a Cycle-GAN like network [197] (**AEGAN**). While GLO obviously shows some memorization of training examples, it is hard to visually assess when overfitting happens for other methods, as discussed in Section 3.3 (with additional details on architectures and training).

(GLO) creates a mapping from a fixed set of latent vectors to training images. The GLO objective is as follows

$$\min_G \sum_{(z_i,x_i)} \mathcal{L}_{rec}(G(z_i), x_i) := \|G(z_i) - x_i\|_2^2 \tag{3.2}$$

where $x_i \in \mathcal{D}$ refers to training images, $z_i \sim \mathcal{N}(0,1)$ samples a Gaussian distribution and the pairs $(z_i, x_i)$ are drawn once and for all before training begins. Because we know the latent distribution is Gaussian, we can easily sample the network after it is trained.

**AutoEncoder** Finally, we train a vanilla autoencoder on CelebA-HQ with the objective:

$$\min_{G,E} \sum_{x_i \in \mathcal{D}} \mathcal{L}_{rec}(G, E, x_i) \tag{3.3}$$

**Hybrid Losses** We consider a generative model combining both the adversarial loss Eq. (3.1) with euclidean auto encoding loss (3.3) which we refer to as AEGAN.

Concerning models trained with a reconstruction loss (GLO, AEGAN and AE), we selected these architectures for a theoretical perspective, as they offer interesting windows into how generators can memorize. In particular, we will study the impact of the training set size $N$ on the overfitting inclination. For example, while we were unable to train a good quality GAN with a small set of images (say 256), GLO converges extremely quickly in such a case. See the GLO-256 network in Fig. 3.3 ($4^{\text{th}}$ row) where memorization is immediately apparent. As a result, we will refer respectively to GLO-$N$, AEGAN-$N$ and AE-$N$, to account for this size. Besides, for both the AE and AEGAN models, we forgo optimization

in Eq. ($\mathrm{NN}_{\mathcal{G}}$) and use the encoder $E$ (3.3) to recover the latent vector, as is natural for autoencoder models.

In the next paragraphs, we will proceed to show that it is possible for generative networks to memorize in the sense that validation and training sets have significantly different recovery error distributions.

### 3.3.2 Comparison of recovery errors

Figure 3.4 shows the histograms of recovery errors on train ($\mathcal{D}$ in green) and validation ($\mathcal{T}$ in red) datasets from CelebA-HQ, for various generators. For the sake of readability, the distribution of errors for generated images (yellow) from $\mathcal{G}$ and distorted images (blue) are only displayed for PGGAN and MESCH. Confirming visual inspection from Fig. 3.1, observe that the recovery errors for generated images (in yellow) are quite low. Increasing the number of iterations and using several random initializations improve results, but have not been used to reduce computation costs.

Now we are going to consider the distribution of recovery errors for test and train. For GLO-$N$ and AEGAN-$N$ generators with $N \in \{128, 1024, 8192\}$, the difference is clear, and is decreasing with the number of training images $N$. For very small datasets of $N = 128$, the train and validation error distributions are disjoint. On the other hand, pure GAN models can not be successfully trained with small datasets. We therefore only trained PGGAN and MESH with full datasets and in both cases, the difference of recovery error distribution between the train (green) and validation (red) set is barely noticeable. Further statistical analysis in the next paragraph shows indeed that such a small gap is very likely for two samples drawn from the same law, demonstrating generalization.

### 3.3.3 Statistical Analysis

In light of the previous results, we propose two simple definitions to measure and detect overfitting without relying on histograms or image inspection. First, to summarize the distribution of errors to a single value, we consider the **Median Recovery Error** (MRE), defined for a generator $G$ and a dataset $Y$ as

$$\mathrm{MRE}_G(\mathcal{Y}) = \text{median} \left\{ \min_z \|y_i - G(z)\|^2 \right\}_{y_i \in \mathcal{Y}} \tag{3.4}$$

Table 3.1 reports such values for other deep generators and other datasets.

Then to measure the distance between two distributions, that is to estimate to which extent the generator overfits the training set, we simply compute the normalized **MRE-gap** between validation $\mathcal{T}$ and train $\mathcal{D}$ dataset, which writes

$$\text{MRE-gap}_G = \left(\mathrm{MRE}_G(\mathcal{T}) - \mathrm{MRE}_G(\mathcal{D})\right) / \mathrm{MRE}_G(\mathcal{T}) \tag{3.5}$$

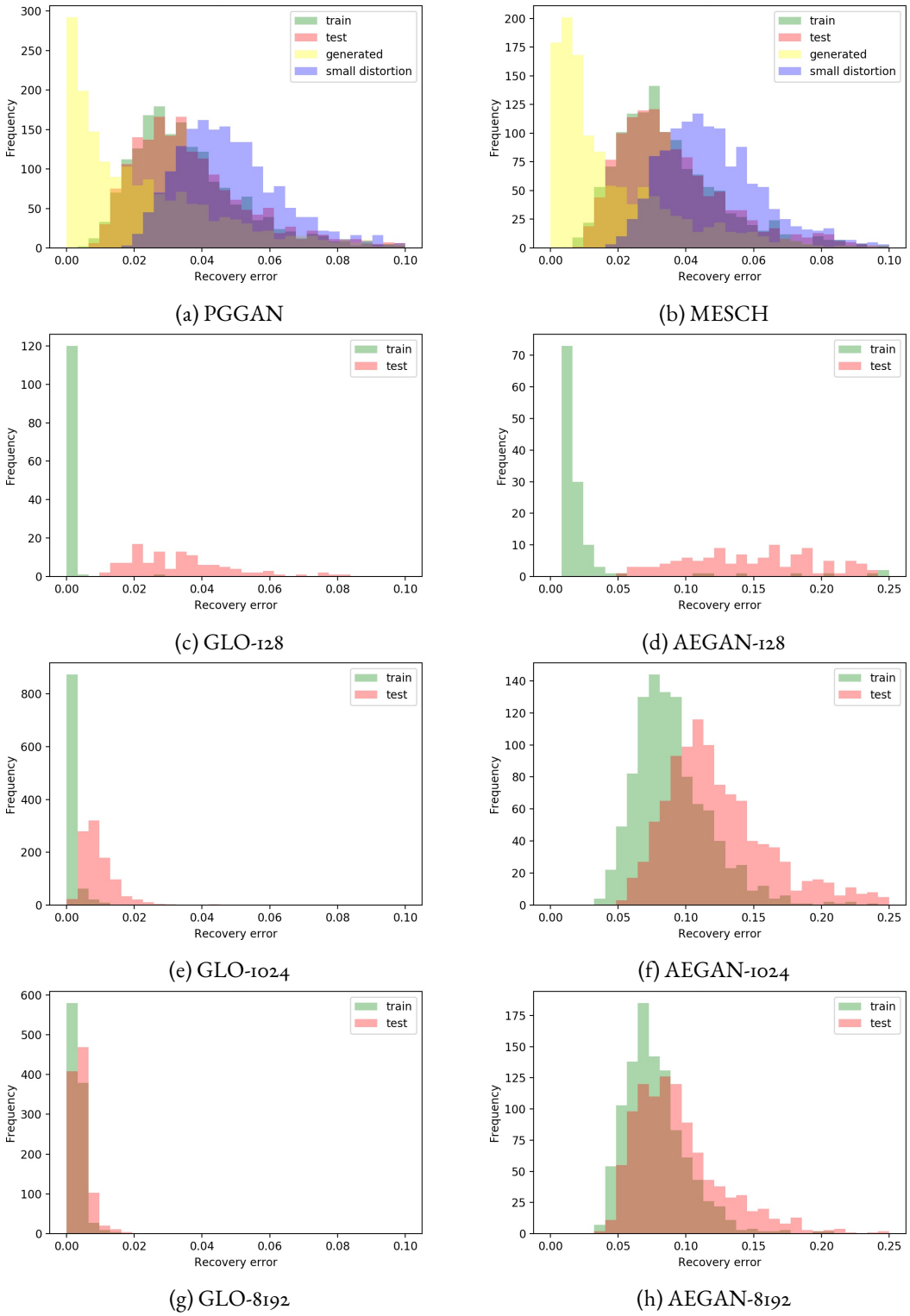
These values are reported[1] in Table 3.1.

Instead of using an empirical threshold to automatically assess if the amount of overfitting is significant regarding the size of the training set, we rely on a statistical test. We compute the $p$-value of the Kolmogorov-Smirnov test (KS) which measures the probability that two random samples drawn from the same distribution have a larger discrepancy, defined as the maximum absolute difference between cumulative empirical distributions, than the one observed.

Such $p$-values are displayed in Table 3.1, and a threshold of $1\%$ is used to detect overfitting (values are highlighted). To show the consistency between the two proposed metric, we also highlight the values of MRE-gap that are above $10\%$.

Observe that the results are mostly confirming previous empirical evidence: memorization is strongly correlated to the number of images seen during training. We also see that the same overfitting occurs on different datasets (CelebA-HQ and LSUN), and for autoencoder (AE). At $N = 26000$ on CelebA-HQ and $N = 32768$ on LSUN bedroom, overfitting is no longer detectable.

[1] Notice that other metrics could have been used, such as the Wasserstein distance.




(a) PGGAN (b) MESCH

(c) GLO-128 (d) AEGAN-128

(e) GLO-1024 (f) AEGAN-1024

(g) GLO-8192 (h) AEGAN-8192

Figure 3.4: Histograms of recovery errors on train $\mathcal{D}$ and validation $\mathcal{T}$ datasets from CelebA-HQ showing that overfitting is not happening for PGGAN and MESCH generators on the training dataset, but is for GLO-$N$ and AEGAN-$N$ when training for a small dataset $N \leq 8192$.

However, using the proposed statistics (p-value, normalized MRE-gap) is much more practical to detect overfitting than only inspecting histograms and easier to threshold than MRE itself. It also illustrates that such statistical principle overrules empirical evaluation, as memorization is indeed sometimes

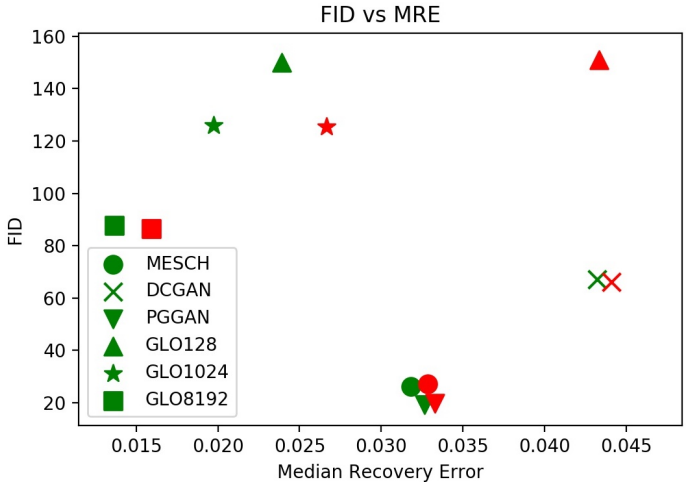



Figure 3.5: Comparison of FID versus Median Recovery Error (MRE) for various models computed over training images (in green) and validation images (in red). FID does not detect memorization in GLO models.

quite hard to tell from simple visual inspection, such as for the AEGAN generator in Fig. 3.3.

### 3.3.4 FID Does Not Detect Memorization

The FID is the standard GAN evaluation metric for images [59], so it is natural to ask whether this metric can be used to detect memorization. Figure 3.5 displays FID scores computed between generated and training images (in green) and generated and test images (in red). While the median recovery error (MRE) is able to detect memorization in GLO models, the FID is not sensitive to this (this fact was also noted by [54]). We do not suggest replacing the FID, but rather using MRE to provide a more complete picture of generator performance. Besides, other metrics such as the precision recall introduced in [147] can be considered as well to tackle more subtle statistical biases such as mode dropping versus mode invention.

## 3.4 Discussion and Future Work

### 3.4.1 Notes on Applications

Recently, GANs have seen wide application to various face generation tasks, such as face attribute modification [29], generative face completion [94] and face super-resolution [34]. In a similar vein, deep image prior [171], recovers images by first fixing a random latent vector, then optimizing over the parameters of a randomly initialized generator. We apply ($\text{NN}_G$) to face inpainting and super resolution for two reasons; first it shows off-the-shelf GAN generators are well suited for a variety of downstream tasks, which is also noted in [184] and second it provides additional visual insight into the observations of the previous section.

Figure 3.6 shows the progressive GAN generator [78] applied to face inpainting ($\phi$ is a mask) and super-resolution ($\phi$ is a 64x pooling). While the face inpainting is artifacted, we note that the results are decent without any post processing and similar to those presented in [94] (while being non-feedforward). As for super-resolution, we obtain results at least on par with [34]. An intriguing property of the images is that the recovery is semantically accurate, in terms of attributes such as gender, facial features and pose, whilst recovering a face that appears to be a different identity. This happens despite the use of images that the PGGAN generator [78] was trained on, which is in accordance with the observations of Sec. 3.3. Put in another way, we believe the fact that PGGAN has generalized well to CelebA-HQ, also means that it

| | | KS p-value | MRE-gap | MRE | | | |
|---|---|---|---|---|---|---|---|
| | | train vs val | | train | val | generated | small distort |
| CelebA-HQ | **dcgan** | 9.43e-01 | 1.79e-02 | 4.95e-02 | 5.04e-02 | 3.68e-03 | 5.69e-02 |
| | **mesch** | 4.55e-01 | 6.96e-03 | 3.40e-02 | 3.43e-02 | 1.77e-02 | 4.63e-02 |
| | **pggan** | 2.22e-01 | 2.22e-02 | 3.31e-02 | 3.39e-02 | 1.78e-02 | 4.65e-02 |
| | **glo**-128 | **0.00e+00** | 9.70e-01 | 9.94e-04 | 3.30e-02 | 5.10e-05 | 9.32e-03 |
| | **glo**-1024 | **0.00e+00** | 7.59e-01 | 1.95e-03 | 8.08e-03 | 1.29e-03 | 4.46e-03 |
| | **glo**-8192 | **2.25e-18** | 1.75e-01 | 3.00e-03 | 3.64e-03 | 1.04e-03 | 3.20e-03 |
| | **glo**-26000 | 2.12e-01 | 3.69e-02 | 4.27e-03 | 4.44e-03 | 4.08e-04 | 4.43e-03 |
| | **aegan**-128 | **0.00e+00** | 9.02e-01 | 1.54e-02 | 1.57e-01 | N/A | 2.82e-02 |
| | **aegan**-1024 | **0.00e+00** | 2.68e-01 | 8.52e-02 | 1.16e-01 | N/A | 8.69e-02 |
| | **aegan**-8192 | **3.17e-27** | 1.61e-01 | 7.42e-02 | 8.84e-02 | N/A | 7.55e-02 |
| | **aegan**-26000 | 1.25e-01 | 1.85e-02 | 9.96e-02 | 1.01e-01 | N/A | 1.00e-01 |
| | **cyclegan**-256 M2F | **0.00e+00** | 4.75e-01 | 9.03e-03 | 1.72e-02 | N/A | - |
| | **cyclegan**-4096 M2F | **0.00e+00** | 2.62e-01 | 6.44e-03 | 8.73e-03 | N/A | - |
| LSUN | **dcgan** (tower) | 7.02e-02 | 1.36e-02 | 7.96e-02 | 8.07e-02 | 1.49e-02 | 7.31e-02 |
| | **dcgan** (bedroom) | 3.65e-01 | 5.34e-03 | 7.06e-02 | 7.10e-02 | 7.03e-02 | 7.09e-02 |
| | **glo**-8192 (bedroom) | **6.70e-06** | 1.70e-01 | 5.45e-03 | 6.56e-03 | 5.37e-04 | 5.01e-03 |
| | **glo**-32768 (bedroom) | 2.62e-01 | 5.40e-02 | 6.58e-03 | 6.25e-03 | 8.40e-04 | 5.44e-03 |
| YOS | **cyclegan**-256 s2w | **1.60e-16** | 3.68e-01 | 1.67e-02 | 2.64e-02 | N/A | - |
| | **cycelgan**-512 s2w | **6.10e-33** | 3.78e-01 | 1.39e-02 | 2.23e-02 | N/A | - |
| MNIST | **dcgan** | 2.41e-01 | 8.85e-02 | 3.00e-02 | 2.75e-02 | 6.89e-03 | - |
| | **glo**-1024 | **0.00e+00** | 6.78e-01 | 2.86e-04 | 8.88e-04 | 1.49e-03 | - |
| | **glo**-16384 | 3.48e-01 | 6.45e-03 | 8.72e-04 | 8.77e-04 | 1.41e-03 | - |
| | **aegan**-16384 | **7.43e-02** | 2.29e-02 | 4.56e-02 | 4.67e-02 | N/A | - |
| CIF-10 | **dcgan** | 5.40e-01 | 3.65e-03 | 2.29e-01 | 2.28e-01 | 1.30e-03 | - |
| | **glo**-1024 | **0.00e+00** | 5.84e-01 | 2.77e-03 | 6.67e-03 | 8.53e-04 | - |
| | **glo**-16384 | 3.48e-01 | 6.45e-03 | 8.72e-04 | 8.77e-04 | 1.41e-03 | - |

Table 3.1: Kolmogorov-Smirnov (KS) p-values, normalized median error difference (MRE-gap) Eq. (3.5), and Median recovery errors (MRE) Eq. (3.4) for a variety of generators. Highlighted values indicate generators for which overfitting of the training set has been detected: (in blue) with the KS test using 1% threshold on p-value, (in green) using 10% threshold on MRE-gap.

will not be able to verbatim recover an identity found in the dataset. For some applications this could be seen as inadequate, such as the domain translation network of StarGAN [29], wherein a user wants to retain identity but change facial features. On the other hand, if a face dataset is considered private or copyrighted, not verbatim copying any training image can be seen as a benefit of the algorithm. Quantifying whether GAN generators really do generalize with respect to identity, using a face identification network like VGG-Face [132], is an interesting issue that we leave for future work.

### 3.4.2 Future Work

Our work is a part of a growing body of research concerned with overfitting of deep generative models [57, 54]. For example, [54] takes a perspective of GAN evaluation, arguing that evaluation with a neural net distance can penalize trivial memorization of the dataset whereas FID cannot. We have a similar perspective, albeit with the goal of merely detecting overfitting and with a simpler approach. Unlike our approach, the NND [68, 54] was able to detect slight overfitting of some GANs, however, the massive size of the validation set and the fact that the NND must be retrained for every generator under evaluation make the analysis computationally burdensome. Additionally, the architecture choice of the NND may be biased to reflect the GAN loss function and not be universal across models, such as the reconstruction based GLO model considered in this work. On the other hand, we present a simple, computationally tractable solution requiring modestly sized (here, just a few thousand) validation images. We found the

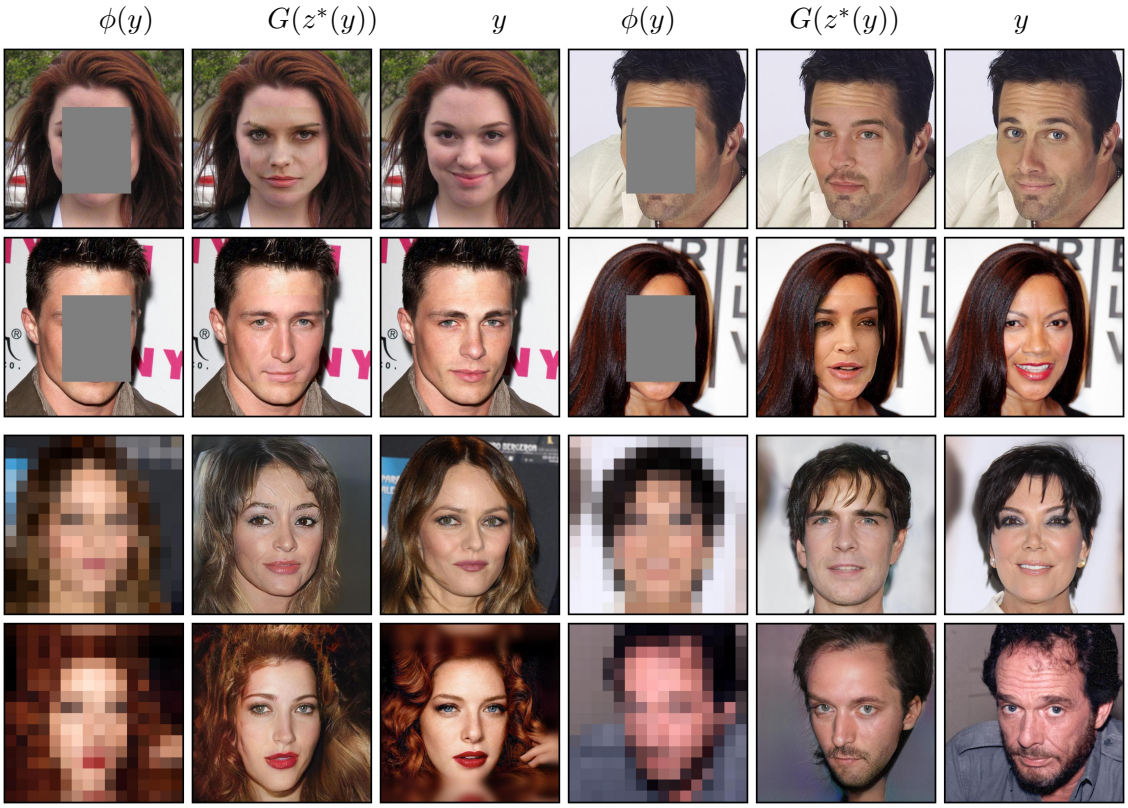



Figure 3.6: Image recovery by solving Eq. (NN$_\mathcal{G}$) with a 1024x1024 generator $G$ by [78]. From left to right: transformed image $\phi(y)$, recovered image $G(z^*(y))$ and ground truth image $y$. The first two rows are image inpainting ($\phi$ is a mask) and next two are image super-resolution of images downsampled by a factor of 64 ($\phi$ is an average pooling).

reconstruction based generator GLO to be interesting from a theoretical perspective. For example, optimization unveiled strong overfitting; training images are nearly verbatim recovered and validation images are blurry (e.g. Fig. 3.3, row 4), whereas FID on GLO samples was insensitive to this difference. Furthermore, a major advantage of our work is the visual interpretability of our results. For example, one can see in Fig. 3.3, the visual reconstructions do in fact reveal visual quality of train versus validation samples. Namely, both training and validation images are well reconstructed for the GAN methods. We believe further work must be done to synthesize the results of [54, 57] and our work. One promising area would be to explore other loss functions. We experimented briefly with perceptual losses (see supplementary) but leave this possibility open. We also think recovery could be guided with a learned NND loss. Another interesting direction is analysis of local overfitting on image patches. Preliminary experiments can be found the in supplementary material, which also show generalization of GAN generators. Finally, Eq. (NN$_\mathcal{G}$) had mixed success for more complex datasets such as LSUN in terms of visual quality. We think that some datasets lead to more complex latent space with many local minima and direct the reader to the supplementary material for more details on optimization.

**Conclusion** In this work, we studied overfitting of deep generators through latent recovery. We saw that a simple Euclidean loss was effective at recovering latent codes and recovers plausible images even after image transformations. We used this fact to study whether a variety of deep generators memorize training examples by asking if the network can generate validation samples. Our statistical analysis revealed that overfitting was undetectable for GANs, but detectable for hybrid adversarial methods like AEGAN and non-adversarial methods like GLO, even for training sets of moderate sizes. Due to the

ever-growing concerns on privacy or copyright of training data and the already widespread application of generative methods, we provide methodology that is a step in the right direction towards analysis of generative overfitting.

# Appendix

## 3.A Additional Results

### 3.A.1 Optimization Failures

We noted that most networks had the ability to exactly recover generated images. This is shown in Fig. 3.A.1, with failure cases highlighted in red. Interestingly, some networks were not able to recover their generated images at all, for example Fig. 3.A.1b was a PGGAN trained on LSUN Bedroom, which did not verbatim recover any image. We think this may suggest a more complex latent space for some networks trained on LSUN, with many local minima to Eq. ($\text{NN}_{\mathcal{G}}$). Because we assert that we are finding the nearest neighbors in the space of generated images, we did not analyze networks which could never recover generated images. It should be noted that some LSUN networks were able to recover their generated images, for instance DCGAN [139].

(a) Recovery for MESCH on CelebA-HQ. (b) Recovery for PGGAN on LSUN-B

Figure 3.A.1: Recovery failure detection with thresholding. First row generated images and second row is recoveries. The MESCH network in 3.A.1a is inconsistent at image recovery, which can be alleviated by restarts. The PGGAN trained on LSUN bedroom in 3.A.1b did not verbatim recover any image.

### 3.A.2 Recovery Success Rate

Disregarding networks which could not recover generated images, some networks had higher failure rates than others. To determine failure cases numerically, we chose a recovery error threshold of MSE $< .1$ to signify a plausible recovery for real images (for generated images a much smaller threshold of MSE $<$ .025 can be used, which corresponds to verbatim recovery, e.g. in 3.A.1). Table 3.A.1 summarizes recovery rates for a few networks. The MESCH resnets were notably less consistent than other architectures. To study if these failures were due to bad initialization, we tried simply restarting optimization 10 times per image, and saw the success rate go from 68% to 98% (shown in Table 3.A.1 as MESCH-10-RESTART). This shows that likely all training and generated images can be recovered decently well with enough restarts.

Table 3.A.1: Recovery success rate for real and generated images. Percentages indicate rate of recoveries with a MSE $< .1$, which corresponds to a plausible synthesis. Failures are related to bad initialization, as simply restarting greatly increases success rate.

| | train | test | generated |
|---|---|---|---|
| MESCH | 68% | 67% | 67% |
| MESCH-10-RESTART | 98% | 99% | 96% |
| DC-CONV | 82% | 82% | 100% |
| PGGAN | 97% | 96% | 95% |

## 3.B Comparison with Other Loss Functions

We visually compare in Fig. 3.B.1 the simple Euclidean loss used in this paper for analyzing overfitting (*i.e.* $\phi = \text{Id}$ in Eq. ($\text{NN}_\mathcal{G}$)) with other operators:

- $\phi =$ pooling by a factor of 32 (as used in applications for super-resolution);
- $\phi =$ various convolutional layers of the VGG-19 (*i.e.* the *perceptual loss* previously mentioned in the paper).

While the perceptual loss has been shown to be effective for many synthesis tasks, it appears to hinder optimization in the case when interacting with a high quality generator $G$. Observing the recovered images in Fig. 3.B.1, the pooling operator seems to help with recovered textures as it relaxes the loss, while still recovering a highly similar face to the naive loss $\phi = \text{Id}$.

## 3.C Convergence results

In general, optimization was successful and converges nicely for most random initializations. We provide numerical and visual evidence in this section supporting fast and consistent convergence of LBFGS compared to other optimization techniques like SGD or Adam.

### 3.C.1 Protocol

To demonstrate that the proposed optimization of the latent recovery is stable enough to detect overfitting, the same protocol is repeated in the following experiments. We used the same 20 random latent codes $z_i^*$ to generate images as target for recovery: $y_i = G(z_i*)$ . We also used 20 real images as targets the same as in Section 2 for local recovery. We also initialized the various optimization algorithms with the same 20 random latent codes $z_i$. We plot the median recovery error (MRE) for 100 iterations. This curve (in red) is the median of all MSE curves (whatever the objective function is) and is compared to the 25th and 75th percentile (in blue) of those 400 curves.

### 3.C.2 Comparison of optimization algorithm

We first show the average behavior in Fig. 3.C.2 the chosen optimization algorithm (LBFGS) to demonstrate that it convergences much faster than SGD and Adam. A green dashed line shows the threshold used to detect if the actual nearest neighbor is well enough recovered (MRE $= 0.024$). One can see that only 50 iterations are required in half the case to recover the target image. Figure 3.C.1 compares recovery images for PGGAN obtained with LBFGS and SGD optimization algorithms, demonstrating that LBFGS gives most of the time better results.

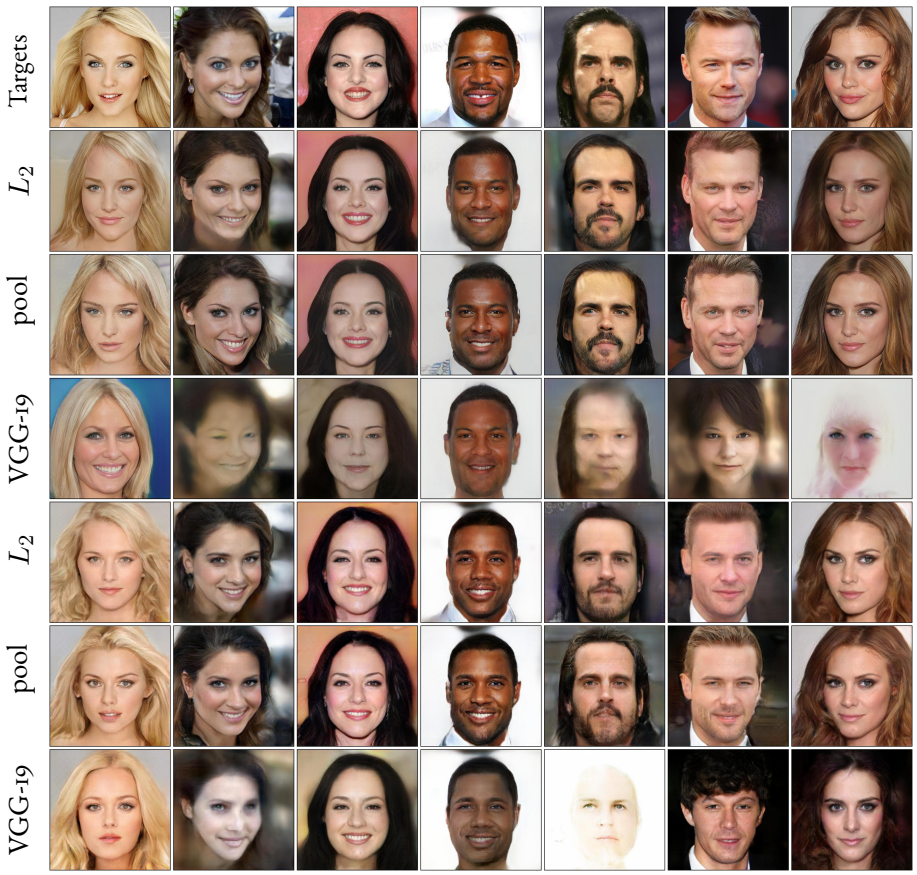



Figure 3.B.1: Using other loss functions for image recovery. The first row is target images from CelebA-HQ, the next three rows are recovery from PGGAN network and the final three are from MESCH generator. Pooling seems to help slightly with textural details without hindering recovery of facial pose. Surprisingly, a VGG-19 loss hinders recovery for the PGGAN generator.

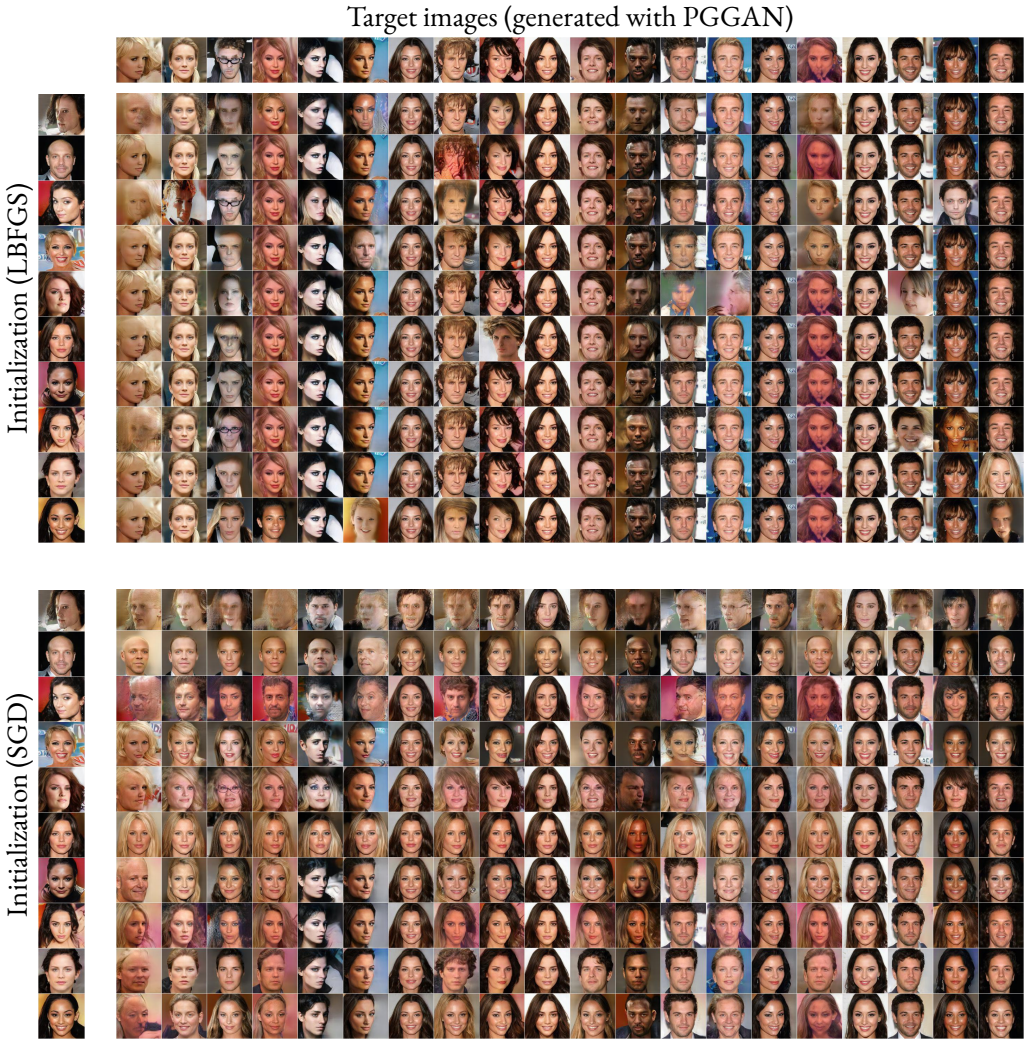
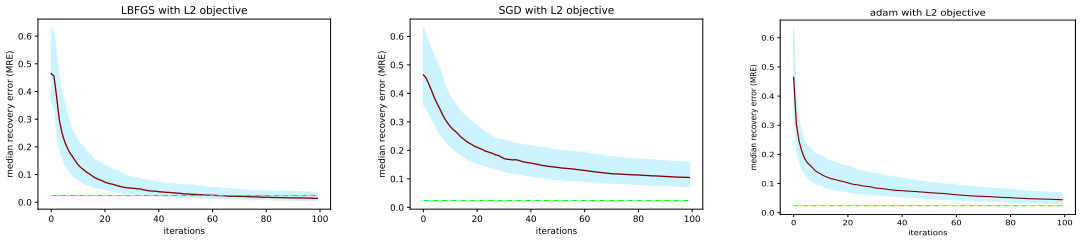



Figure 3.C.1: Visual comparison of recovery with **LBFGS** (top) and **SGD** for the Euclidean loss (see (NNG) optimization problem in the main paper.). First row: target (generated) images $y_i = G(z_i^*)$. First column: initialization ($G(z_i^{(0)})$). Second column: optimization after 100 iterations ($G(z_i^{(100)})$). LBFGS gives much better results than SGD that is much slower to converge, but still needs sometimes some restart (here shown without restarting).




LBFGS SGD Adam

Figure 3.C.2: First row: median recovery error (MRE) curve. Second row: 400 superimposed recovery error curves for 20 images with 20 random initialization. LBFGS (first column) converges faster than SGD or Adam (second and third column respectively).

Chapter 4

# Generating Private Data Surrogates

In the previous chapter, we demonstrated how latent recovery could be used to diagnose when certain deep generative models overfit their training data. In the extreme case, reconstruction errors for training points had a distribution which was disjoint from held out test samples. In this case, it's possible to infer the model's training set, even without any a priori knowledge of which training points were used, by merely choosing reconstruction errors that are sufficiently low. In this chapter, we'll introduce the concept of a membership attack, which is a binary classification problem that tries to do just this. We'll use the latent recovery technique we developed in the previous chapter (namely Eq. $\text{NN}_{\mathcal{G}}$) and even improve this technique via the use of an autoencoder. We also use our observation that GANs trained with sufficient data demonstrate robustness to membership attacks; thus, the goal of this chapter is to create synthetic data that is both useful and yet does not reveal information about specific dataset samples. Since publication, there have been many works on training GANs in low data settings [183, 80]. As the primary contribution, we demonstrate how to construct surrogate datasets, using images from GAN generators, labelled with a classifier trained on the private dataset. Next, we show this surrogate data can further be used for a variety of downstream tasks (here classification and regression), while being resistant to membership attacks. We study a variety of different GANs proposed in the literature, concluding that higher quality GANs result in better surrogate data with respect to the task at hand.

Finally, in the next chapter, we'll show that models trained with few data points are vulnerable to the attacks presented in this chapter, and we'll present a potential defense using the tools developed therein. Additional insights based on a different membership attacks are presented in Chapter 6, wherein bias, and not just dataset size, can effect privacy.

**Continued Work** At the time of publication, both membership attacks against generative models and attack defenses were relatively new subjects. Since then, a variety of other membership attacks and defenses have been proposed. Most membership attacks exploit the models loss function, or a proxy loss function retrained from model outputs [156, 55]. In [98], the entirety of the loss during training is used to infer training points and is more sensitive to output only attacks. Other works try to use less information from model outputs, such as the work in [31] which uses hard labels only rather than continuous losses. There are also an even stronger form of membership attacks, extraction attacks, which don't require training input images but rather just conditional labels or text. We explore such an attack in Chapter 7.

We also note attack defense guarantees using differential privacy have significantly improved. If PATE-GAN [186] was unable to provide realistic synthesis for reasonable guarantees, new methods using diffusion models have made progress [51]. The DDPM approach in [51] has largely the same goal as in this chapter, i.e., to release surrogate images that are still useful for other machine learning problems, or for aggregation for more large scale training. Still, these methods have much lower quality than non-DP training and still provide "trivial" privacy bounds; in other words as DP only provides an upper bound on the information that leaks, in practice these upper bounds are too weak and many acceptable DP parameters are still empirically vulnerable to membership inference [71].

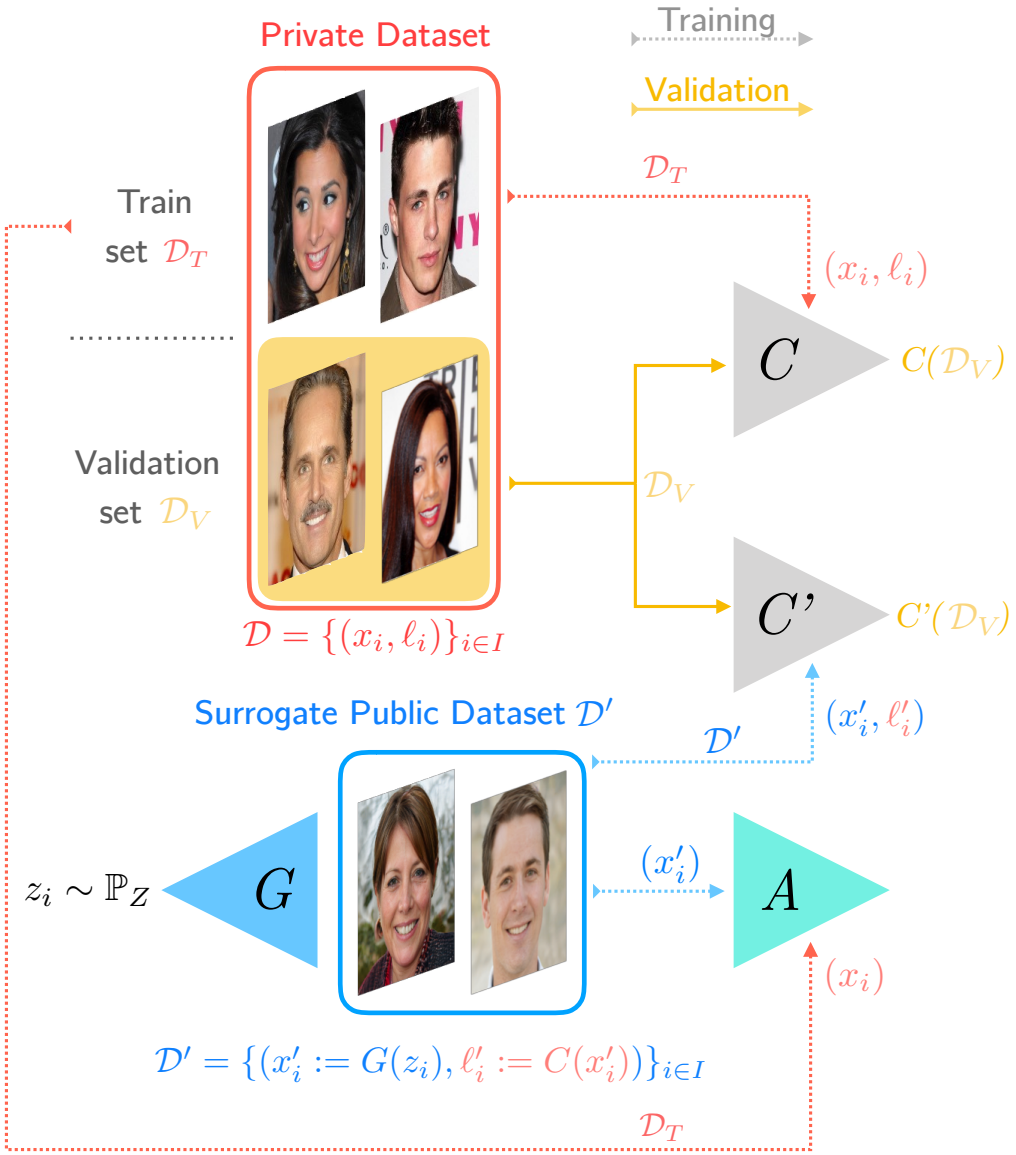



Figure 4.0.1: **Overview of the proposed framework for creating private data surrogates and its application to train a private task-driven network.** In a nutshell, the data surrogate $\mathcal{D}'$ is simply obtained by combining image samples $x'_i$ from a generator network $G$ (e.g. using GAN, trained with an adversarial network $A$ with latent code $z_i$ sampled from a random distribution $\mathbb{P}_Z$) and associating them with plausible labels $\ell'_i$ obtained from a classifier $C$ trained on the private train dataset $\mathcal{D}_T$ composed of pairs $(x_i, \ell_i)$ of real images with labels. From this public dataset, it is possible to train a privacy preserving classifier $C'$ displaying similar performance and accuracy (in practice by comparing $C'(\mathcal{D}_V)$ and $C(\mathcal{D}_V)$ on a separate validation set $\mathcal{D}_V$). We further demonstrate empirically that the obtained public dataset $\mathcal{D}'$ (and by composition the network $C'$) is robust to membership attacks described in Algorithm 2 and performed by a network $A$.

## 4.1 Context and Motivation

The fantastic recent advances in deep learning are strongly related to the existence of public datasets. Such datasets not only allow researchers to learn from and experiments with the data but also to measure progress and challenge themselves with other researchers in competitions. If Pascal VOC [44] was among the first, many followed such as ImageNet [36] for object classification or recently CelebA-HQ [78] as

a benchmark for generative models. The contemporary machine learning research landscape would undoubtedly be very different without such datasets.

However, several important application areas do not fully benefit from the progress of machine learning because of the lack of massive public datasets. Indeed, building such datasets is difficult due to privacy issues that will inevitably arise. Even if only the model parameters are released, convolutional neural networks have been shown to memorize data and leak information about their training sets [154, 189].

This poses huge problems for the applicability of deep learning. For instance, while deep learning has shown huge potential for diagnostic assistance in the medical domain, training data is often hard to obtain due to strict privacy laws on medical data. Likewise, biometric or other user data in mobile applications often is privately held, and can not be shared for more public usage. For generative applications, training data can even be visually apparent in models that overfit [176], which means copyright for data needs to be obtained before training.

As a fundamental solution to these problems, we propose releasing a generated dataset which highly resembles the original data, but exhibits privacy. This is different from the standard approach to privacy, which involves adding privacy to a model e.g. during training [41]. In many ways, directly releasing privatized data is more useful, as it can be used for a variety of tasks, whereas a private model is limited to one.

### 4.1.1 Membership attacks

One common attack against machine learning models is the *membership attack*, which discerns data points that were used for training. Neural networks performing classification on images are known to be vulnerable to such attacks. For instance, in [156], a membership attack was successfully performed against an MNIST model, even if the network parameters were not available to the attacker. In essence, these attacks exploit neural networks tendency to overfit, and can use simple cues such as output logit entropy.

In response to such membership attacks, a huge amount of recent works have proposed heuristic or theoretical attack defenses [130, 100, 41]. There is the mathematical framework of *differential privacy* [41], which is often used to provide privacy guarantees in machine learning frameworks. For example, classifiers with privacy guarantees can be achieved with the teacher ensemble mechanism [130] (PATE), or through knowledge transfer [128]. Empirical defenses have also been proposed for membership attacks in particular. For example, [121] proposes adding an adversary during training which simulates an attacker and therefore minimizes a utility privacy trade-off.

### 4.1.2 Privacy of Generative Models

While generative adversarial networks (GANs) [52] are a relatively new advent, they certainly have changed the landscape of machine learning, for example, achieving state of the art in image synthesis [82, 118, 66] and a plethora of other generative tasks.

Recent work has been devoted to defining and measuring overfitting in generative models, for example, in [176, 68], and [54]. As an approach to evaluating GAN quality, [68] proposed training a third network to discriminate validation and generated samples after GAN training had completed. Similarly, in [54], the authors used a neural net distance to define a GAN evaluation metric which importantly penalizes models memorizing training samples, unlike the Frechet Inception Distance (Eq. 2.1). In the work of [56], membership attacks are directly performed against the discriminator of GANs (referred to as Adversarial Network $A$ in Fig. 4.0.1). Finally, [176] proposes directly recovering training and test images with the generator and comparing recovery error, with the hypothesis that training images should exhibit smaller error.

Last, some recent works approach the question of deep learning with privacy via direct sanitation of data for release. Mirjalili *et al.* [117] used convolutional auto-encoders to remove other information

(*e.g.*gender) than the one related to identity from training face images. Sokolic *et al.* [160] proposed a data sanitization mechanism during which users' data is modified to prevent specific attacks before these data are actually used for training. The same goal is sought by Bertran *et al.* [16], Wu *et al.* [181] or Rezaei *et al.* [143], by defining a collaborative sanitization function retaining valuable information for the tasks while eliminating private information. Finally, PATE-GAN [186] proposes a framework to generate data using a GAN like framework and adopting privacy via the PATE mechanism.

### 4.1.3 Contributions and outline

In this work, we also seek to directly release data immune to membership attacks by presenting two contributions: First, a simple but efficient surrogate technique is proposed to generate a synthetic dataset that can be released in public, which ideally achieves the same level of performance as the original dataset while ensuring its protection to membership attack. The methodology for generating, labeling and evaluating this data is documented in Fig. 4.0.1.

Second, two recent membership attacks against generative models are executed against GAN generators and shown to be ineffective. Finally, while this study is preliminary, we evaluate two face datasets, CelebA-HQ and UTK-Face, across a variety of state of the art GAN generators as well as various tasks to demonstrate consistency of our observations.

The rest of the paper is organized as follows: Section 4.2 exposes the construction of the surrogate dataset as well as its evaluation. Section 4.3 details the proposed membership attack protocol to assess the efficiency of our surrogate dataset. Section 4.4 shows experimental results, followed by a discussion in Section 4.6.

## 4.2 Surrogate Data Creation and Evaluation

Typically, methods offering privacy in machine learning have a privacy and utility trade-off, for instance losing generation quality in return for privacy [122, 186, 130]. For example PATE-GAN [186] demonstrates the utility of generated data through unsupervised tasks under various privacy guarantees.

We propose evaluating utility on supervised tasks by first labeling images generated from an unconditional GAN and then taking the standard validation accuracy on real images. Figure 4.0.1 details the entire pipeline:

- train a classifier $C$ and a generator $G$ from the private training dataset $\mathcal{D}_T$;
- build and release publicly a synthetic dataset $\mathcal{D}'$ : any sample from this set is randomly generated as $x' = G(z)$, where $z \sim \mathbb{P}_Z$ (the training latent distribution) and associated with the predicted label $\ell = C(x')$;
- train a classifier $C'$ from the surrogate dataset $\mathcal{D}'$;
- evaluate and compare $C'$ and $C$ on a validation set $\mathcal{D}_V$, in terms of privacy and utility.

Experiments detailed in Section 4.4 demonstrate for various GANs that the surrogate classification network $C'$ can achieve performance similar to the one trained directly on private data.

Next section presents two membership attacks used to assess privacy of surrogate data.

## 4.3 Assessing privacy of Generative Models by Membership Attacks

In this work, we consider two membership attack models against generative networks. Both approaches utilize overfitting of the generative network $G$, with respect to some attack function $A$. Following LOGAN [56], we discern samples that likely belong to the training set by selecting those with the lowest values for the attack function $A$. For instance in LOGAN, $A$ is taken to be the trained adversarial network (as illustrated in Figure 4.0.1), in which case the training set is taken as those samples which the

discriminator most confidently predicts to be real. Algorithm 2 details precisely how this attack is performed.

**Input:** Training set $\mathcal{D}_T = \{y_1, \cdots, y_N\}$ and validation set $\mathcal{D}_V = \{y_{N+1}, \cdots, y_{2N}\}$.

1: Set the attack score function $A$, either from the recovery loss function in Eq. (4.1) or the trained adversarial network [56].

2: Let $y_i \in \mathcal{D}_T \cup \mathcal{D}_V$, such that

$$\begin{cases} y_i \in \mathcal{D}_T & \text{if } i \leq N \\ y_i \in \mathcal{D}_V & \text{if } N+1 \leq i \leq 2N \end{cases}$$

3: Sorted indices: $I \leftarrow argsort\{A(y_i)\}_{1 \leq i \leq 2N}$

**Output:**

4: Estimated set of training images: $\mathcal{T} \leftarrow \{y_{I(i)}\}_{1 \leq i \leq N}$

5: Membership attack accuracy:
$Acc \leftarrow |I \cap \{i : 1 \leq i \leq N\}|/N$

**Algorithm 2:** Membership attack

### 4.3.1 Recovery Attacks

In [176], training and validation images were recovered using optimization. Generative networks were said to overfit if the statistics of training and validation recovery errors were different in some measure, for example the difference of medians. In [97], recovery errors were used similarly to perform membership attacks, where the optimization was performed over the parameters of an input layer to the generator, rather than the latent codes themselves.

In [14], GAN inputs are recovered with a trained encoder, in order to visualize phenomena such as mode dropping. As inversion of many layer generators can be difficult, they use layer-wise training with a perceptual loss. Here, we found the perceptual loss to be sufficient for inversion. We define our attack function (referred to as $A$ in Algorithm 2 and illustrated in Figure 4.3.1) as the perceptual loss of this inversion as

$$f_E(y_i) := \|\phi(G(E(y_i))) - \phi(y_i)\|^2 \tag{4.1}$$

where an image $y_i \in \mathcal{D}_T \cup \mathcal{D}_V$ is in either the training or validation sets, $\phi$ are image features (such as convolution layers of VGG-19), $G$ is the GAN generator trained on $\mathcal{D}_\mathcal{T}$ and $E$ is the attack encoder. The primary advantage of using an encoder is a fast feed-forward image recovery, rather than optimizing per image. We train $E$ solely on generated samples

$$\min_E \mathbb{E}_{z \sim \mathbb{P}_Z} f_E(G(z)) \tag{4.2}$$

where $\mathbb{P}_Z$ is the random distribution of latent codes used to train the generative model $G$. As is common with encoder design, we choose the architecture of $E$ to resemble a transpose of G, with upsampling replaced by downsampling.

We consider using perceptual features for $\phi$ from the VGG-19 network (see for instance [73]), but also consider using other feature networks, such as VGG-Face [133] or even $\phi = Id$ for an image domain loss. Furthermore, notice that we only train $E$ on generated samples $G(z)$ and keep real ones aside so that $E$ is not dependent on the potential overfitting pattern of $G$.

### 4.3.2 Discriminative Attacks

Following the work of [56], *Discriminative* membership attacks are performed against GANs by using the adversarial network $A$ to sift between test and train images. Compared to the recovery attack, this

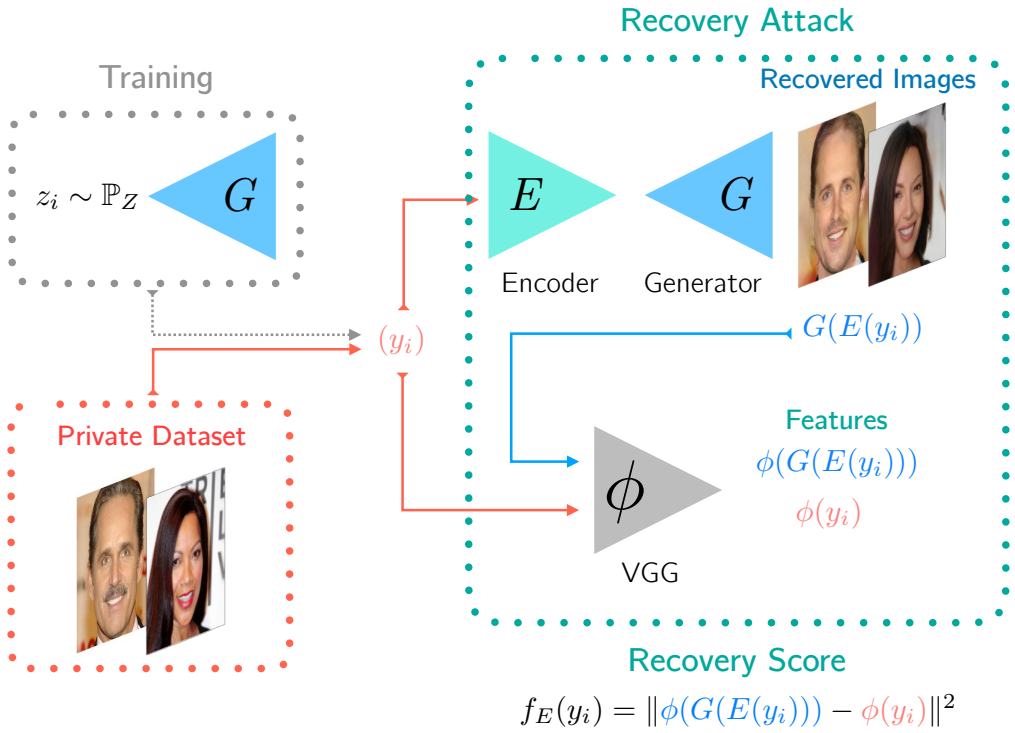



Figure 4.3.1: **Overview of the proposed membership recovery attack on a generative network.** This attack relies on an encoder which acts as the pseudo-inverse of a given generative network to detect overfitting. The encoder $E$ is trained to recover the latent codes from randomly generated samples $y_i = G(z_i)$ by minimizing the discrepancy between features $\phi$ (*e.g.*VGG-19 network) of $y_i$ and $G(E(y_i))$. During the attack, images $y_i \in \mathcal{D}_T \cup \mathcal{D}_V$ are sampled from the private dataset, *i.e.*$\mathcal{D}_T$ used to train the generator $G$ and $\mathcal{D}_V$ unseen during training; the recovery score is then used to sift training images as described in Algorithm 2.

methods requires the discriminator $A$ to be publicly released, which is usually not the case as it does not fulfill any purpose after training. Recall that $A$ is trained along with the generator $G$ on the training set $\mathcal{D}_T$, as illustrated in Fig. 4.0.1. For convenience, we assume here that the discriminator $A$ is trained to score 0 on real images $x_i \in \mathcal{D}_T$ and 1 on generated images $x'_i = G(z_i)$.

## 4.4 Results

We train the following GANs and will use the abbreviations in parentheses: Progressive GANs presented in [78] (PGGAN) using official code, the zero-centered gradient penalty Resnet in [112] with the official code (MESCH) and finally deep convolutional GAN [139] (DCGAN) and least squares GAN [106] (LSGAN) with our own implementation.

### 4.4.1 Encoder training for latent recovery

As described in Section 4.3.1 and Figure 4.4.1, we train for each GAN an encoder for pseudo-inversion using various perceptual loss functions.

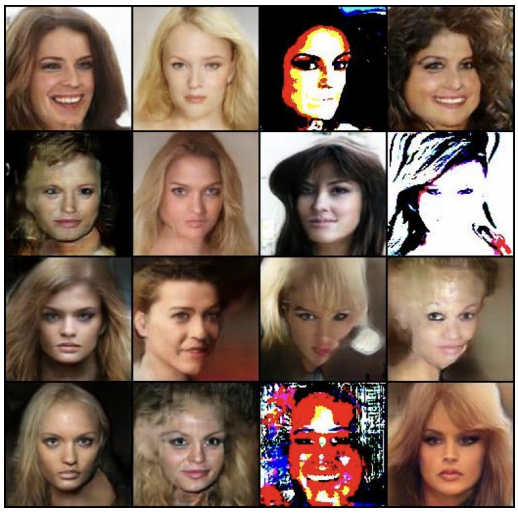
Figure 4.4.1 shows recovery using different recovery methods and different GAN generators. Note that a simple image domain recovery (as in [176]) for LSGAN is not reliable, whereas the encoder approach is. Note as well that after training of encoder $E$ is complete, it only takes one network evaluation to retrieve a recovery score, compared with the hundreds of evaluations required for [176]. Furthermore,

training of the encoder is quite fast; here we used 25 thousand generated images for training and witnessed convergence of the recovery scores.

Last, Figure 4.4.1 illustrates the interest of latent recovery attack with the proposed perceptual encoder $E$ (see Section 4.3.1) on PGGAN and CelebA-HQ. Not only does the encoder speed up the attack it also improves the stability of recovery. Note also the histogram of recovery scores for test and train images are roughly the same for the GAN models, which results in an unsuccessful membership attack.

Real images $y_i \in \mathcal{D}_T \cup \mathcal{D}_V$

Recovered images $G(E(y_i))$ from PGGAN with Encoder



Recovered images $G(z_i)$ from LSGAN with latent recovery optimization [176]

Recovered images $G(E(y_i))$ from LSGAN with Encoder

Figure 4.4.1: **Illustration of latent recovery attack for PGGAN and LSGAN.** Real images $y_i$ from the test and train dataset are analyzed with the encoder $E$ then synthesized with the generator $G$ (right). The encoder $E$, trained on generated images, improves latent recovery in comparison with explicit optimization [176]. The discrepancy between recovery errors (see Eq. (4.1)) is used to perform membership inference attack, which is unsuccessful in this setting (52% as reported in Table 4.4.2).

### 4.4.2 Attribute Recognition on CelebA-HQ

The CelebA-HQ dataset [78] is a typical GAN benchmark dataset, which includes 40 attributes $\ell_i$. Because many of these attributes contain label noise or are redundant, we choose five prominent attributes: *gender*, *smiling*, *young*, *glasses*, *blond*.

Table 4.4.1 and Table 4.4.3 shows that a classifier $C'$ trained with a surrogate dataset $D'$ (generated with the proposed approach described in Section 4.2) performs as well as a classifier $C$ directly trained on the private dataset $D$. As demonstrated by the FID (Eq. 2.1) scores that evaluate the quality of the generator (lower is better), the drop in performance strongly correlates with the quality of the synthesized images used for training.

| | | Gender | Smiling | Average (5 attributes) | Drop in Performance | FID |
|---|---|---|---|---|---|---|
| $C$ | Real Data | 94.50 | 85.20 | 90.64 | - | - |
| $C'$ | DCGAN | 91.90 | 82.10 | 86.50 | 4.14 | 67.07 |
| | MESCH | 92.60 | 81.45 | 88.90 | 1.74 | 26.31 |
| | LSGAN | 92.10 | 80.80 | 88.35 | 2.29 | 42.01 |
| | PGGAN | 93.10 | 83.05 | 89.35 | **1.29** | **19.17** |

Table 4.4.1: Performance of various surrogate datasets on the CelebA-HQ [78] binary attribute recognition task, for VGG Face features. Top row represents a classifier $C$ trained on the original dataset $\mathcal{D}_T$, subsequent rows represent classifiers $C'$ trained with GAN images that are labelled with $C$ (see Section 4.2 for details). Accuracy represents percent correct on a validation set $\mathcal{D}_V$ with the random guess baseline being 50%. The quality measure of generated images, assessed by the Frechet Inception Distance (FID) [59], is strongly correlated with the performance of the surrogate classifiers $C'$.

| | $L_2$ Recovery | VGG-Face Recovery | VGG-19 Recovery | Discriminator $D$ |
|---|---|---|---|---|
| DCGAN | 54.1 | 54.5 | 51.6 | 57.1 |
| MESCH | 53.9 | 50.8 | 52.5 | 50.1 |
| LSGAN ($\lvert\mathcal{D}_T\rvert = 26k$) | 54.8 | 54.1 | 54.0 | 62.9 |
| LSGAN ($\lvert\mathcal{D}_T\rvert = 5k$) | 58.1 | 56.2 | 57.8 | 99.4 |
| PGGAN | 52.0 | 50.3 | 52.1 | N/A |

Table 4.4.2: Membership attack accuracies (in %) for various GAN methods trained on the CelebA-HQ dataset and various attack methods (see Algorithm 2). When not specified otherwise, the size of the training dataset is $\lvert\mathcal{D}_T\rvert = 26k$ and for the validation set $\lvert\mathcal{D}_V\rvert = 2k$. GAN methods are reported in the first column. The next three columns use latent recovery attack with loss function $f_G$ (see Eq. 4.1), with $\phi$ taken to be the identity, VGG-Face or VGG-19 features respectively. The final column reports the discriminative attack accuracy with the discriminator $D$ from the GAN training (the discriminator of PGGAN requires feeding a whole batch which prevented us to implement this attack). As a baseline, the same discriminative attack is done on LSGAN with a smaller training dataset (5k) demonstrating that in such setting the discriminator network is capable of memorizing almost perfectly the entire training dataset.

### 4.4.3 Safety to Membership Attacks

We assess safety of every GAN model trained versus the membership attacks described in Section 4.3. Table 4.4.2 and Table 4.4.4 shows that membership attacks described in Algorithm 2 are largely unsuccessful when the size of the training set $|\mathcal{D}_T|$ is large enough. For evaluation of membership attack accuracy, we have used $N = 2000$ images from $\mathcal{D}_T$ and $\mathcal{D}_V$.

The discriminator attack of LOGAN is overall most successful, for example achieving nearly $63\%$ accuracy on LSGAN even with $26k$ training images, although interestingly the regularized discriminator in MESCH will yield unsuccessful discriminator attacks even for small datasets, while appearing to be slightly vulnerable to recovery attacks. Note that this observation is not in contradiction to the results in LOGAN [56], which showed successful discriminator attacks across the board, as that study primarily studied small training sets and focused on the DCGAN training technique. To show our observation is consistent with LOGAN, we also include LSGAN with a small training set of $|\mathcal{D}_T| = 5k$, which yields a near perfect discriminator attack (the discriminator outputs can be perfectly separated, see Fig. 4.4.2, top left).

Finally, we note that a discriminator attack is a fairly unrealistic scenario, as the discriminator parameters are typically never used from an application standpoint. Furthermore, an attacker has to have moderate knowledge of the dataset if he wants to retrain a discriminator (as in [68]), which is somewhat counter intuitive to the attack in the first place. On the other hand, optimization of encoder $E$ using Eq. (4.1) can be done merely by sampling $G$, as is done in this document, but requires the parameters of $G$. Finally, we note that we did not in fact use the labels given to the surrogate data $D'$ to perform our membership attacks. We do not believe this would significantly affect the membership attack accuracy, as this information is implicitly available to the attacker, for example when using a semantic network like VGG-Face for a recovery attack.

### 4.4.4 Additional experiments on UTK-face with age regression

Tables 4.4.3 and 4.4.4 report additional results on the UTK-face dataset [193] that are consistent with the experiments on CelabA-HQ. Particularly, networks with higher quality samples (as measured by the FID score) are performing better on the task. This dataset is composed of 20k images of faces ranging from 0 to 116 years old. The task networks are here trained on $|\mathcal{D}_T| = 10k$ images to perform age regression instead of binary attribute classification. The test set is again composed of $|\mathcal{D}_V| = 2k$ images.

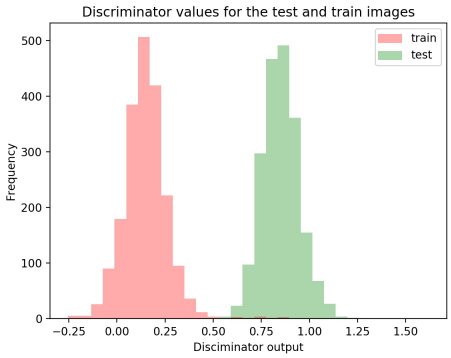
The only difference with the previous experiments on CelebA-HQ (in Table 4.4.1) is that the performance of the task networks $C$ and $C'$ for age regression is measured using median absolute difference between estimated labels $C(x_i)$ and the ground truth $\ell(x_i)$, which writes:

$$\text{MAD}(C) = \text{median}\{|C(x_i) - \ell(x_i)|, x_i \in \mathcal{D}_V\}. \tag{4.3}$$

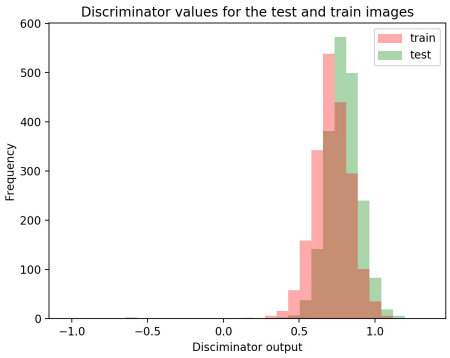
Finally, note that in Table 4.4.4 the discriminative attack on LSGAN is slightly more successful, likely due to the fact of the much smaller training set size of $|\mathcal{D}_T| = 10k$. The discriminative attack seems far less consistent, as it performs no better than random guessing on the DCGAN network with the same training set size. Future work should investigate discriminative attacks in more detail, across a wider range of datasets, GAN techniques and dataset sizes.

## 4.5 Additional Visual Results

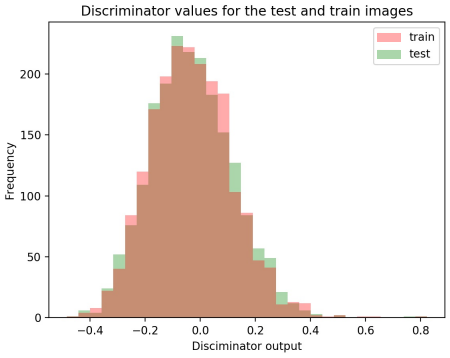
Here, we provide some visual results for the the attacks in Fig. 4.4.2.



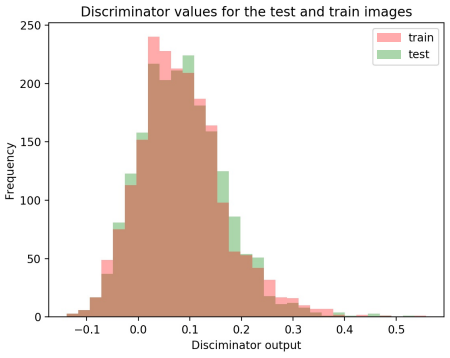

LSGAN, $|\mathcal{D}_T| = 5,000$, Acc $= 99.4\%$




LSGAN $|\mathcal{D}_T| = 26,000$, Acc $= 62.9\%$




MESCH $|\mathcal{D}_T| = 5,000$, Acc $= 50\%$




MESCH $|\mathcal{D}_T| = 26,000$, Acc $= 50.1\%$

Figure 4.4.2: **Histogram of attack scores based on the Discriminator** $A$ for networks trained on CelebA-HQ. LSGAN has high quality samples but is more susceptible to membership attacks. See Fig. 4.5.1 in the appendix for corresponding visual results.

| | | Age (MAD error, in years) | Change in Performance (in years) | FID |
|---|---|---|---|---|
| $C$ | Real Data | 5.22 | - | - |
| $C'$ | DCGAN | 12.03 | 6.81 | 89.68 |
| | LSGAN | 5.56 | 0.34 | 31.05 |
| | PGGAN | 5.12 | **-0.10** | **30.65** |

Table 4.4.3: Performance of various surrogate datasets on the age regression task of **UTK-Face** [193], using VGGFace features. Top row represents a classifier trained on the original dataset, subsequent rows represent classifiers trained with GAN images (see Section 4.2). MAD is the Median Absolute Difference (see Eq. 4.3) on the predicted versus ground-truth age for the validation set $\mathcal{D}_V$ (lower is better). FID scores are reported in the last column (lower is better) to assess the quality of generated images.

## 4.6 Discussion and Conclusion

In this work, we presented a technique for the public release of data using GANs and verified empirically the data appear retains its utility while gaining privacy. More precisely, we have demonstrated that GANs surrogates are effective for age regression and face attribute classification. To verify privacy, two different inference attack mechanisms have been investigated. The first one is based on image recovery and the second one on the discriminator optimized during GAN training, which has been reported to overfit the

| | $L_2$ Recovery | VGG-Face Recovery | VGG-19 Recovery | Discriminator |
|---|---|---|---|---|
| DCGAN | 52.3 | 53.5 | 52.1 | 50.9 |
| LSGAN | 53.4 | 53.9 | 53.6 | 75.8 |
| PGGAN | 54.7 | 56.8 | 54.1 | N/A |

Table 4.4.4: Membership attack accuracies (in %) for various GAN methods trained on the **UTK-Face** dataset and various attack methods (see Algorithm 2). When not specified otherwise, the size of the training dataset is $|\mathcal{D}_T| = 26k$ and for the validation set $|\mathcal{D}_V| = 2k$. GAN methods are reported in the first column. The three next columns use latent recovery attack with loss function $f_G$ (see Eq. 4.1), with $\phi$ taken to be the identity, VGG-Face or VGG-19 features respectively. The final column reports the discriminative attack accuracy with the discriminator $D$ from the GAN training.

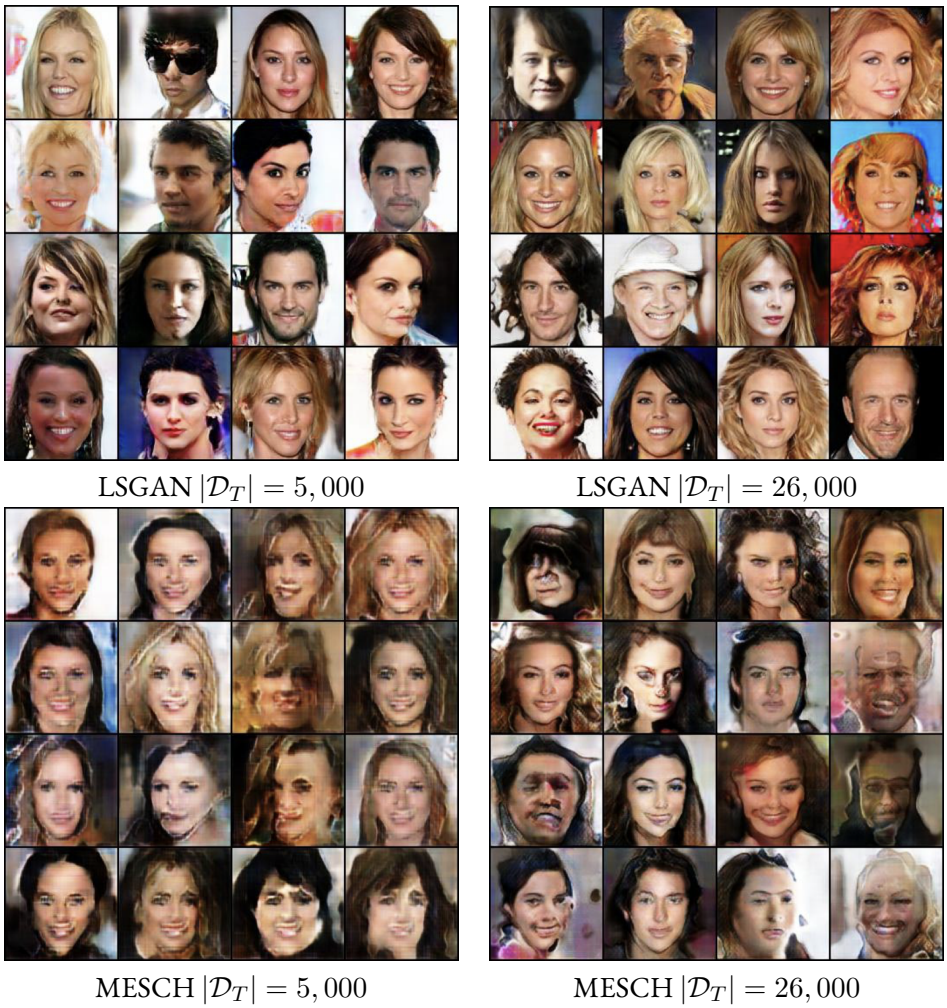



Figure 4.5.1: Visual results corresponding to the histograms in Fig. 4.4.2, on CelebA-HQ. LSGAN has decent quality but is susceptible to membership attacks. MESCH is trained with regularization, which may explain it's robustness verses attacks, but has poor quality in the low data setting.

training dataset. A major advantage to the presented method is that it can work off-the-shelf with any GAN generator. On the other hand, complex training procedures such as PATE-GAN only exacerbate already unstable GAN training and may result in low quality data samples. While we demonstrated

data surrogates greatly reduce vulnerability to membership attacks, more insight should be shed into the mechanism behind this. Hopefully, this would allow for mathematical guarantees instead of purely empirical ones.

Future investigations will include the question of adapting this strategy to conditional GANs for situations where training an additional classifier could be avoided. Additionally, GANs need large datasets for training which raises the question of the extension of the proposed framework for small private datasets. The extension to multiple private datasets is also not straightforward, but would, however, provide a useful tool for distributed learning.

Chapter 5

# Efficient GAN Learning with Parameter Sharing

Thus far, we have demonstrated several ways to evaluate GAN generated images, in terms of quality, memorization and privacy. Namely, in the previous chapter, we saw how GANs appear to provide empirical privacy guarantees when trained with sufficient data. However, this chapter will use tools to train GANs in the very low data setting (for instance, just a few thousand samples). Here, we'll see that GANs are more vulnerable to membership inference attacks. We'll introduce a parameter sharing technique, wherein we'll re-use knowledge learned from one dataset for training on another. We show that parameter sharing is an effective way to train with few parameters in terms of quality, and finally show that training with few parameters can negate the ability of an attacker to perform membership inference. Thus, we largely use the tools developed in Chapter 2 for quality assessment and Chapter 4 for privacy assessment.

In this chapter, we propose a new parameter efficient sharing method for the training of GAN generators. While there has been recent progress in transfer learning for generative models with limited data, they are either limited to domains close to the original one, or adapt a large part of the parameters. This is somewhat redundant, as the goal of transfer learning should be to re-use old features. In this way, we propose width wise parameter sharing, which can learn a new domain with ten times fewer trainable parameters without a significant drop in quality. Previous approaches are less flexible than our method and also fail to preserve image quality for challenging transfers. Finally, as our goal is ultimately parameter re-use, we show that our method performs well in the multi-domain setting, wherein several domains are learned simultaneously. As the trend in GAN models is learning larger models with millions of parameters, we believe our approach is a step towards more distributable and generic GAN models.

**Continued Work** Whilst we study StyleGAN-ADA[84] for the low data settings, several advancements have been made since this publication. For instance, in Insgen [183], sample efficient learning is achieved by augmenting the discriminator network via a contrastive loss. More in line with the topic of this chapter are methods that learn small, domain specific sets of new parameters. For DDPMs, methods such as Textual Inversion [49] are trained on normally just a few samples, and leverage the general synthesis capacity of stable diffusion. Also, LoRA methods [61] originally designed for language models, can be applied to Stable Diffusion [32]. These add a low rank set of weights alongside the original network, whilst the original model is unchanged. For GANs, hyper networks serve a similar purpose, in only modifying a small part of the network for synthesis with few samples [4].

## 5.1 Introduction

Recent years has seen substantial progress on improving image generation quality. Notably, the StyleGAN2 network substantially improved the state of the art for image generation and can fool even human

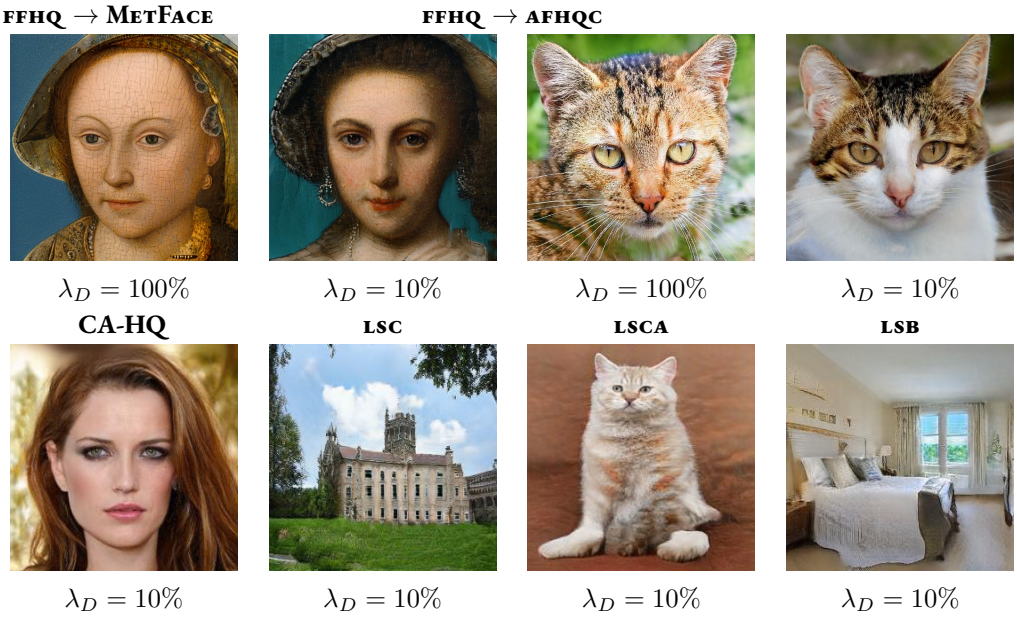



Figure 5.0.1: Near state of the art generation quality can be obtained on transfer learning with ten time less trainable parameters (top row). This is true even for very different datasets, such as FFHQ (human faces) transferred to cat faces AFHQC (top row). Our proposed width-wise sharing strategy outperforms other strategies, and spreads capacity of trainable parameters throughout the network. It can also be applied to learning several distributions simultaneously (bottom row).

observers at high resolutions [84]. Even so, StyleGAN2 is a resource intensive network with millions of parameters even if only for a single domain or with few images. As such, the research focus for image generation has shifted to more specialized tasks aiming at increasing the efficiency or usability of these generators. For instance several works have addressed transfer learning between datasets [175, 119, 80] or learning with limited data [80, 195]. Indeed, the *status quo* for GANs is to re-train from scratch millions of parameters for every new image dataset. Clearly this is a poor approach, as even disparate image classes can contain commonalities, such as color and texture distributions. Several methods have employed explicitly freezing discriminator layers when training a new generator on similar data [80, 119], which helps train on limited data by regularizing the learning problem and converging faster as less parameters need to be learned. Similarly, MineGAN [175] appends layers to a generator and freezes the original generator parameters.

In this work we address to what extent image features can transfer across datasets. Unlike previous work, we consider capacity needed to perform transfer learning and multi domain learning in terms of trainable parameters per domain. This is useful because after training, one only needs to store and distribute the domain specific parameters. For multi domain learning, one also has a shared representation which contains features from many distributions. As an additional result, we demonstrate intra layer redundancy of Stylegan2 parameters. In summary, we provide the following contributions:

- We demonstrate a parameter sharing method for GAN generators, which we call *width-wise* sharing, summarized in Fig. 5.1.1 and detailed in Sec. 5.3. Naive sharing strategies used in previous works reported in Sec. 5.2, such as sharing by layer, are unable to generate realistic samples with small parameter budgets. In Sec. 5.4, we show our sharing procedure can generate new images with ten times less trainable parameters without significant drop in image quality, as illustrated in Fig. 5.0.1.

- In Sec. 5.4.1, we show results for transfer learning in the low data regime. Notably, our sharing

method offers decent quality, whilst offering robustness to membership attacks, unlike full fine tuning.

- We train a model on multiple image generation benchmarks simultaneously. Our method outperforms the baseline multi domain image generation method of StarGAN-V2 in terms of FID (Eq. 2.1).

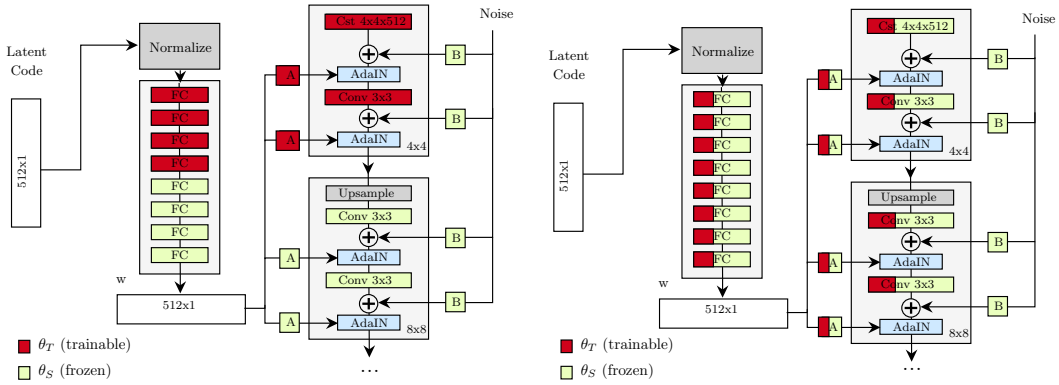



Figure 5.1.1: Sharing strategies illustrated on the StyleGAN network. Left: layerwise sharing, a fixed ratio of the style / convolutional layers are domain specific (red) and the rest are shared between models (green). Right: widthwise sharing, in each style and convolutional layer a fixed ratio of the parameters are specific (red) and the remainder are shared (green).

## 5.2 Previous Work

We start by reviewing several recent works related to transfer learning with GANs and parameter sharing in generative networks.

### 5.2.1 Generative Transfer Learning

Recently, transfer learning has been shown to be effective in the generative setting to address the problems of limited data training and slow training time [175, 80, 195]. The quality of state of the art GANs, such as StyleGAN2 or BigGAN [21], degrades significantly with "small" amounts of training data, which can even be as large as 1k-5k samples. This significantly limits the applicability of such methods in real world scenarios.

Various approaches have addressed this problem via transfer learning. In [119], several input layers of the *discriminator* are frozen and the generator is learned from scratch. In MineGAN [175], knowledge is transferred from a previously learned generator to a new domain by either freezing all layers of the generator and relearning a compact MLP layer appended to the input, or by finally allowing fine tuning after an initial learning step. In this work, we consider only the setting where the source generator parameters are frozen, as we consider parameter efficiency and finetuning corresponds to fully relearning the generator. MineGAN is attractive in that only a very compact set of parameters needs to be stored, alongside the original generator to generate new data. However, we will demonstrate that MineGAN is unable to tackle difficult transfer problems, when the new data distribution is far from the original distribution. Finally, we note the work of [196], which only retrains layers near the input of the source generator, and then only relearns filter statistics of subsequent frozen layers, which they dub "AdaFM," due to it's similarity to the AdaIN layers in Stylegan [84]. While this is in some ways similar to our layerwise strategy

(see next section), we do not consider this method directly as they re-learn a substantial portion of the generator parameters.

### 5.2.2 Learning GANs on Multiple Domains

Many problems seek to train GANs to generate data from many different domains. In conditional image generation, one seeks to generate potentially thousands of classes, each containing a small amount of samples [21, 190]. In BigGAN for example, class specific generation is controlled by an embedding layer at the input, and all other parameters are shared amongst every class. In domain translation, one seeks to translate samples from domain to another with a generative model, typically with a GAN training loss [64, 30, 197]. For instance in StarGAN-V2 [30], each image domain shares a common representation with one another, and each domain has specific style layers. Then samples from each domain can be translated simply by replacing the style. These methods have a similar objective to this work in that they promote models with parameter re-use across different domains. Finally, also related is the task of lifelong learning, wherein data is learned in an online fashion and may be subject to domain shift [180, 33].

In this work, we consider two settings where models share parameters to generate data from different domains. The first is transfer learning where most parameters are simply re-used as is (frozen) while the remaining ones are adapted to generate a new domain. We also consider a use-case wherein multiple domains are learned simultaneously with the majority of parameters shared. However, we believe our parameter sharing procedure, introduced in the next section, can be applied to many of the multi-domain settings introduced herein.

## 5.3 Sharing Strategies for Transfer Learning

We consider transferring knowledge from a GAN generator/discriminator pair $(G_1, D_1)$ trained on dataset $\mathcal{S}_1$ to a GAN $(G_2, D_2)$ trained on dataset $\mathcal{S}_2$ (denoting henceforth the transfer as $\mathcal{S}_1 \rightarrow \mathcal{S}_2$). We initialize training of $(G_2, D_2)$ with the parameters of $(G_1, D_1)$. Then, a subset of parameters are frozen to force the new generator to use knowledge from the old one. Shared (frozen) parameters are referred to as $\theta_S$ and domain (trainable) parameters are referred to as $\theta_D$. As ultimately we wish to train $G_2$ with as few parameters as possible, we measure the number of new parameters needed to train $G_2$ by simply measuring the ratio of trainable parameters, which we refer to as the learn ratio $\lambda_D$, i.e. $\lambda_D = |\theta_D|/(|\theta_D| + |\theta_S|)$. Note that $\lambda_D$ is also the ratio of memory needed to store $G_2$; if one has already stored $G_1$, $G_2$ can be constructed by loading $\theta_S$ from $G_1$. We detail several strategies for splitting capacity of $G_2$ amongst $\theta_D$ and $\theta_S$ given $\lambda_D$ in the following paragraphs.

**MineGAN** In MineGAN [175], MLP layers are appended to the input of the generator and all other parameters are frozen. Therefore the trainable parameters in this layer are $\theta_D$. Furthermore, $\theta_S = \theta$ is simply all original parameters frozen at the start of training. While in [175], the generator may also be fine tuned after the MLP layers are learned, we consider only the case where no finetuning takes place, as we are investigating the parameter efficiency of different sharing procedures. Thus, MineGAN will only refer to the learned MLP layers with every other layer frozen.

**Layer wise sharing** As a simple baseline, we explore simply choosing entire layers of StyleGAN2 to be within $\theta_D$ or not. We consider two configurations for $\lambda_D = 1\%$ and $\lambda_D$=10%. For $\lambda_D = 1\%$, we choose only the first two "style" MLP layers in the mapping network and for $\lambda_D = 10\%$ we choose the first four style layers and first convolutional block (see Fig. 5.1.1). We chose to favor early layers as they have more of a global influence on generation, however the choice of layers to share is not trivial and is part of the drawback of a layer wise approach. Note that several methods in the literature use the early layers of Stylegan2 as the domain specific layers [30, 194]. Note that both StarGAN-V2 [30] and [194] are layer wise procedures, which learn the MLP layers of Stylegan2 as the domain specific parameters and then

share the generator network. For our multi domain learning experiments, discussed in Sec. 4.5, we train StarGAN-V2 using the official code, and then only use the networks for generating random samples. Note as well that unconditional generation from each domain is one of the objectives for StarGAN-V2, so it serves as a decent comparison for our setting.

**Width wise sharing** In an effort to give a more flexible sharing strategy w.r.t. the parameter budget and also spread capacity through the various resolutions of the network, we propose splitting *each layer* into trainable and shared parameters. Note that in the related domain of continual learning for classification, existing models are augmented by appending new filters.

To do this, we split weight tensors in each layer according to $\lambda_D$, taking the first dimensions to be $\theta_D$ and the second part to be $\theta_S$. The style blocks in StyleGAN2 are based off adaptive instance normalization [63] with an affine layer taking as input style codes and applying a gain to convolution channels. The affine layer contains a $I \times I$ matrix, we take the first $\lfloor \lambda_D I \rfloor$ columns to be $\theta_D$. Similarly for convolution layers we take the first $\lfloor \lambda_D I \rfloor$ input filters to be $\theta_D$. In practice, these weight tensors are kept separate where $\theta_D$ and $\theta_S$ are concatenated on the forward pass. In this way, our method can be added in place to any generator, without changing output. See the rightmost diagram of Fig. 5.1.1 for more details.

### 5.3.1 Sharing for multi-domain learning

In the previous section, we discussed sharing in a transfer learning setting where parameters $\theta_S$ are merely frozen. In this work, we also explore a distributed setting where multiple domains $\mathcal{S}_1$,$\mathcal{S}_2$,$\mathcal{S}_3$,... are learned *simultaneously*. In this case $\theta_S$ is learned along with $\theta_D$, however $\theta_S$ is trained in a distributed fashion wherein gradients are synchronized at each optimizer update. Thus, the downstream distribution and storage of the network still depends on $\lambda_D$, as it pertains the number of unique parameters needed for each generator.

## 5.4 Experimental results

In this section, we view generation results for the various training strategies detailed in Sec. 5.3, namely MineGAN, layerwise sharing and finally our width-wise sharing method for the transfer learning problem.

We indicate by $\mathcal{S}_1 \rightarrow \mathcal{S}_2$ the transfer learning from a domain $\mathcal{S}_1$ to a new domain $\mathcal{S}_2$, where only a fraction $\lambda_D$ of the generative network parameters are retrained on the new domain. Various settings are tested for FFHQ $\rightarrow$ METFACE and FFHQ $\rightarrow$ LSC for different methods: MineGAN (where $\lambda_D$ is close to 1%), the naive layer-wise approach and the proposed width-wise sharing for $\lambda_D = 10\%$, and finally full training the generator ($\lambda_D = 100\%$). FID values for corresponding experiments are also reported in Table 5.4.1.

One could expect the transfer of FFHQ $\rightarrow$ METFACE to be easier than the transfer of FFHQ $\rightarrow$ LSC, because while the color and textures within METFACE are different from FFHQ, the datasets still comprises registered faces, and contains similar attributes to FFHQ. Still, the results show that MineGAN is unable to produce plausible samples for METFACE, while layerwise and width-wise training are both able to handle this transfer. On the other hand, FFHQ $\rightarrow$ LSC requires the generator to learn completely new large scale structures. In this case, while layerwise training can produce plausible backgrounds and colors, the objects (churches) are blurry and unrecognizable. Surprisingly, even with ten times less learnable parameters, the width-wise sharing method is able to handle this difficult transfer, both in terms of the visual quality of samples (see Fig. 5.0.1) and the FID values provided in Table 5.4.1.

**Precision Recall Results** Next, we compute the precision / recall values for Table 5.4.1 using the method underlined in Chapter 2. Specifically, we compute the Inception-V3 [164] features for each tar-

| | FFHQ→MetFace | | FFHQ→LSC | |
|---|---|---|---|---|
| strategy | $\lambda_D$ (%) | FID | $\lambda_D$ (%) | FID |
| fully trainable | *100* | *22.1* | *100* | *10.7* |
| MineGAN[1] | ≈1 | 59.5 | ≈1 | 254 |
| layer-wise | 10 | 50 | 10 | 191 |
| | 1 | 52.1 | 1 | 136 |
| width-wise | 10 | **27.8** | 10 | **22.5** |
| | 1 | **39.2** | 1 | **31.1** |

Table 5.4.1: FID values for different sharing strategies under various parameter budgets $\lambda_D$. Both MineGan [175] and layer-wise strategies fail for the difficult transfer of faces to churches FFHQ → LSC, as indicated by the high FID's over 100, contrary to the proposed width-wise approach. See the supplementary material for visual comparison.

get dataset $\mathcal{S}_\in$ and $5k$ generated samples. We use the same classifier architecture as in Sec. 2.6 and train with binary labels on specifying real and generated images.

Table 5.4.2 shows the PR calculations for the same training setup as Table 5.4.1. The results are consistent with the FID calculation, however, it appears that training with fewer parameters has a larger effect on recall than precision. We also observed this visually, as generators with low $\lambda_D$ still appeared to generate realistic images, but with low diversity. For instance, the colors for the MetFace generators would appear "washed out," and the more unique images (such as sculptures) appeared less often in generation. However, the images still appeared realistic in this highly constrained setting.

| | FFHQ→MetFace | | | FFHQ→LSC | | |
|---|---|---|---|---|---|---|
| strategy | $\lambda_D$ (%) | Drop P (%) | Drop in R (%) | $\lambda_D$ (%) | Drop P (%) | Drop in R (%) |
| fully trainable | 100 | 0 | 0 | 100 | 0 | 0 |
| MineGAN | ≈1 | 148.1 | 214.5 | ≈1 | 2470.0 | 4102.4 |
| layer-wise | 10 | 132.3 | 20.5 | 10 | 1307.8 | 1778.7 |
| | 1 | 112.7 | 141.1 | 10 | 1114.7 | 1501.6 |
| width-wise | 10 | **22.5** | **41.5** | 10 | **117.5** | **155.9** |
| | 1 | **45.7** | **82.2** | 1 | **214.9** | **291.7** |

Table 5.4.2: Drop in precision and recall values for different sharing strategies under various parameter budgets $\lambda_D$. As with Table 5.4.1, width wise out performs MineGAN and layerwise training for all settings. Interestingly, networks lose recall more than precision with less parameters. This usually corresponds to realistic, but not diverse synthesis.

**Additional Visual Results** sectionAdditional Visual Results First, we compute additional visual results for the 256×256 generators in Fig. 5.4.1. Each generator was trained with $\lambda_D = .1$ trainable parameters and using width wise sharing. Note that all four examples can generate plausible samples, despite transferring to a very different image class (registered faces versus outdoor scenes).

### 5.4.1 Limited Data Setting

In this section, we explore the challenging setting of learning with limited training samples. Learning with limited data is a known challenge for GANs [175]. Progress has been made in the recent literature with Stylegan2-ADA, which adds differentiable data augmentation to learning [80]. We employ Stylegan2-ADA training for the two transfers FFHQ→AFHQC and FFHQ→MetFace and results are shown in Table 5.4.3.

(a) FFHQ → LSC

(b) LST → LSC

(c) LSC→ FFHQ

(d) LST → FFHQ

Figure 5.4.1: Several cross domain transfers with $\lambda_D = 0.1$ and a resolution of 256×256. These samples correspond to the generators in Table 2.

**Privacy Gained from Parameter Sharing** Recall that in Chapter 4, we didn't train in the low data setting as the GAN generators would be unstable or produce very low quality images. However, memorization is especially important in low data settings, as intuitively it should be "easier" to memorize samples. In this section, we explore whether generators trained with StyleGAN2-ADA are susceptible to the membership inference attacks presented in Chapter 4.

We study several settings similar to those in Table 5.4.3. For each dataset, we choose to train on $N$ samples randomly chosen from the target dataset $\mathcal{S}_2$ and then holdout another $N$ samples for performing the membership attacks. We look at two transfer settings on faces, (1) FFHQ→METFACE and AFHQC→FFHQ. For METFACE, the dataset has only 1k samples, thus we study $N = 100, 500$ and for FFHQ, $N = 100, 500, 1000$. We train for 100k batches in all settings, wherein we saw the FID saturate (i.e. it would not continue to improve), but where membership attack inference would potentially increase. Table 5.4.4 shows results for several values of $\lambda_D$. For METFACE, training with $\lambda_D = 1\%$ offers near perfect defense versus both membership attacks presented in Chapter 4. Furthermore, FID is still decent for this setting, and images appear high quality. In general, as less images are used for training,

| | FFHQ → AFHQC (5k) | FFHQ → METFACE (1k) |
|---|---|---|
| $\lambda_D$ = 100% | *4.7* | *17.1* |
| $\lambda_D$ = 20% | 5.2 | 17.9 |
| $\lambda_D$ = 10% | 5.9 | 18.6 |
| $\lambda_D$ = 5% | 7.1 | 20.2 |
| $\lambda_D$ = 1% | 14.4 | 22.7 |

Table 5.4.3: FIDs for limited data transfers for several parameter budgets $\lambda_D$ using ADA augmentation [80].

the generator becomes more vulnerable to membership attacks and improving resistance to the attacks as $\lambda_D$ increases. For FFHQ, we noticed sample quality dropped significantly once $N < 1000$ samples, even in the fully trainable case. However, for $N = 1000$, we saw decent sample quality and resistance to membership attacks in the $\lambda_D = 1\%$ setting.

| $\mathcal{S}_1$ | $\mathcal{S}_2$ | $N$ | $\lambda_D$ (%) | LOGAN (%) | Latent Recov. (%) |
|---|---|---|---|---|---|
| FFHQ | METFACE | 100 | 100 | 98 | 85 |
| FFHQ | METFACE | 500 | 100 | 86 | 71 |
| FFHQ | METFACE | 100 | 10 | 87 | 72 |
| FFHQ | METFACE | 500 | 10 | 84 | 54 |
| FFHQ | METFACE | 100 | 1 | 55 | 51 |
| FFHQ | METFACE | 500 | 1 | 51 | 50 |
| AFHQC | FFHQ | 100 | 100 | 99 | 64 |
| AFHQC | FFHQ | 500 | 100 | 99 | 91 |
| AFHQC | FFHQ | 1000 | 100 | 98 | 86 |
| AFHQC | FFHQ | 100 | 1 | 64 | 55 |
| AFHQC | FFHQ | 500 | 1 | 53 | 51 |
| AFHQC | FFHQ | 1000 | 1 | 52 | 53 |

Table 5.4.4: Vulnerability to membership attacks for the low data training setting. Here $N$ is the size of the target dataset. The right two columns are accuracies for membership attacks performing LOGAN [57] and a latent recovery based attack (see Chapter 4). Training with $\lambda_D = 1\%$ seems to offer defense against both attacks, with 10% offering defense against recovery attacks.

### 5.4.2 Sharing strategies for Multi-domain distributed learning

In the previous section, we explored transfer learning from a single source domain to a target domain and froze some source generator parameters to enforce parameter sharing. In this section, we'll explore learning multiple domains simultaneously. In some generative applications, parameter sharing is built in to the learning procedure. For example, in domain translation, a shared representation is used to translate samples from one domain to another [30, 63, 64]. Here, we'll compare to the state of the art in image translation network StarGAN-V2 [30] and only use the final network for unconditional image generation. As was discussed in Sec 5.4, StarGAN-V2 takes $\theta_D$ to be the MLP mapping layers in the Stylegan2 network, and $\theta_S$ to be all other parameters, which is equivalent to a layerwise strategy. Four our sharing strategies, we'll learn several domains $\mathcal{S}_1, \mathcal{S}_2, ...\mathcal{S}_n$ whilst learning both $\theta_S$ (shared amongst every domain) along with $\theta_D$. Note that in this setting, the overall memory needed to store all parameters is $n|\theta_D| + |\theta_S|$, and thus $\lambda_D$ is an important parameter effecting the bandwidth needed for parameter updates (if training is done in a distributed parameter-server paradigm) and distribution of the network after training.

Table 5.4.5 compares FID scores for StarGAN-V2, layerwise and width-wise sharing strategies when training on 4 datasets (CA-HQ, LSC, LSCA, LSB). In the case of $\lambda_D = 1\%$, width-wise sharing beats layerwise by a large margin for every dataset. When $\lambda_D = 10\%$, the two sharing methods are more comparable. This is somewhat to be expected; a particular choice of layers for layerwise sharing might perform better for a specific set of datasets, as style layers each control different image semantics and particular layers may represent commonalities better than others (see Sec. 5.3). On the other hand, when $\lambda_D$ is too small, the choice of layers is limited and we chose input layers as it reflects the sharing in BigGAN or MineGAN. StarGAN-V2 performs comparably on 3 datasets and has generator collapse on CelebA-HQ. This is likely because it is the most different of the three datasets, and the extra constrain of domain translation has forced the shared representation to favor similar domains. Width-wise is clearly a better approach in general as it spreads domain specific capacity throughout the network. We note that good performance with low $\lambda_D$ desirable because as with the previous setting it effects the downstream storage and distribution of parameters.

| Strategy | Layer-Wise | | StarGAN-V2 | Width-Wise | |
|---|---|---|---|---|---|
| $\lambda_D$ | 1 | 10 | 10 | 1 | 10 |
| CA-HQ | 23.71 | 18.84 | 254.7 | **17.06** | **15.08** |
| LSC | 17.29 | 15.49 | 21.31 | **14.45** | **14.46** |
| LSCA | 39.40 | 43.43 | 47.24 | **36.44** | **32.05** |
| LSB | 19.14 | 18.67 | 19.15 | **17.29** | **13.56** |

Table 5.4.5: FID values for different sharing strategies under various parameter budgets $\lambda_D$ for multi-domain distributed learning (here 4 datasets simultaneously). Note that width-wise has the highest quality samples for each domain, across both parameter budgets. See the supplementary material for visual comparison.

## 5.5 Conclusion

In this chapter, we addressed the problems of transfer learning and multi domain learning with a GAN generator. We proposed the simple yet effective weight sharing method of width wise sharing. This method was more efficient at transfer learning than currently the currently proposed method of MineGAN when constrained to a set parameter budget. We saw our method was effective at learning in the low data transfer learning setting, whilst only learning a very small portion of parameters. In addition, generators trained without parameter sharing were highly vulnerable to membership attacks. Training with weight sharing offered robustness to these attacks; in light of Chapter 4, it would be possible to generate data surrogates that are empirically robust to several membership attacks. Finally, we used our width wise sharing method to learn multiple domains simultaneously, and again showed that our method was more effective than a state of the art method StarGAN-V2, when simultaneously learning several challenging datasets.

Chapter 6

# Identity Membership Attacks

Thus far, we have exposed several membership attacks designed to discern individual training samples given model outputs. These attacks are, in general, more successful when less training data is used. In this chapter, we will see how more global information about the training data can leak into generation. Namely, we will explore how identities of face images can be detected in generated samples. Unlike in the previous chapters, we will explore how dataset bias, rather than just dataset size, creates vulnerability to membership inference attacks. At the time this work was conducted, generative adversarial networks (GANs) were just beginning to achieve realism that fools even human observers. Indeed, the viral website "this person does not exist" suggested that generated images were entirely novel creations. On the other hand, GANs can leak information about their training data, as evidenced by membership attacks demonstrated in this thesis. In this chapter, we explore how GANs can leak information about training data that lacks diversity, by demonstrating a new membership attack. Unlike previous works, our attack can accurately discern identities of facial images, without having access to any of the dataset samples. Furthermore, our attack is stronger than previous attacks which requires the GAN discriminator, as we only require generated images. We demonstrate the interest of our attack across several popular face datasets and GAN training procedures. Notably, we show that even in the presence of significant dataset diversity, an over-represented person can pose a privacy concern.

**Continued Work** Whilst the study of identity membership attacks was introduced in this work, several works have since studied a similar problem. In [60] identity membership attacks is studied again in the context of multi-modal image data, namely the CLIP network [138]. As CLIP can take text input, the membership attack in this work can simply utilize any name, and then construct an attack based on model outputs.

## 6.1 Introduction

StyleGAN has shown state of the art results for several face generation benchmarks. Together with the benefits of GAN genreation (such as single pass generation), they are still widely used today in favor of diffusion models. They also have a wide set of rich applications such as age modification, inpainting, super-resolution, or other attribute modification [91, 187, 177, 29]. Less studied lines of research include measuring the statistical consistency between the generated distribution and the real one [59, 147, 158], or in theoretical analysis of the GAN learning problem [10].

GANs have made incredible progress in terms of visual quality, as measured by the popular Frechet Inception Distance (FID) 2.1, or merely by sample inspection (for example, see StyleGAN samples in Fig. 6.1.1). However, it remains unclear in what ways GANs generalize. Several works have noted that the GAN *discriminator* tends to overfit the training set [8, 21], in the sense that it will label hold out test images as fake. Such an observation was used to exploit the privacy of GANs in the LOGAN approach

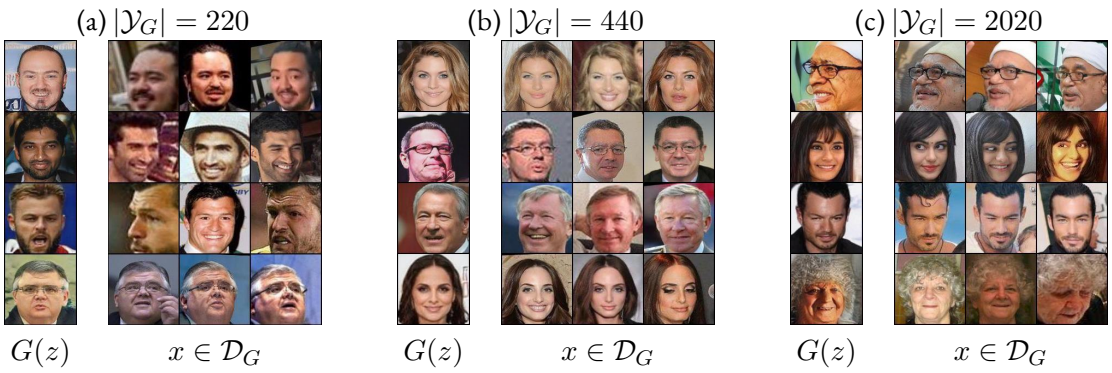



Figure 6.1.1: Extracting identities from GAN images. Each row displays a GAN generated image (left) with three training images (right) having the same predicted identity. Images are generated with StyleGAN [83] using $N$ training images (respectively 40k, 80k and 46k) from VGGFace2 and identified with a face identification network. We investigate two different scenarios in this paper: in (a) and (b) identities are evenly distributed over the datasets, where in (c) a small subset is more represented. While some samples merely bear resemblance, other generated images strongly share idiosyncratic features of training identities. Such a nearest neighbor search helps factor out the ways in which GANs can generalize (via pose, lighting and expression) and elucidate overfitting on identities. This is a threat to privacy as we demonstrate in our blind identity membership attack.

[56]. Nonetheless, the LOGAN leak challenges mainly the discriminator, leaving the case of the generator in a somewhat gray area. In fact, several heuristics exist to show GANs do in fact produce novel data, whether it be by sample interpolation and attribute modification as demonstrated in the first GAN works [52, 139], or by the ability of GANs to generalize to novel pose and expression [153]. Furthermore, a general belief is that the generated images are always entirely original (in terms of identity), as can be testified by the popular website https://thispersondoesnotexist.com/.

On the other hand, to our knowledge no metric can properly answer this question and the purpose of this work is to measure to what extent generated identities resemble those in the training set. Alongside the generalization aspect, this question is fundamental to evaluate the potential privacy risk posed by GANs, for instance concerning those appearing in the above mentioned website. There are several reasons why this problem is important and inadequately solved by existing literature. First, recent attempts to apply differential privacy mechanisms on GANs [182, 76] have not been so successful. Indeed, to our knowledge, published works lead to an unsatisfactory tradeoff between the privacy guarantees and the quality of the learnt distribution, in particular generating far from plausible images. Besides, following the work of [177] showing some form of GAN generalization, some attempts were made to leverage GANs to produce ersatz datasets aiming at more privacy. For instance in Chapter 4 a dataset which is kept private is used to train a GAN. Then the GAN generated images are then made publicly available for downstream tasks. Again, the rationale behind such a strategy is that a synthetic dataset produced from the GAN can provides the means to achieve the downstream task allegedly without compromising the privacy of the original dataset.

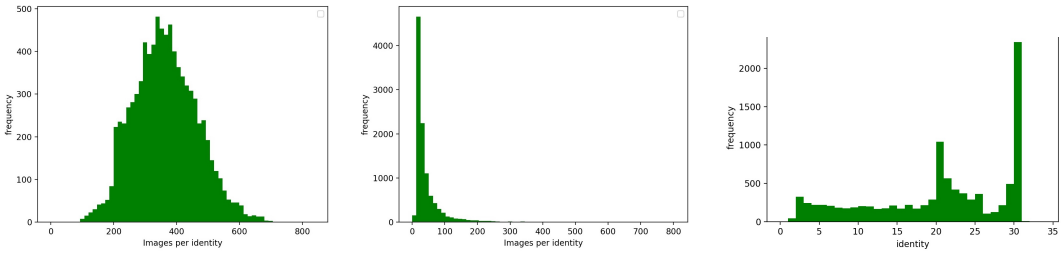
In addition, neural networks leak information in numerous ways about the data on which they are trained [162, 25]. When no privacy mechanism is used, the most common way to expose potential privacy leaks from a neural network consists in applying one of several off-the-shelf attacks. Amongst the most well known attacks are the *membership inference attacks* [156]. In this type of attack, an attacker tries to discern which samples were used during model training. In [156], training set samples could be nearly perfectly determined from model outputs on the MNIST classification task. Such an attack typically utilizes statistics of the model output. For example, a model that overfits will have low values of the loss

function on the training samples, thus allowing for a simple thresholding attack. Similar attacks can be seen in [109, 48, 110]. More recently, membership attacks have been devised against generative models, such as LOGAN, leveraging the GAN discriminator [97, 56]. These attacks are slightly more intuitive, in the sense that generated samples live in the same space as the data, and they may be visually similar or identical to exact training samples. In [177], the non-adversarial procedure of GLO [18] reproduces some training samples verbatim, while not being able to reproduce test samples coming from the same distribution. Such direct leaks were not observed for typical GANs. However, we argue that generated images can resemble training samples in even more subtle ways, as seen in Fig. 6.1.1: generated faces can highly resemble exact identities, even if the images themselves are quite different.

**Identity Membership Attacks** The main contribution of this paper is to define a refined attack objective and implement such an attack. This new objective is meaningful in situation where a clear notion of identity exists. This is the case for example for datasets of face images (the central case considered in here). But such a notion make sense in many other contexts such as dataset of paintings (think of the identity of the painter), bio-metric data, or medical images. In our scenario, the attacker would have access to a collection of query samples, but instead of trying to discern samples that were truly part of the training set of a GAN, the attacker should rather determine if samples with the same identity were used. An attack of this sort shall be referred henceforth as an *identity membership attack*. In addition to this fundamental difference with currently published attacks, we shall design an attack with the following properties:

- contrary to LOGAN [56], the attacker can only exploit the generator and not the discriminator,
- the attacker should not be aware of the ratio of query identities that were part of the training set (i.e, a *blind* attack).

We hold that if successful, such an attack would reveal as a serious hurdle for the safe exchange of GANs in sensitive contexts. For instance, in the context of paintings or other art pieces, distributing a non-private generator might well be ruled-out for obvious copyright issues. More importantly, consider a biometric company A releasing a generator exposing its consumer identity. Another company B could potentially detect which of their own consumers are also clients of company A. Similar situations can pose serious issues for medical data, where revealing a GAN could breach personal information about a patient disease.




VGGFace2 CASIA-WebFace CelebA

Figure 6.1.2: **Histogram of number of instances per identity for different datasets (VGGFace2, CASIA-WebFace & CelebA).** *Far from being uniform, face recognition datasets include bias in terms of the number of examples per identitiy. VGGFace2 was designed to be balanced in this sense, compared here to the CASIA-WebFace containing bias in some samples.*

**Face Generation** In this work, we focus on GANs trained on facial image data. Face image datasets are amongst the most popular for demonstrating GAN efficacy, partly because the human eye is particularly

attuned to detect artifacts in these images and due to the wide availability of high quality face datasets [85, 23, 99]. Face datasets include some form of bias in regard to the real distributions of faces. For instance, datasets are frequently composed of a decent number of instances of the same person (see Fig. 6.1.2). Besides in some datasets [174], the number of instances per identity is highly versatile (e.g. in MEGAFACE [85] it ranges from 3 to 2k). One should also note that datasets dedicated to generation such as FFHQ [83] may well display similar biases due to the lack of any specific safeguard.

**Outline** The rest of the paper is organized as follows. In Section 6.2 we expose the proposed protocol used to train the face detection network, to perform the attack and to evaluate its performance. This protocol is tested in Section 6.3 on various datasets (CASIA-WebFace and VGGFace2) with different GANs (StyleGAN and LSGAN) across different attack scenarios. Performance curves and visual evidence are provided to show membership attacks can be highly successful against GANs even in the blind scenario, and when diverse training data is used. A discussion and perspectives on future work are given in Section 6.4.

## 6.2 Identity Membership Attack

As argued earlier, because standard datasets gather several instances of some individuals, training generative models on such datasets exposes those individuals to privacy leaks. Figure 6.1.1 illustrates such a leak for a GAN trained on a subset of VGGFace2. There is no doubt that in many cases, the displayed generative sample is but an instance of the training identity shown under various poses (last three columns). As such the generated samples can be leveraged to extract information on the identities that were seen during the training of the GAN.

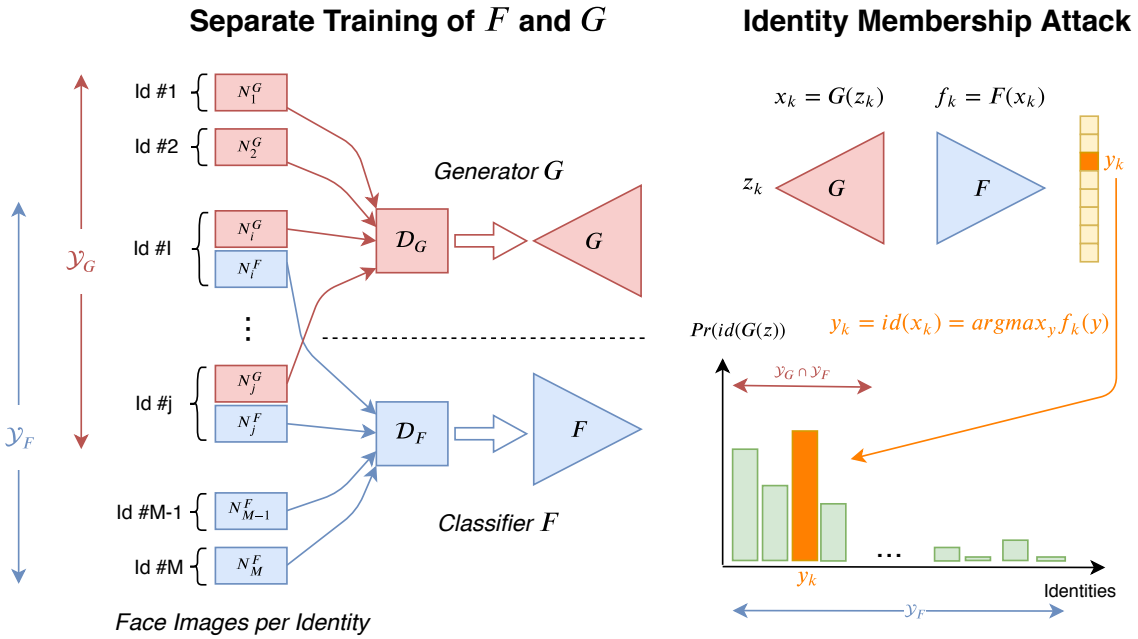



Figure 6.2.1: Illustration of the protocol used for the attack. A generator $G$ is trained from a dataset $\mathcal{D}_G$ gathering image instances of identities $y \in \mathcal{Y}_G$ (red samples). A face classifier $F$ is trained by an attacker to recognize identities $y \in \mathcal{Y}_F$ from a separate dataset $\mathcal{D}_F$ (blue samples). Although the samples are completely separate, the two datasets share some common identities $\mathcal{Y}_G \cap \mathcal{Y}_F \neq \emptyset$. Samples $x_k$ are generated by $G$, are fed to $F$ to infer identities used for training.

### 6.2.1 Attack Assumptions

We consider the following attack scenario : the attacker wants to determine if instances of certain query individuals were used in the training of the GAN. We reference this type of attack as an *identity membership* attack. More formally, our attack protocol is illustrated in Fig. 6.2.1 and delineated as follows:

- A GAN $G$ is trained on a dataset $\mathcal{D}_G := \{(x_i^G, y_i^G)\}$ where $\forall i, y_i^G \in \mathcal{Y}_G$;
- The attacker trains a face identification network, $F$ on a dataset $\mathcal{D}_F = \{(x_i^F, y_i^F)\}$ where $\forall i, y_i^F \in \mathcal{Y}_F$ to recognize instances from identities $y \in \mathcal{Y}_F$;
- It is assumed that $\mathcal{Y}_G \cap \mathcal{Y}_F \neq \emptyset$[1] but $\mathcal{D}_F \cap \mathcal{D}_G = \emptyset$ . The first assumption reflects the fact that the attacker has some founded suspicion that some identities were used in the training of $G$. Nonetheless the second assumption ensures that no instance of $D_G$ is explicitely known to the attacker.

### 6.2.2 Attack Algorithm

The basic principle of the attack consists in randomly generating faces $x_k := G(z_k)$ for $k \in \{1, \dots, N\}$. Then using the network $F$, these random samples are identified to identities $y_k := id(x_k)$ where $id(x) := \arg\max_{y \in \mathcal{Y}_F} F_y(x)$. Eventually, the attacker will suspect the identities of $\mathcal{Y}_F$ that are more often predicted as the ones that were seen during the training of $G$. This mechanism is described in more details in Alg. 3 and merely corresponds to thresholding the number of times the query identity $y$ was predicted by $F$ on the generated samples $x_k = G(z_k)$.

To be effective, such an algorithm requires to generate a large enough set of samples $K$. Typically this parameter shall be set to many times the number of identities in $\mathcal{D}_F$ i.e. $K = \lambda \times |\mathcal{Y}_F|$. In such case, it is natural to fix the frequency threshold $T$ to $\lambda$. We will denote this natural value of the threshold $T_0 = \lambda$. Yet, of course, one can trade recall for precision by increasing this threshold. In our experiments, we shall consider also $T_1 = 10\lambda$.

**Inputs:** the query identity $y \in \mathcal{Y}_F$,
the number of generated samples $K$,
the frequency threshold $T$
**Output:** a boolean prediction of $\mathbb{1}_{y \in \mathcal{Y}_G}$
**Algorithm** `identityAttack(`$y$`)`

```
k_y = 0
for k = 0 to K do
    z = randomSample()
    x = G(z)
    if id(x) = y then
        k_y += 1
    end
end
return 1_{k_y ≥ T}
```

**Algorithm 3:** *Proposed identity membership attack*

### 6.2.3 Classifier Training

To train the classifier $F$, we use pre-trained features from the VGGFace network [133]. Note that VGGFace and VGGFace2 share 53 identities. Therefore, to ensure that $\mathcal{D}_F \cap \mathcal{D}_G = \emptyset$, *i.e.* $F$ is not trained

[1] In practice, we shall consider a worse case situation where $\mathcal{Y}_G \subsetneq \mathcal{Y}_F$.

on any of the sample samples as $G$, we remove these 53 identities from the VGGFace2 dataset. Then, we pool the relu5_3 layer of VGGFace to a $2 \times 2$ spatial resolution. Finally, we train a single fully connected layer with $|\mathcal{Y}_F|$ output classes. Note that $|\mathcal{Y}_F| = 8631$ for VGGFace2 and $|\mathcal{Y}_F| = 1292$ for CASIA-Webface. Finally, models are trained in pytorch with SGD, a learning rate of .1, and a momentum .9. This simple model for $F$ achieves an average top-1 classification accuracy of 86.1% on VGGFace2 and 94.7% on CASIA-Webface. Likely, using even better classifiers should improve the membership attack accuracy, but we leave this direction for future work.

### 6.2.4 Attack Evaluation

The attack can be seen as a binary classification problem where the attacker must classify identities from $\mathcal{Y}_F$ according to whether they also belong to $\mathcal{Y}_G$ or not. It is therefore natural to evaluate the performance of Alg. 3 in terms of precision and recall. Taking the attacker point of view, we will tag as positive all the identities in $\mathcal{Y}_G \cap \mathcal{Y}_F$ and the remaining ones as negative. Therefore, denoting $C(y) := \mathbb{1}_{k_y \geq T}$ the decision made by the attacker, the precision and recall will be computed as follows:

$$\alpha := \frac{\sum_{y \in \mathcal{Y}_G \cap \mathcal{Y}_F} C(y)}{\sum_{y \in \mathcal{Y}_F} C(y)} \quad \text{and} \quad \beta := \frac{\sum_{y \in \mathcal{Y}_G \cap \mathcal{Y}_F} C(y)}{|\mathcal{Y}_G \cap \mathcal{Y}_F|} \tag{6.1}$$

As the goal of privacy is to protect all individuals equally, an attack with high precision (and positive be it small recall) should be considered more troublesome than one with high recall. In other words, if an attack can accurately discern training information even on a small subset of data, it still violates the fundamental goal of privacy. In general, we'll explore how precise membership attacks can be for fixed values of recall. Then, we'll choose a threshold that works well in practice using the computed PR curves.

### 6.2.5 Visual Evaluation

As a visual sanity check, we include a nearest neighbor search utilizing the recognition network $F$. An image or feature space nearest neighbor search is a common procedure in the GAN literature, to dispel suspicions of overfitting [52, 21, 83]. We visually compare generated samples falling inside the GAN training set $y \in \mathcal{Y}_G$, and those detected to lie outside $y \notin \mathcal{Y}_G$. More formally, we generate an image $x := G(z)$, find the identity using $F$ as $y := id(x)$ and finally perform the intra-identity nearest neighbor search as

$$x_{\text{NN}} = \arg\min_{x_{y_i}} \|f(x_{y_i}) - f(x)\|_2^2 \tag{6.2}$$

where $x_{\text{NN}}$ is the retrieved nearest neighbor, $f$ is the before softmax feature representation of $F$ and finally $x_{y_i}$ represents the i-th image among the instances of identity $y$. Results are displayed for three attack scenarios in Fig. 6.1.1. Indeed, in both the scenarios of low diversity and dataset bias, generated images highly resemble those in the training set. Intuitively, these also correspond with attack scenarios that are highly accurate at guessing training identities (see Table 6.3.1 and Table 6.3.3). Note that this visual demonstration should be robust to any distortions or variations found in generated images, as $F$ is specifically designed to be robust to natural variations in the dataset like pose or expression.

## 6.3 Membership Attack Results

### 6.3.1 Experimental Protocol

We evaluate the efficacy of our identity membership attack on a variety of training settings, in order to elucidate what factors in training data influence privacy. As is shown in Figure 6.1.2, face datasets exhibit varying number of identities, and importantly bias in the number of samples they have per identity. VGGFace2 contains a relatively balanced number of samples per identity, while VGGFace2 has far

more average samples per identity than CelebA for example. On the other hand, in CASIA-WebFace the number of samples per identity ranges from a few to many. In the following training settings, we denote *diversity* as number of training identities and *(imbalance) bias* when some identities have many more samples than others. Unless special dataset augmentation is done, GAN generators typically require many samples from a distribution to train effectively [81]. Together with our disjointness assumption for blind attacks (see Sec. 6.2.1), we also curate CASIA-WebFace to contain at least 80 samples per identity so that the generators have sufficient training data. We then explore two real world training scenarios in the following sections. Throughout experiments, the parameter $\lambda$ discussed in Sec. 6.2.2 used to sample the Generator during attack is fixed to $\lambda = 2$.

### 6.3.2 Setting 1: Low bias and varying diversity

In this setting, we simply vary the number of identities $|\mathcal{Y}_G|$ ranging from 30 to 880, taking a large set of samples from the the first $|\mathcal{Y}_F|$ identities. The number of samples per identity $N_i^G$ used during the training of the generator is fixed to simulate datasets that are evenly distributed. In addition, we consider two successful GAN methods, the Least Squares GAN (LSGAN) [106] and the state of the art StyleGAN [83] network.

**Adaptation of LOGAN** We adapt the method of LOGAN [56] to this setting. LOGAN was originally used for membership inference against individual training samples. LOGAN uses the discriminator outputs to classify whether or not samples came from the training set. As in [156], this output can then be sorted and a threshold can be selected depending on what percentage of the training set is suspected to be present in the attacker's training set. Of course, our method does not assume the discriminator decision is available; this is a far more realistic scenario as the discriminator is normally discarded and only the generator is used for the majority of downstream applications.

We take an unbiased sample from the attacker's set of images $x \sim \mathcal{D}_F$. Then we take the discriminator responses $D(x), x \in \mathcal{D}_F$ to make our decision. The precision recall are then computed in the same way, except that we use $\mathbb{1}_{D(x)\geq T}$ as a positive inclusion in the training set (rather than the frequency of detection as in our method).

Table 6.3.1 gives the precision and recall rate for each setting, on two different datasets (VGGFace2 and CASIA-WebFace ). The total number of samples used during training is indicated by $N = \sum_i N_i^G$, and is increasing with the number of identities. Each column corresponds to a different generation training set, with a varying number of identities while the total number of identities is kept fixed ($|\mathcal{Y}_F| \approx 8$K). Precisions are computed for several values of recall to compare how training set size effects vulnerability to attack.The baseline corresponding to random face identification has a precision of $\frac{|\mathcal{Y}_G|}{|\mathcal{Y}_F|}$ at any recall. Finally, after demonstrating that attacks can indeed be successful, we choose a simple threshold that works reasonably well in practice and observe it's efficacy. Figure 6.3.1 displays Precision-Recall curves obtained for some settings when spanning the frequency threshold.

### 6.3.3 Setting 2: Varying bias and high diversity

We also examine scenario when training data is highly diverse, yet due to natural bias in the data collection procedure or data availability, certain identities contain many more images than others. For this setting, we vary the number of biased identities but enrich each dataset with a large set of diverse unbiased faces. More formally, we take $\mathcal{Y}_G := \mathcal{Y}_{G_1} \cup \mathcal{Y}_{G_2}$, with $\mathcal{Y}_{G_1}$ containing many more samples per identity than $\mathcal{Y}_{G_2}$. Furthermore, we consider when $|\mathcal{Y}_{G_2}| \gg |\mathcal{Y}_{G_1}|$, *i.e.* the dataset has high class diversity. Such is naturally the case in CASIA-WebFace , see Fig. 6.1.2. One may however suspect that the GAN be prone to reproduce mainly identities from $\mathcal{Y}_{G_1}$ (since they are more present in the dataset). We therefore only compute the precision/recall in Eq. (6.1) for the biased set $\mathcal{Y}_{G_1}$ in lieu of $\mathcal{Y}_G$.

Table 6.3.3 gives the precision and recall rates of the attack in such setting on VGGFace2 dataset. Attack are performed on LSGAN and StyleGAN trained with an increasing number of identities for subset $\mathcal{Y}_{G_1}$, while the second set $\mathcal{Y}_{G_2}$ is fixed to 40,000 samples evenly distributed across 2000 identities. The bias is therefore decreasing, as measured by the ratio $\frac{|\mathcal{Y}_{G_1}|}{|\mathcal{Y}_F|}$ corresponding to the precision of random identification, as reported in the Table. The total number of samples used during training is indicated by $N = N_1 + N_2$, and is increasing with the number of identities. Figure 6.3.2 displays Precision-Recall curves obtained for some settings and highlights several thresholds that work well in practice.

| Method | $\lvert\mathcal{Y}_G\rvert$ ($N$) | 30 (10k) | 58 (20k) | 111 (40k) | 220 (80k) | 440 (160k) | 880 (320k) |
|---|---|---|---|---|---|---|---|
| Random | | 0.35 | 0.67 | 1.29 | 2.55 | 5.10 | 10.2 |
| Ours | Recall = 10% | 100.0 | 100.0 | 91.7 | 51.2 | 8.5 | 14.1 |
| | Recall = 50% | 83.3 | 76.3 | 8.0 | 10.9 | 6.4 | 10.6 |
| LOGAN | Recall = 10% | 100.0 | 100.0 | 35.1 | 12.4 | 6.3 | 11.4 |
| | Recall = 50% | 100.0 | 95.1 | 7.2 | 3.7 | 5.3 | 10.3 |

Table 6.3.1: Setting 6.3.2 on VGGFace2 with the StyleGAN network. Attacks are a random guess, ours and LOGAN. Generators are vulnerable to both attacks when too few identities are present and appear to be robust against attacks with 880 identities. Notice our method is comparable to LOGAN[56], despite not needing the discriminator, and performs significantly better in the low recall regime.

| Method | $\lvert\mathcal{Y}_G\rvert$ ($N$) | 30 (4.5k) | 58 (8.7k) | 111 (16.65k) | 220 (33k) |
|---|---|---|---|---|---|
| | Random | 0.35 | 0.67 | 1.29 | 2.55 |
| Ours | Recall = 10% | 100.0 | 71.4 | 52.4 | 30.7 |
| | Recall = 50% | 14.6 | 23.4 | 26.0 | 25.3 |
| LOGAN | Recall = 10% | 100.0 | 98.3 | 49.4 | 25.0 |
| | Recall = 50% | 100.0 | 90.3 | 24.4 | 19.3 |

Table 6.3.2: Setting 6.3.2 on CASIA-WebFace with the StyleGAN network. Both attacks are more successful on this dataset, likely due to the fact there are less sampler per identity. Again, our method out performs in the low recall regime.

### 6.3.4 Analysis

In the first setting, shown in Table 6.3.1 for VGGFAce2 and Table 6.3.2 for CASIA-WebFace , we display the precision of our attack versus LOGAN and a random baseline. With too little diversity, StyleGAN is susceptible to membership attacks versus both methods. With enough diversity (here, towards 880 identities), attacks are reduced to near guessing. We also note that despite our attack using less information than LOGAN, it still outperforms in the low recall setting (for instance, for 111 identities used).

The first column contains only 10k images, compared to modern datasets containing hundreds of thousands or millions of images. Thus the first setting only demonstrates vulnerability when the absolute training set size is small. Thus, in the second setting we also enriched the datasets with a large and diverse set of faces containing 2000 identities, with some identities being over represented in terms of samples per identity. Table 6.3.3 show that diversity itself is not enough to protect the data in the presence of bias.

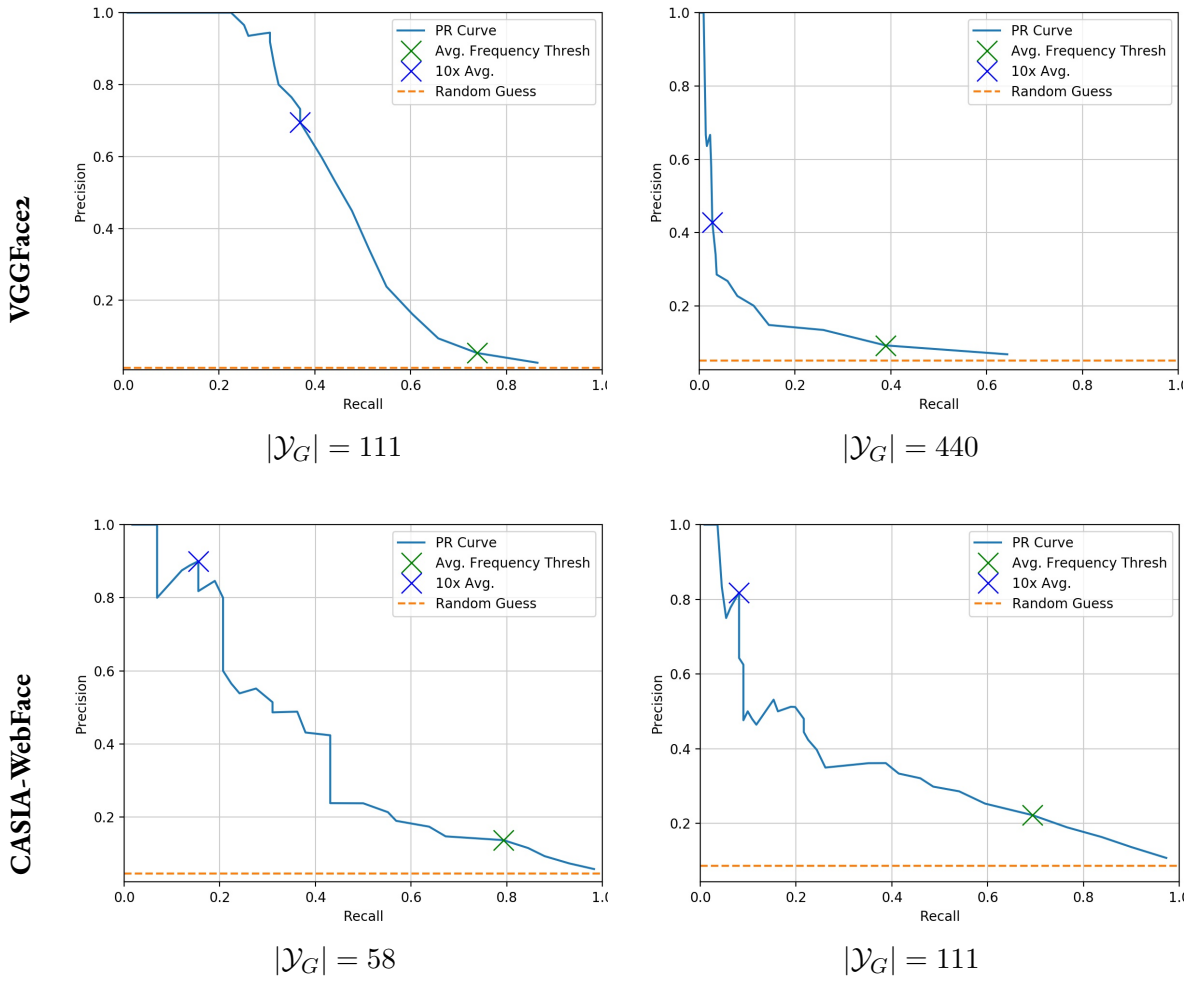



Figure 6.3.1: **Precision-recall curves for identity membership attacks for setting 1**. *Curves are obtained using Eq.*(6.1)*, the Style-GAN generator, two datasets and varying diversity* $\mathcal{Y}_G$*. Models trained on more diverse data appear to be more private. In addition, the threshold* $T_1 = 10T_0$ *can discern training identities with a high precision.*

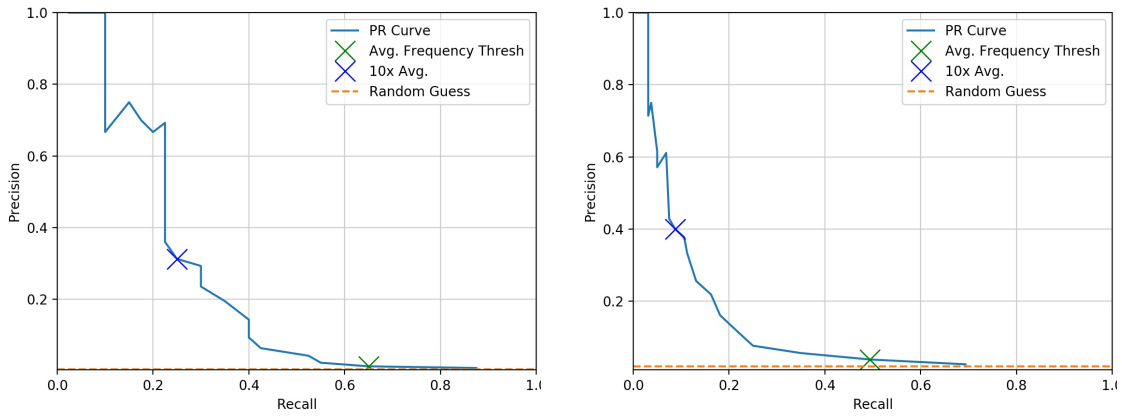
Even after taking more than 2000 identities, several scenarios show attacks that still have relatively high precision. Adding additional data does seem to provide some level of protection however. Consider the 220 and 160 identity training sets for setting 1 and 2 respectively, each having roughly 80k datapoints; the attack was successful in setting 1 with a precision of 51% and only 6.5% in the latter.

### Threshold selection

While Table 6.3.1 and Table 6.3.3 show attacks can be successful, a blind attacker needs to select a once and for all threshold for making his decision. Using values of $T_0 = \lambda, T_1 = 10\lambda$ (See Sec. 6.2.2) yields successful attacks, as indicated in the PR curves in Fig. 6.3.2, Fig. 6.3.1. Choosing only identities which are very frequent with $T_1$ yields a highly precise attack, albeit only some individuals are effected; thus if privacy is unilaterally important, such an attack would pose a risk.

| | $\lvert\mathcal{Y}_{G_1}\rvert$ ($N_1$) | 20 (6k) | 40 (12k) | 80 (24k) | 160 (48k) |
|---|---|---|---|---|---|
| | $\lvert\mathcal{Y}_{G_2}\rvert$ ($N_2$) | 2000 (40k) | | | |
| | Random | 0.23 | 0.46 | 0.93 | 1.85 |
| StyleGAN | Recall = 10% | 13.3 | 66.7 | 100 | 6.5 |
| | Recall = 50% | 2.0 | 4.1 | 7.6 | 3.4 |
| LSGAN | Recall = 10% | 70.4 | 61.1 | 57.1 | 3.1 |
| | Recall = 50% | 40.3 | 28.4 | 3.8 | 2.5 |

Table 6.3.3: Precision / Recall (in %) for the Membership Attack (Alg. 3) on GANs trained on the VGGFace2 dataset in the second setting. The dataset $\mathcal{Y}_G$ is now composed of two sets of distinct identities: $\mathcal{Y}_{G_1} \cup \mathcal{Y}_{G_2}$. GANs are trained on a large number of identities $|\mathcal{Y}_G| = |\mathcal{Y}_{G_1}| + |\mathcal{Y}_{G_2}|$ (exceeding 2000), with different bias towards a small set of identities (as measured by the ratio $|\mathcal{Y}_{G_1}|/|\mathcal{Y}_G|$). $N_1$ and $N_2$ represents the number of samples for $\mathcal{Y}_{G_1}$ and $\mathcal{Y}_{G_2}$. For the attack, $|\mathcal{Y}_F = 8631|$ identities are used (including $\mathcal{Y}_G$) and precision/recall are reported for identities from $\mathcal{Y}_{G_1}$ against $\mathcal{Y}_F \cap \mathcal{Y}_{G_1}$. The baseline is given by *random* guessing, which is the proportion of training identities $\frac{|\mathcal{Y}_{G_1}|}{|\mathcal{Y}_F|}$.

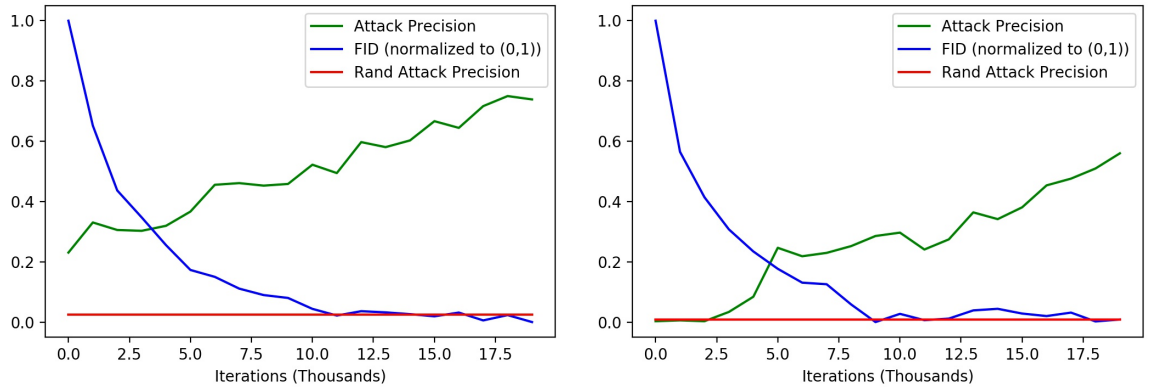



$|\mathcal{Y}_{G_1}| = 40, |\mathcal{Y}_{G_2}| = 2000, N = 52k$ $|\mathcal{Y}_{G_1}| = 160, |\mathcal{Y}_{G_2}| = 2000, N = 88k$

Figure 6.3.2: Precision-recall curves for membership attack on biased data described in Sec. 6.3.3. Attack is performed against the StyleGAN network on the VGGFace2 dataset, with various values of $\mathcal{Y}_{G_1}$. Even though both settings are trained with significant diversity (more than 2k identities and 50k images), the biased data in $\mathcal{Y}_{G_1}$ is still detectable.

### 6.3.5 Early Stopping

Thus far, we have only explored dataset size and diversity as possible factors effecting attack efficacy. However, a common tool in image classification to prevent overfitting is early stopping. In order to asses GAN sample quality, we refer to the commonly used Frechet Inception Distance (FID) [59]. Fig. 6.3.3 investigates the interaction between FID and the number of GAN training iterations. Indeed, beyond a certain iteration (here for StyleGAN, around 17000), continuing to train is detrimental to privacy, with no gain in FID.




Setting 1 ($|\mathcal{Y}_G| = 220$) Setting 2 ($|\mathcal{Y}_{G_1}| = 20$ & $|\mathcal{Y}_{G_2}| = 2000$)

Figure 6.3.3: FID vs attack precision. The graphs above suggest that early stopping may be useful in GAN training, w.r.t. privacy. While the FID converges, membership attacks continue to gain precision.

## 6.4 Discussion and Conclusion

In this chapter, we exposed several properties of GANs trained on facial data not previously discussed in the literature. By controlling the identities seen during training and subsequently detecting those identities with a face identification network, we demonstrated a successful blind membership attack. We identified several factors influencing susceptibility to attack, most notably that datasets with more identities have less detectable overfitting. Furthermore, dataset diversity alone will not protect against the presence of dataset bias. Finally, more iterations seemed to exacerbate attack success while not necessarily better image quality.

Contrary to most membership attack in the literature, this is a pure *black-box* attack in the sense that it does not require any further training based upon the generator, nor additional information about its architecture and parameters [146]. In addition, our attack is driven directly against the generator, and it did not require the exact training samples to be successful, as in the LOGAN approach. Finally, this work addresses a problem with training GANs on sensitive data (e.g. copyrighted or private face images). Careful dataset curation or early stopping may mitigate these problems. On the other hand, more sophisticated solutions may exist, such as those integrated into the GAN objective function, or those which directly modify the generation such as a post filtering.

# Appendix

## 6.A Additional Results for Setting 1

In this section, we provide additional results to for our attack against the LSGAN network. Table 6.A.1 and Table 6.A.2 show results for the VGGFace2 and CASIA datasets respectively.

| Method | $\lvert\mathcal{Y}_G\rvert$ ($N$) | 30 (10k) | 58 (20k) | 111 (40k) | 220 (80k) | 440 (160k) | 880 (320k) |
|---|---|---|---|---|---|---|---|
| Random | | 0.35 | 0.67 | 1.29 | 2.55 | 5.10 | 10.2 |
| Ours | Recall = 10% | 100.0 | 100.0 | 3.5 | 3.9 | 8.6 | 13.4 |
| | Recall = 50% | 100.0 | 100.0 | 3.0 | 2.9 | 6.1 | 11.7 |

Table 6.A.1: Setting 6.3.2 on VGGFace2 with the LSGAN network. Attacks are a random guess and ours.

| Method | $\lvert\mathcal{Y}_G\rvert$ ($N$) | 30 (4.5k) | 58 (8.7k) | 111 (16.65k) | 220 (33k) |
|---|---|---|---|---|---|
| | Random | 0.35 | 0.67 | 1.29 | 2.55 |
| Ours | Recall = 10% | 100.0 | 98.3 | 58.4 | 25.0 |
| | Recall = 50% | 99.3 | 90.3 | 24.4 | 19.3 |

Table 6.A.2: Same as table above, but on the CASIA-WebFace dataset.

With a high amount of data, at 80k images and 220 identities, attacks against the LSGAN network are near random guessing, whereas the Stylegan network is quite vulnerable to attack. This may be due to the fact Stylegan has more parameters than LSGAN (roughly 3x as many) and has higher capacity to overfit its samples. It's also interesting to note that Stylegan images are high quality (as measured by FID to training set), so that when the generator has evaded the attack as is the case for 880 identities, it is not merely because artefacts or distortion is throwing off the classifier.

## 6.B Results per-identity

Figure 6.B.1 shows the distribution of identities detected by the face identification network $F$ based on the number of samples recovered from StyleGAN trained on the VGGFace2 dataset, for different settings. Identities from $\mathcal{Y}_F \setminus \mathcal{Y}_G$ and $\mathcal{Y}_F$ are displayed in different colors to highlight the fact that the tail of the distributions corresponds to frequent recovered identities that are indeed in the training set $\mathcal{Y}_G$, and

prone to membership attack. Observe that the distribution has a lighter tail when the number of identities $|\mathcal{Y}_G|$ and the number of samples $N$ increase (first scenario), or when the bias ratio $\frac{|\mathcal{Y}_{G_1}|}{|\mathcal{Y}_F|}$ decreases (second scenario). As discussed later in Section 6.2.5, samples from these identities can be extracted to visually assess overfitting, as done in Fig. 6.1.1.

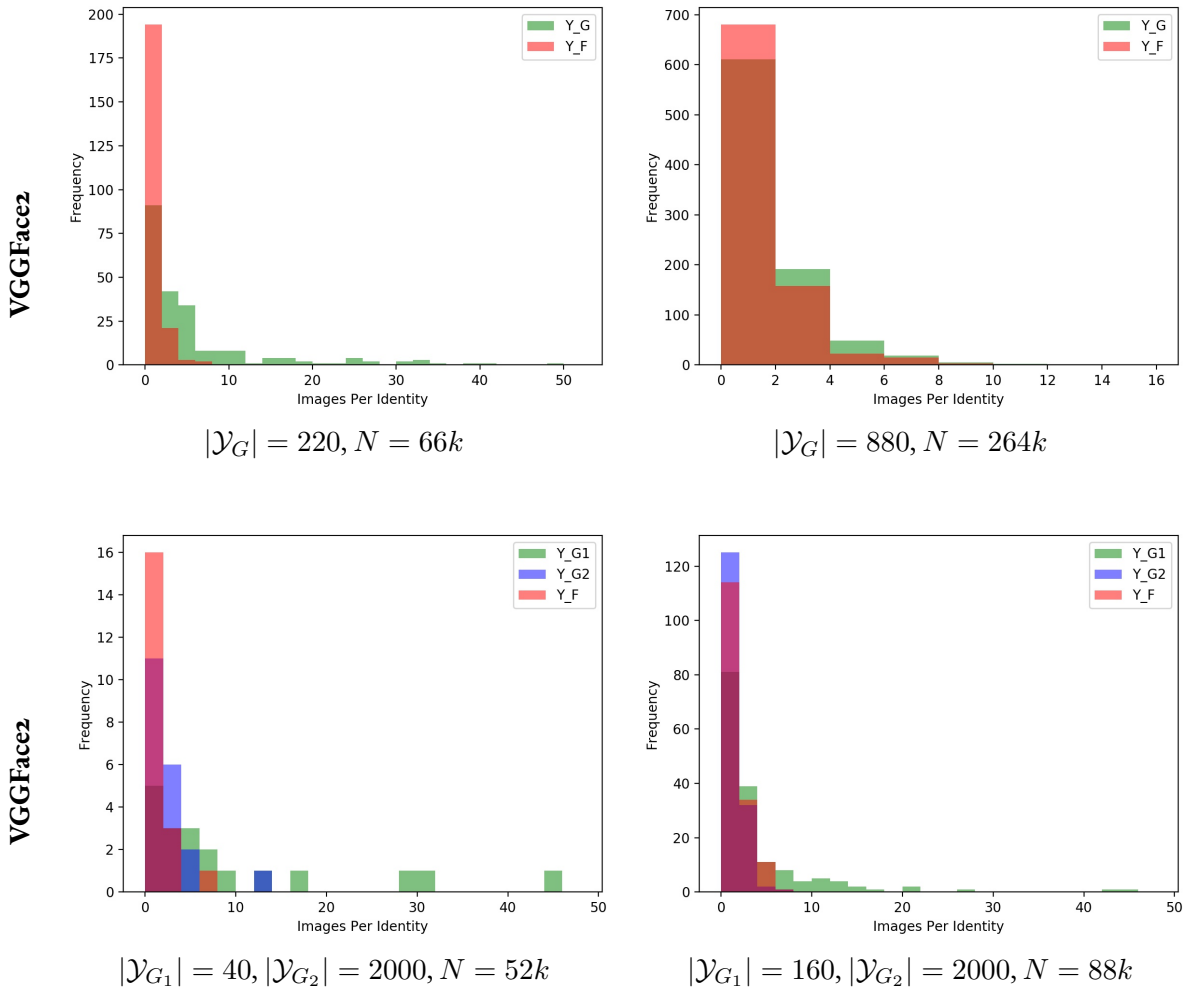



Figure 6.B.1: Per identity frequency histograms of generated samples. In the first row, StyleGAN generators are trained with unbiased data (setting 1), see Sec. 6.3.2. Without enough diversity, membership attacks are highly precise, as the bars in green (representing private training identities), can be easily distinguished. The second row showcases the high diversity & high bias scenario in Sec. 6.3.3. Here, blue represents a third, diverse auxiliary set, which cannot be distinguished, contrary to the biased samples (in green).

## 6.C Additional Visual Results

In this section, we view some additional visual results for various training sets and GAN training methods. We take generated samples, infer their class and then do an intra class search as in Eq. 2. The datasets with $|\mathcal{Y}_G| = 58$ (first two rows) contain less examples and therefore result in less realistic generations. However, the generated samples still highly resemble the training identities, albeit artifacted. With 220 identities, Stylegan is still overfitting despite a realistic synthesis.


Figure 6.C.1: **Additional nearest neighbor visual results.** Each row displays a GAN generated image (left) with three training images (right) having the same predicted identity. Notably, StyleGAN $|\mathcal{Y}_G| = 220$ achieves realistic synthesis while still overfitting.

Chapter 7

# An Extraction Attack Versus Diffusion Models

The membership attacks presented in Chapter 3 and Chapter 4 assume adversaries have some training images available to perform inference. However, in some settings, training samples may be unknown. Extraction attacks are when an adversary is able to reconstruct training images from model outputs. In this chapter, we'll study extraction attacks against text-to-image diffusion models and show it is possible to extract training images given only text captions.

Recent demonstrations show duplicates pose serious problems to large-scale text-to-image diffusion models. In this chapter, we provide an efficient pipeline to de-duplicate the most widely used public dataset LAION-2B and demonstrate that roughly a third of the dataset, around 700M images, are near duplicates. We then provide an efficient extraction attack, on par with the recent attack in Carlini et al. , that requires 3 orders of magnitude less network evaluations. In the process, we expose a new phenomenon, which we dub *template verbatims*, wherein a diffusion model will regurgitate a training sample largely intact. Template verbatims are more troublesome for newer systems, even those which de-duplicate their training set, and we give insight into why they appear. As our final contribution, to our knowledge we are the first to successfully extract images from Midjourney, a closed source model whose training set is unknown.

## 7.1 Introduction

Over the past few years, advances in large scale image generation systems have been brought on by publicly available billion-scale datasets [152, 151, 163, 144]. Due to their high quality, generality and ease of use, image generation systems such as Midjourney [116] and Stable Diffusion [163] have garnished millions of activate users, most of which have little technical knowledge. To train such models, largely automated bots search for suitable text and image pairs from all over the web, as is the case with the training set of Stable Diffusion (SD), LAION-2B (**L2B**) [151]. The widespread use of these generation systems is a testament to their generalization capacity; users often want to generate an image which does not exist or transform existing images to add new content. However, recent demonstrations show that popular systems such as Stable Diffusion can regurgitate exact copies of training images [126, 24, 161]. In [126, 24], where highly duplicated images appear to be a necessary but not sufficient precursor to model memorization.

The images extracted from diffusion models have important implications; in the medical domain, it's important that training data is not generated and for artistic use cases the generated images may pose a copyright risk to the model. In this work, we largely reproduce the extraction attack in [24], but much more efficiently and with several additional insights.

We provide the following contributions

- In Section 7.4, we present a feature compression method, build an image index on top of the compressed features and use it to de-duplicate **L2B**. We show that our de-duplication is in line with standard de-duplication methods, such as raw features or perceptual hashes. Notably, we show an extremely high level of duplication on **L2B**, with roughly 700M near duplicates.
- In Sec. 7.6, we present our training data extraction attack built upon our de-duplication. Notably, we show that only a single denoising step is required to unveil verbatim copied images and design an efficient black-box attack against diffusion models. After designing an evaluation protocol (Sec. 7.7), our attack is shown in Sec. 7.8 to be on par with the previous one with several order of magnitudes less network evaluations. Finally, we show how templates can be extracted from Midjourney and several other state-of-the-art models.

Code will be available online, as well as a version of **L2B** with near duplicates removed.

## 7.2 Related work

**Billions Scale Datasets** If the first "web scale" image dataset, LAION-400M, was released just a few years ago [152], even larger datasets have been subsequently released [22, 151]. The most widely used is the LAION-5B dataset, with roughly 5 billion text image pairs and the corresponding English language subset LAION-2B-en [151]. These datasets are automatically collected and text image/text pairs are only selected if they have a high enough CLIP score [138]. The CLIP network [138, 67], is trained to align image and text features with a contrastive loss and can provide a score that corresponds to a caption's relevance to an image. Thus, during construction of LAION-5B, CLIP features are computed, and the authors released CLIP features alongside the dataset.

**Deduplication** In [24, 126], it was noted that highly duplicated samples tend to be memorized by diffusion models and in [126], they de-duplicated before training. Likewise, in Stable Diffusion 2.0 [163], the author's used a perceptual hash to deduplicate before training. As we'll see in Sec. 7.8.3, this is effective at mitigating the verbatims found in [24], but not all. In any case, to study verbatims in existing systems, it is necessary to perform a billion scale de-duplication.

Many methods have been proposed for image de-duplication (e.g. [92, 93]), including those that use perceptual hashes [70] or end-to-end representations [136]. Unfortunately, this requires computation from the image domain and even storing billions of images on the cloud is expensive, not to mention computing descriptors. However, LAION [151] has released CLIP features for the entire dataset. In SemDeDup [1], these features were used to remove near duplicates on a subset of LAION, whilst keeping or improving downstream CLIP training. LAION has also released an index built upon these features using the popular tool Faiss [74]. In this work, we also build indices using Faiss, namely with product quantization [72] for de-duplication and later search.

**Membership Inference** In general, the process of determining which training samples were used to train a model is known as *membership inference*. For systems like DALLE2 or Midjourney, where the model parameters and training set are hidden, the task is known as the challenging "black box" scenario. Membership inference has been widely studied against GAN generated images, with varying success for a wide variety of different settings, for instance in [55, 28]. Several very recent approaches have designed black box attacks specifically against diffusion models [107, 39, 62], but conduct studies versus relatively small datasets (<1M samples). Namely, in the recent work in [62] a loss based attack is presented which bears resemblance to our white box attack (See Eq. 7.6), however they do so in the unconditional case.

**Extraction Attacks** A special case of membership inference, is the ability to actually reconstruct training samples from model outputs [26, 24]. Extraction is a much harder problem and typically is only pos-

sible for very few samples. In [24], only roughly 100 images were reconstructed from stable diffusion, out of 350K attempted prompts. Thus, this setting typically is only concerned with precision, and number of samples extracted, rather than the typical precision recall in membership inference. Extracted samples, however, clearly have higher implications to privacy or copyright.

## 7.3 Deduplicating LAION-2B

In this section, we describe an efficient method to de-duplicate LAION-2B. Whilst useful in its own right, we use this dedplucation in Sec. 7.6 for our extraction attack. A normal approach would involve computing a descriptor, such as in the state-of-the-art copy detection SSCD method [136], or the widely used perceptual hash [150]. Unfortunately, the typical cloud setup needed to download and store **L2B** is extremely expensive, not to mention doing a network pass on the entire dataset. We thus design a method to de-duplicate using the CLIP features already computed and released by LAION [151]. Still, doing de-duplication with raw CLIP features is still infeasible. Also, as we'll use search extensively to construct our ground truth (see Fig. 7.6.1 or Eq. 7.12), we choose to build an image index, capable of both semantic search and deduplication. An image index is a structure which both compresses a database of features, and can perform efficient search [74]. Ultimately, our image index closely matches the one in [74], with a compression step that is better adapted for CLIP features.

## 7.4 CLIP Feature Compression

We begin with a generic auto-encoding baseline using mean squared error, which is a standard technique for feature compression [167]. Denoting $(x^T, x^I)$ as minibatches of CLIP text and image features we have

$$\mathcal{L}_{\text{MSE}}(x^I; E^I, D^I) = \|x^I - D^I(E^I(x^I))\|^2. \tag{7.1}$$

where, $D^I, E^I$ are the encoder/decoder networks for the image features. The compression rate is thus controlled by the output dimension of $E^I$, or the latent space dimension, which will be used for index creation and search later on.

The compression can be done on either image CLIP or text CLIP descriptors. However, the limitation is that if the compression of the two modalities is done independently of each other, there is a risk of losing the alignment between the modalities. Our experiments with hybrid encoders (fusing modalities or mixing contrastive and reconstruction losses) showed worse results than this baseline or the methods presented next.

**Contrastive Compression** We propose a second compression method designed to preserve text and image feature alignment. To achieve this, we suggest using the original contrastive loss function proposed in [138]. There are several approaches to compressing features this way, including the use of an autoencoder or applying the clip loss directly in the original feature space. However, we found that using a "latent" clip loss as $\mathcal{L}_{\text{CLIP}}(E^T(x^T), E^I(x^I))$, with $E^I, E^T$ as the image and text encoders, was better overall and use this loss for all experiments.

Although using CLIP loss alone was effective in some tasks, such as zero-shot ImageNet classification, we observed that to enable more accurate nearest neighbor search, it was better to also introduce a term to maintain distance properties between neighboring elements in the dataset. Therefore, we propose a new approach that involves computing the nearest neighbors (w.r.t. image features) in a "chunk" (here, we used sliding chunks $k = 10$M samples), defined as follows

$$\mathcal{L}_{\text{SNIP}}(x^T_k, x^I_k; E^T, E^I) = \mathcal{L}_{\text{CLIP}}(E^T(x^T_k), E^I(x^I_k)) + \lambda\, \mathcal{L}_{\text{CLIP}}(E^I(x^I_k), E^I(x^I_{\text{1-NN}(x^I_k)})) \tag{7.2}$$

where 1-NN($x_k^I$) is the nearest neighbor of sample $x_k^I$ in the chunk $k$. The nearest neighbor search within each chunk is done exhaustively by measuring the L2 distances between all the pairs of the chunk.

**Approximate Search** Even compressed, searching through CLIP descriptors at billions scale is still too expensive. We therefore use techniques for approximate nearest neighbor search akin to those found in Johnson *et al.*[74]. One of the simplest and oldest techniques for fast nearest neighbor search relies on an inverted file system (IVF). First one computes the k-means centroids of the database vectors and then groups vectors to their closest centroid to form the inverted files. During search, queries only look through the $\tau$ closest inverted files by exhaustive search over the closest centroids, which is tractable because the number of centroids is typically small. We explore building a two-level quantizer on top of our compressed features, which compresses database vectors as follows:

$$y \approx q_1(y) + q_2(y - q_1(y)) \tag{7.3}$$

where $y$ is first approximated with $q_1$, and then its residual quantized by $q_2$. $k$-NN's are retrieved as those minimizing a quantity known as the asymmetric distance:

$$d_{ADC}(x, y) = \|x - q(y)\| \tag{7.4}$$

where $q(y)$ is a quantized database vector. In the case when $q_1$ is an IVF, this quantity is only minimized over database vectors mapping to the same centroid as the query or top $\tau$ centroids (referred to as the nprobe parameter). Specifically, we explore Product Quantization (PQ) as our secondary quantizer [72]. PQ splits vectors into $M$ sub-vectors, $y = [q^0(y^0), ...q^M(y^M)]$ where $M$ is an even divisor of $y's$ dimension and the sub-quantizers are again k-means. Higher values of $M$ result in higher compression ratios.

## 7.5 Image Similarity Search

We generate a synthetic ground truth by computing a small set of $k$-nearest neighbors using brute force. We then compared the index retrieval results with this ground truth as follows:

$$R \cap k := \frac{1}{N} \sum_{q \in Q} \frac{|k\text{-NN}(q_i) \cap k\text{-ANN}(q_i)|}{k} \tag{$R \cap k$}$$

The approximate $k$-nearest neighbors ($k$-ANN) were computed for the index under review, while the exhaustive search on raw features ($k$-NN) served as the ground truth. It's worth noting that the approach used to construct the ground truth for the (non-multimodal) Deep1B dataset [12] is similar to the one described in [74].

We did ablations over the latent space dimensions and found that encoding to a relatively high dimension of 128 first (from 1024 for ViT-H-14) was more effective, than for example encoding to a low dimension directly and giving more memory to the quantizer. We note that the approach in [74] also used a similar dimension for the compressed space before quantization. Also like in [74], we used the IVFPQ index for every index we built atop of our descriptors and finally did several comparisons to a vanilla IVFPQ on raw features.

For the IVF, we chose $2^{16}$ centroids, and for the PQ we constructed collections of indices with different values of $M$ (the number of dimension chunks) with $2^8$ centroids each. Finally, we highlight the popular open source tool AutoFaiss, which will automatically construct and efficient PQ index given a memory budget. All of our AutoFaiss indices use multiple PQ steps, with an initial optimized product quantization (OPQ) step for dimensionality reduction (see [50]), as well as a hierarchical navigable world search (HSNW) [105]. For instance, when providing the tool with "6G" total memory for the index, it produced a faiss construction pipeline of "OPQ16_112,IVF65536_HNSW32,PQ16x8" (Please see the faiss

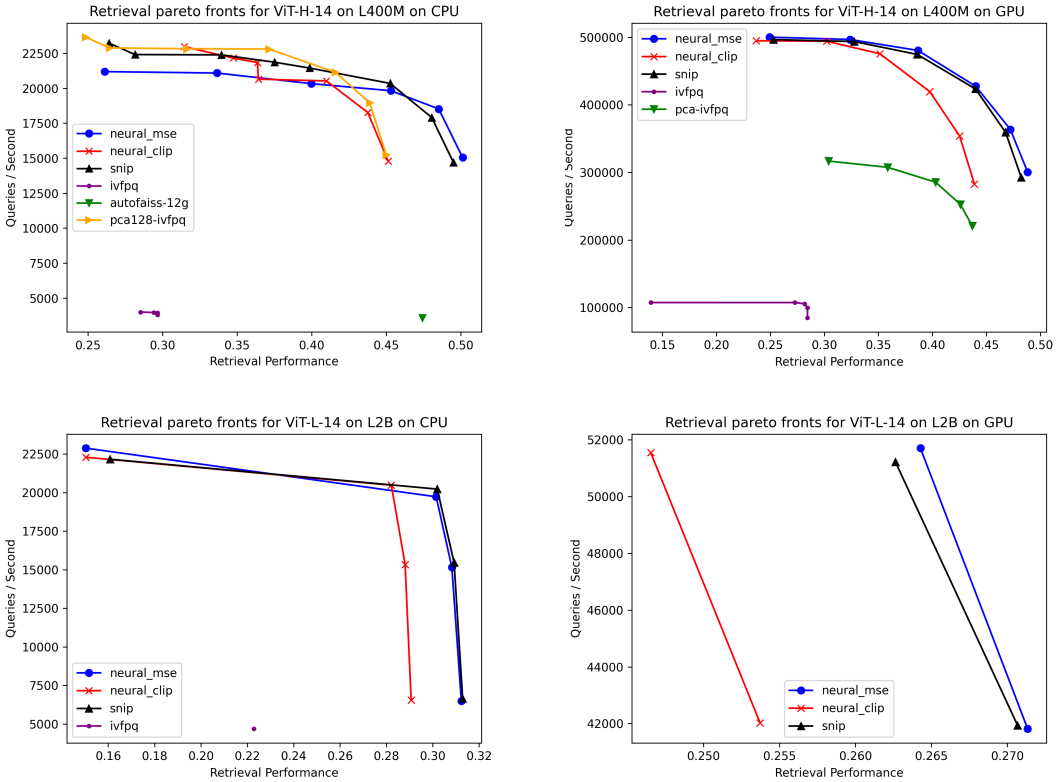



Figure 7.4.1: Image feature only retrieval pareto fronts for ViT-H-14 indices on L400M on the CPU (top left), GPU (top right) and on L2B on CPU (bottom left) and GPU (bottom right). MSE based losses perform similarly on this task compared to the contrastive ones, and the SNIP loss performs better than the a contrastive only CLIP loss.

library for more details[74]). We noted these indices were not more effective on the GPU and excluded them from these benchmarks. To form Pareto fronts, we merely built a collection of indices as before, but varying the memory parameter.

**Multi-modal Search Results** The original CLIP network demonstrated impressive performance on the challenging zero-shot ImageNet classification task. With CLIP, this can be achieved by getting an encoding for each category $c$ with the caption "this is an image of [$c$]", and then performing an exhaustive search with CLIP encoded ImageNet samples via taking a maximum inner product (as is maximized for positive pairs during contrastive training). We'd like to not only explore how well our descriptors preserve multi-modal information, but the quantized representation in the index as well. For this, we reconstruct database vectors using the `sa_encode` and `sa_decode` functions of the faiss library, and then do multi-modal search on reconstructed vectors. We compare to AutoFaiss (see previous section), which is used for multi-modal search hosted by LAION [151], but in a more extreme compression regime.

Figure 7.8.2 shows the performance of each index type versus the "compression ratio," which is simply the uncompressed size of all database features divided by the index size (plus a negligible size of the encoding network). Again, we create a collection of indices with varying $M$ parameters to form the IVFPQ pareto fronts. The $\mathcal{L}_{\text{MSE}}$ descriptor indices zero-shot performance are clearly less efficient than the $\mathcal{L}_{\text{SNIP}}$ ones. Autofaiss performs the worst at these compression ratios (in general we saw a sharp drop in the AutoFaiss performance above a certain compression ratio for every task). For text-to-image search, our snip model also performs better than AutoFaiss. We now move on to a wider variety of comparisons.

| Index | Size (GB) | Duplicates Found | Image MSE | pHash |
|---|---|---|---|---|
| ViT-H-14 SNIP (GPU,M16) | 49G | 602M | 98 | 92 |
| ViT-L-14 RECON (GPU,M16) | 49G | 692M | 97 | 84 |
| ViT-L-14 (CPU,M4) | 25G | - | 65 | 47 |
| MD5 file hash | - | 485M | 100 | 100 |

Table 7.5.1: Consistency of the proposed indices for deduplication versus other standard measures. Note that whilst the precision versus a perceptual hash is sometimes low, many of these images are still near duplicates (see the supplementary). The score (in %) is calculated for 100K random pairs marked as duplicates.

**Image to Image Search Results** Figure 7.4.1 presents the retrieval benchmarks for the L400M and L2B datasets. We assessed each method across a range of benchmarks and created Pareto fronts by varying the number of chunks for product quantization (i.e., $M = 2, 4, 8$) and the n probe parameter ($\tau = 1, 2, 4$). Notably, MSE and SNIP performed similarly on image similarity search, while CLIP-only loss networks consistently performed worse. However, as mentioned, the CLIP networks (both CLIP and SNIP) do perform better on multi-modal tasks. Thus, SNIP can be seen as a best of both worlds descriptor for these types of indices. Figure 7.4.1 also indicates that indices similar to those in [75], using a PCA as a first compression and IVFPQ as a quantizer, have some drop off in performance, as with the CLIP only indices.

### 7.5.1 De-duplicating with the ADC

Normally, de-duplication with descriptors is done by threshing distances to nearest neighbors. Our indices return the asymmetric distance (7.4), which can result in non-zero distances even for in-database queries. Additionally, since the secondary quantizer operates globally over the IVF, different centroids may contain significantly different $ADC$ distributions. Consequently, we found that applying an absolute threshold on $ADC$ alone was insufficient for effective de-duplication performance.

We thus use an adaptive threshhold. This approach takes into account that duplicated images may not necessarily have a low absolute $ADC$, but they will all have the same $ADC$. Specifically, duplicates $y \in k\text{-NN}(x)$ of a query image $x$ are identified if

$$\frac{|ADC(x, x) - ADC(x, y)|}{ADC(x, x)} < T_{ADC}. \tag{7.5}$$

### 7.5.2 High level of duplication on L2B

To de-duplicate, we compute (7.5) for every database vector, against its $k = 32$ nearest neighbors (given via our index). We form a large sparse graph on disk, and propagate transitivity using a stochastic variant the FastSV algorithm [192] (see the supplementary for details).

Our goal is to extract as many duplicates as possible, so long as marked duplicates are consistent with standard de-duplication measures. To evaluate, we draw 100k random pairs of duplicates by our method, fetch the images and compute a variety of other metrics (i.e. we measure precision using other methods as a ground truth). Table 7.5.1 presents the results for several de-duplications we ran. We found it surprising that roughly a third of L2B are near duplicates. Furthermore, LAION also provided MD5 file hashes and surprisingly, 25% of the dataset are duplicates at the file level. We note that even samples marked as "false positives" by a pHash, are still near duplicates by inspection (for instance, up to imperceptible artifacts or a resizing).

## 7.6 Attack Model

In this section, we present extraction attacks on conditional diffusion models. This attack seeks to find any image generated via a text prompt, which is identical to a training set image (up to mild perturbations). Such an attack was proposed in [26], where the prompts of highly duplicated training examples of **L2B** are chosen as potential copies and then (un)validated based on the variability of generated samples from each prompt. This attack requires a large number of generation per candidate prompt. Compared with this attack, we make the following contributions: 1) we improve the efficiency of the highly duplicated candidate selection by relying on our ViT-H-14 SNIP approach (see Table 7.5.1); 2) we found verbatim copies have the property that they can be synthesized in a single iteration. Indeed Fig. 7.6.1 shows clearly different behaviors for prompts that are copied versus those that are not. The intuition is that for regurgitated samples, the denoiser will immediately map the noise image far from its starting point. Incidentally, the output of the denoiser after a single step already resembled a natural image (the one that would be generated if a full synthesis had been performed). We use this as an intuition to construct both attacks presented next.


Figure 7.6.1: Training images can be extracted from Stable-Diffusion in one step. In the first row, a verbatim copy is synthesized from the caption $c$ corresponding to the image $x$ on the last column. In the second row are *template verbatims* copies that are harder to detect because they are related to many prompts variations. These images are detected using CLIP retrieval as detailed in (7.12). They exhibit variations in fixed locations of the image; here the color of the carpet can change so they must be masked to be correctly detected. Non-verbatims have no match, even when retrieving over the entire dataset. See Sec. 7.7 for how we construct the ground truth here in this case.

### 7.6.1 White-box Attack

In this setting, we assume the attacker has both the captions and the model parameters. We propose an easy to compute metric, which can capture the one-step synthesis property illustrated in Fig. 7.6.1, although for stable diffusion it works in the first stage encoder latent space (see [144] for details). Recall that conditional diffusion models train a denoising autoencoder $D(z_t, c)$, which will predict the real image (or latent image) $z_0$ given its noised version $z_t$. Note that the synthesis process starts at step $T$ from a white Gaussian noise $z_T$. Our metric relies on measuring how far apart stands the input noise $z_t$

from its denoised version $D(z_t, c)$:

$$\text{DCS}(c) := \|z_T - D(z_T, c)\|_2^2 \tag{7.6}$$

We call this error the Denoising Confidence Score (DCS). We can turn this into a classifier marking extracted samples by thresholding this value with $\tau_{\text{DCS}}$ as follows

$$F_{\text{DCS}}(c; \tau) = \mathbb{1}_{\text{DCS}(c) \geq \tau_{\text{DCS}}} \tag{7.7}$$

Whilst referred to as "whitebox", we in fact do not need access to the parameters of the diffusion model, just the input noise $z_T$.

### 7.6.2 Black Box Setting

We consider the setting similar to [24], wherein an attacker has captions but can only invoke $\texttt{Gen}(c)$ for caption $c$ (or $\texttt{Gen}[r](c)$ if we need to specify the random seed $r$), and we also assume the ability to control the timesteps $T$ (as is the case for Midjourney). Henceforth, we use $\texttt{Gen}(c, T = 1)$ when we respace to a single time step and simply $\texttt{Gen}(c)$ for full synthesis (i.e. $T = 50$). We noticed non verbatims normally start as highly blurry images (Fig 7.6.1, last row). Blurry images will not have consistent edges (or no edges at all) in contrast to the realistic images that emerge from certain prompts. We thus employ a simple image processing technique by first computing the images edges, and then look for how consistently these same edges appear for other seeds. Letting the operator $\texttt{Edge}()$ outputting a binary image of contours (e.g. with a sobel filter and thresholding), we define the Edge Consistency Score (ECS) as follows

$$L_{\text{ECS}}(c) = \left\| \left( \frac{1}{J} \sum_{j \leq J} \texttt{Edge}(\texttt{Gen}[r_j](c, T = 1)) \right) \geq \gamma \right\|_2^2 \tag{7.8}$$

Where $J$ represents generating over several random seeds and $\gamma$ a threshhold. Whilst rather simplistic, we find that this score works decently in practice.

(a) Midjourney v4 [116]


(b) Deep Image Floyd [157]

(c) Stable diffusion v1 [144]

(d) Stable diffusion v2 [163]

Figure 7.6.2: Template verbatims for various networks: Left is generated, middle is retrieved image and right is the extracted mask. Template verbatims originate from images that have variation in fixed spatial locations in **L2B**. For instance, in the top-left, varying the carpet color in an e-commerce image. These images are generated in a many-to-many fashion (for instance, the same prompt will generate the topleft and bottom right images, which come from the "Shaw floors" series of prompts).

## 7.7 Constructing a Ground Truth

As we'd like to evaluate the precision of the above attacks, we need to ascertain prompts that actually lead to copies, by accessing all images on **L2B** For dataset pair $(x_D, c)$, we generate image $x_G = \texttt{Gen}[r](c)$ and then mark them as copied entirely for the corresponding image $(x_D)$ or copied with respect to a mask on any dataset image (retrieval verbatims).

**Matching Verbatims (MV)** In [24], the dataset image $x_D$ is by-design chosen among the one having caption $c$, and the corresponding generated image $x_G$ is determined as the minimizer (over a fixed number of seed trials) of:

$$L_{MV}(x_D, c; J) = \min_{j \leq J} \|x_D - \texttt{Gen}[r_j](c)\|_2^2 \tag{7.9}$$

One needs to take the minimum over $J$ random seeds $r_j$ as sometimes verbatims appear after a few seeds. Then, the ground truth labels are simply a threshold on this distance

$$\texttt{IsVerb}(c; J, \delta_V) = \mathbb{1}_{L_{MV}(x_D, c; J) \leq \delta_V} \tag{7.10}$$

We chose $\delta_V = .12$. This is slightly more relaxed than what was chosen in [24], and we simply prune false positives by hand (such as images of textures or without objects that can be false positives), which given the rarity of positively labeled images is feasible.

**Retrieval Verbatims** We noticed that some images would not correspond to their matching image, despite having the curious property of one-step synthesis (see Fig. 7.6.1). Thus, we retrieved the training images with an index our ViT-H-14 SNIP index (Sec. 7.5). More precisely we extract $x_G$ and $x_D$ as the minimizer of

$$L_{RV}(c) = \min_{j \leq J} \min_{x \in NN_{K,j}} \|x - \texttt{Gen}[r_j](c)\|_2^2 \tag{7.11}$$

where $NN_{K,j}$ denotes the set of K nearest neighbor in the training set of the generated image $\texttt{Gen}[r_j](c)$ (retrieved via our index constructed in Sec. 7.4). We call these retrieval verbatims (RV).

Furthermore, we found that many images which had very few duplicates had extremely many near duplicates that differed in only one region of the image. These images would commonly be e-commerce images that would vary an aspect of the sale item: e.g. an image of furniture with a large variety of carpet colors. We thus update our retrieval verbatim condition to accommodate masking as follows

$$L_{TV}(c) = \min_{j \leq J} \min_{x \in NN_{K,j}} \|m(x) \odot (x - \texttt{Gen}[r_j](c))\|_2^2 \tag{7.12}$$

Here $m(x)$ denotes a spatial mask corresponding to the training image $x$. In practice we manually select samples that had near duplicates through Eq. 7.11 and appeared to be templates, and pruned images that were adversarial to the MSE (such as those with all white backgrounds). We note this procedure could potentially be automated, for instance with object detection on the unmasked regions.

**Evaluation** Of course, we will not be able to construct the ground truth for every image of L2B, as it entails generating images. Besides, extraction attacks typically are concerned with precision and number of samples found, rather than recall. We thus only generate and compute the ground truth for the top images selected by our attacks Eq. 7.6 and Eq. 7.8 and measure precision versus number of verbatims found.

## 7.8 Results

In this section, we evaluate our white box and black box extraction attacks against several popular diffusion models. We evaluate each in the following way

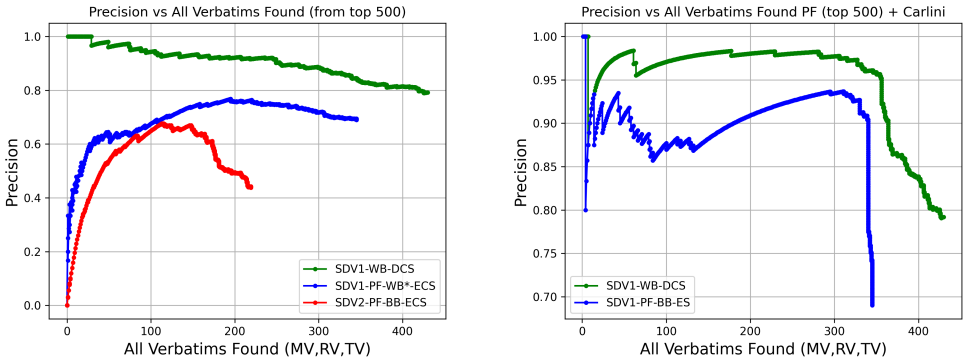



Figure 7.7.1: Precision recall curves for the whitebox attack and black box attacks (see Sec. 7.7. For both black box settings, we first pre-filter for the top 30K images selected via the whitebox score, then sort with the black box edge score. On the right, we sort via the black box score, synthesize 500 samples (setting "+Carlini," [24]) and then perform their attack. This makes the attacks much more precise.

**Whitebox** We take the 2M most duplicated samples and compute the DCS Eq. 7.6 for every caption for the Stable Diffusion v1 network [144]. This attack takes only a single unet evaluation per caption.

**Blackbox** This setting requires several unet evaluations per caption (the $J$ parameter in Eq. 7.8), which is too expensive for millions of captions. Thus in practice, we pre-filter using the whitebox DCS (only for SDV1) for the top 30k samples and compute our black box $L_{\text{ECS}}$ Eq. 7.8. For a black box attack versus stable diffusion V1, this is technically not black box. Even so, extracting verbatims from 30k images still is a strong indicator of the success of a true black box attack given their rarity (around 1% chance of being randomly selected). For black box attacks against stable diffusion V2, this is a purely black box procedure. We compute Eq. 7.8 for $J = 4$ samples and still only one timestep of the Heun sampler; for a total of 4 unet evaluations per caption. In [24], they performed 500 full generations. Assuming a conservative estimate of 16 iterations per generation (although they used 50 in the paper), makes our attack $500 * 16/4 = 2000$x times more efficient than the attack in [24].

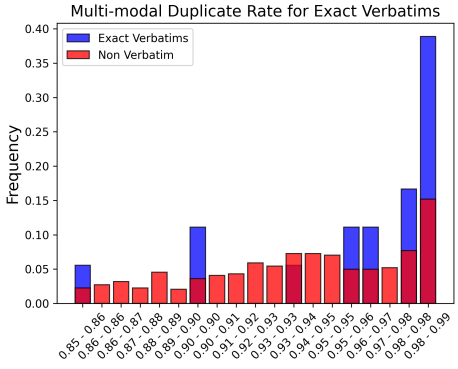
**Post filtering with Carlini et. al** Our attack is not incompatible with the one in [24]. For those images we deem most likely to be verbatims (given our whitebox or black box method), we then synthesize 500 samples per caption and mark captions as verbatims when inter-sample synthesis is overly repetitive (i.e. there is many duplicate synthesis images).

### 7.8.1 Analysis

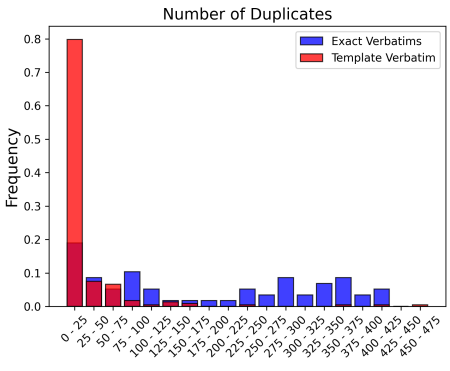
Fig. 7.7.1 shows the precision of the attacks versus the number of verbatims found for Stable Diffusion v1 and v2. Our whitebox attack has much higher precision than the black box variants. The black box attacks start with low precision, as there were some samples that were "adversarial" to the method, such as images of textures or that were mostly backgrounds. When combining our black box attack with [24], our black box attack becomes much more precise.

### 7.8.2 What Makes Prompts Prone to Regurgitation?

It is still unclear why diffusion models will regurgitate some samples and not others. Duplication alone is not a sufficient condition for the model to memorize the sample. The authors in [24] found that anomaly detection was not successful at detecting them. As we de-duplicate with image features, we explore what percentage of these samples are multi-modal duplicates. In Fig. 7.8.1 we show the percentage of features




(a) Multi-modal duplication rates for exact verbatims Eq. 7.9 versus non-verbatims. Exact verbatims tend to be multimodal duplicates (duplicates in text and image) more often than other sample highly duplicated in image feature space only.




(b) Duplication rates for exact verbatims versus template verbatims. Template verbatims may be highly duplicated w.r.t. their mask, but without the masks they have relatively low levels of duplication as they variable parts of the image.

Figure 7.8.1: Histograms of duplication rates.

that share the same prompt within a duplicate group. The verbatim samples show a significantly higher rate of multimodal duplication than non-verbatims (randomly chosen in the top 2M most duplicated).

**Template Verbatims** Template verbatims are more difficult to label as ground truth, not only because they require retrieval and masking, but also because they're not highly duplicated, as shown in Fig. 7.8.1. Note that Stable Diffusion v2 did deduplicate **L2B** before training. Unsurprisingly, we did not find many examples of exact verbatims being copied on Stable Diffusion v2, however, we still found many template verbatims, see Tab 7.8.1. Thus, a more relaxed duplication detection, such as the semantic duplicates found in SemDeDup [1], may be necessary to weed out these samples. We leave this for future work.

### 7.8.3 Extracting Verbatims in Other Models

| Model | Deduplicated Training? | Retrieved | Template | Exact |
|---|---|---|---|---|
| Stable Diffusion V1 [144] | No | 37 | 45 | 71 |
| Stable Diffusion V2 [163] | Yes | 0 | 21 | 4 |
| Deep Image Floyd [157] | Yes | 0 | 15 | 2 |
| Midjourney v4 [116] | Unknown | 5 | 8 | 2 |
| OpenJourney (from [32]) | No | 14 | 29 | 73 |
| RealisticVision (from [32]) | No | 15 | 32 | 90 |

Table 7.8.1: Number of ground truth verbatims extracted from several models. For deep floyd and Midjourney, we use the top 500 prompts sorted from Eq. 7.8 from SDV1 (i.e. we don't perform the attack, just extract images). De-duplicated models seem less susceptible to exact verbatims, but are still vulnerable to template verbatim extractions. See Sec. 7.8.3

Having obtained our ground truth for Stable Diffusion V1 (SDV1) and V2, we test whether these prompts are also verbatim copied by a variety of other models, such as state of the art diffusion model DeepIF [157], and the closed source system MidJourney [116]. For Midjourney, this was done manually through discord for around 100 prompts, and we swept over the number of time steps until a verbatim


Figure 7.8.2: Several shortcomings of using an MSE for ground truth labeling. On the left is the real image and on the right generated. These samples fall just below our MSE threshold.

was found. Tab. 7.8.1 shows the total number of ground truth verbatims extracted. With Midjourney [116], the model is entirely black box, and likewise for the training set. Still, we find it still regurgitates some of the same prompts as other models, which are known to be trained on **L2B**. Interestingly, it is also less susceptible to the exact verbatim regurgitation like SDV2, so we hypothesize that Midjourney de-duplicated their training set before training. Finally, we note the popular stable diffusion checkpoint model openjourney, which was fine tuned from SDV1 using images and prompts from Midjourney, and typically generates fantasy and surreal images better than the original model. In contrast, realistic vision, which is also a checkpoint, focuses on more photographic realism. Both models regurgitate the almost the same prompts, and for realistic vision, the problem is exacerbated.

### 7.8.4 Limitations

Fig 7.8.2 shows shortcomings for the ground truth construction. In many images, the retrieved image is cropped and scaled slightly different (bottom left), and thus was not labeled as a template verbatim. In general, for future work, it may be worthwhile to explore more flexible copy detection, such as those which are invariant to some permutations of patches [13]. This would also cover the patch copying phenomenon observed in [161], which our template verbatim Eq. 7.12

## 7.9 Conclusion

In this work, we presented an extraction attack successful versus several widely used diffusion models. First, we designed an efficient de-duplication routine over compressed CLIP features and revealed that roughly a third of **L2B** are near duplicates. Our attack was on par with previous methods, whilst requiring significantly less network evaluations. Furthermore, we extended the previous work to accommodate template verbatims, i.e. images that showed non meaningful variations in fixed locations in the image. We shed insight into why these images still appear even in models which have deduplicated their training data, such as Stable Diffusion 2.0 and deep image floyd; they are not highly duplicated in the standard sense, but likely are highly duplicated with respect to a mask. We believe this work can not only help build better datasets but more useful and safe generative models.

# Appendix

In this section, we give additional details for our indices constructed in this chapter and their performance.

**Notes on Architecture and Training** We use an MLP architecture for $E$ in all experiments, containing a batch norm at the input, two MLP blocks consisting of a linear layer projecting to an intermediate dimension of 512, gaussian error linear units (GELUs) and finally a linear layer projecting to the compressed latent dimension. For the autoencoder networks, the decoder $D$ is simply the transpose of $E$ (in the sense of the layer definition, the parameters are not shared). We did architectural ablations, including choice of non-linearity and choosing architectures and found that they do not improve retrieval performance (see next section). Finally, for L2B, we perform 2 epochs over 200M features and 50M for L400M.

**Additional retrieval results** Tab. 7..1 and Tab. 7..2 summarizes the results for the ViT-B-32 and ViT-H-14 networks. The imagenet zero-shot score is computed against the ImagenetV2 dataset [141], hence the slightly lower than normal scores (even for raw features). Note that SNIP is performing overall the best, with the best imagenet zero-shot score, as well as nearly the best retrieval accuracy (for $M$=16). As was discussed before, the MSE nets perform poorly on the (multi-modal) Imagenet zero-shot task, but quite well on retrieval. We also did some ablations on the encoding dimension $d$ before quantizing with IVFPQ. For instance, see $d = 32$ for the CLIP and SNIP networks. Note that the index size will be the same (and in fact, just depends on the IVFPQ parameters), however, doing so in this way shifts the compression to the neural compression step. However, this is met with a drop in performance, which gets worse as the latent dimension becomes smaller.

For ViT-B-32, whilst observing the same behavior for compressed CLIP representation using encoders trained with MSE and SNIP losses, it is interesting to notice the expected drop in performance in zero-shot retrieval when using ViT-B-32 instead of ViT-H-14, as already observed for uncompressed features. We also note that ViT-B-32 nets have a larger relative drop in performance on the imagenet zeroshot task, compared to ViT-H-14 networks.

| Dimension Reduction | Index Info | Memory (GB) | $R \cap k$ | IN Zero-shot Top-1 |
|---|---|---|---|---|
| Raw ViT-B-32 | - | 239 | 100 | 56.1 |
| $\mathcal{L}_{\text{SNIP}}$, $d = 128$ | IVFPQ, M=4 | 2.2 | 25.4 | 31.4 |
| $\mathcal{L}_{\text{SNIP}}$, $d = 128$ | IVFPQ, M=16 | 4.2 | 35.5 | 34.5 |
| $\mathcal{L}_{\text{MSE}}$, $d = 128$ | IVFPQ, M=4 | 2.2 | 25.1 | 20.7 |
| $\mathcal{L}_{\text{MSE}}$, $d = 128$ | IVFPQ, M=16 | 4.2 | 35.4 | 30.4 |
| OPQ, $d = 56$ | AutoFaiss | 2.9 | .05 | 1.0 |
| OPQ, $d = 168$ | AutoFaiss | 5.6 | 21.4 | 23.1 |

Table 7..1: Comparisons of the k-NN accuracy (with k=5) of different networks and losses on ViT-B-32 [138] for queries outside of the database vectors, against an exhaustive search. The compressed CLIP features have a dimensionality of $d$, and the number of chunks for product quantization is denoted by $M$.

| Dimension Reduction | Index Info | Memory (GB) | $(R \cap k)$ | IN Zero-shot Top-1 |
|---|---|---|---|---|
| Raw ViT-H-14 | - | 708 | 100 | 67.1 |
| $\mathcal{L}_{\text{SNIP}}$, $d = 128$ | IVFPQ, M=4 | 2.2 | 29.2 | 40.3 |
| $\mathcal{L}_{\text{SNIP}}$, $d = 128$ | IVFPQ, M=16 | 4.2 | 49.5 | 48.7 |
| $\mathcal{L}_{\text{SNIP}}$, $d = 32$ | IVFPQ, M=4 | 2.2 | 36.4 | 39.2 |
| $\mathcal{L}_{\text{MSE}}$, $d = 128$ | IVFPQ, M=4 | 2.2 | 43.1 | 30.6 |
| $\mathcal{L}_{\text{MSE}}$, $d = 128$ | IVFPQ, M=16 | 4.2 | 50.1 | 42.3 |
| $\mathcal{L}_{\text{CLIP}}$, $d = 128$ | IVFPQ, M=4 | 2.2 | 32.1 | 40.2 |
| $\mathcal{L}_{\text{CLIP}}$, $d = 32$ | IVFPQ, M=4 | 2.2 | 28.1 | 39.4 |
| $\mathcal{L}_{\text{CLIP}}$, $d = 32$ | IVFPQ, M=8 | 2.9 | 35.5 | 42.2 |
| $\mathcal{L}_{\text{CLIP}}$, $d = 8$ | IVFPQ, M=4 | 2.2 | 19.2 | 0.0 |
| OPQ, $d = 56$ | AutoFaiss | 2.2 | 0.1 | 2.7 |
| OPQ, $d = 112$ | AutoFaiss | 2.9 | 1.6 | 8.7 |
| OPQ, $d = 168$ | AutoFaiss | 5.6 | 44.1 | 43.4 |
| - | IVFPQ, M=4 | 2.4 | 14.6 | 40.4 |
| - | IVFPQ, M=16 | 5.5 | 44.7 | 46.3 |

Table 7..2: Comparisons of the k-NN accuracy (with k=5) of different networks and losses on ViT-H-14 (OpenClip [67]) for queries outside of the database vectors, against an exhaustive search. The compressed CLIP features have a dimensionality of $d$, and the number of chunks for product quantization is denoted by $M$.

**Visualization of duplicates** In this section, we provide some visual results for how CLIP duplicates compare to MSE image space duplicates. Figure 7..1 shows some false positives and false negatives for CLIP.

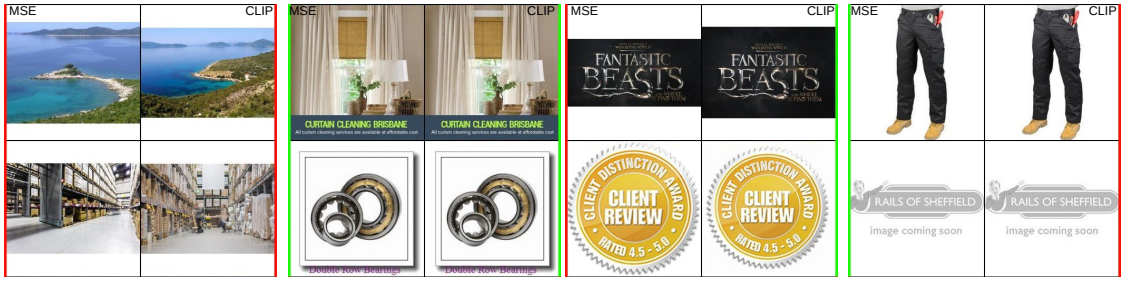
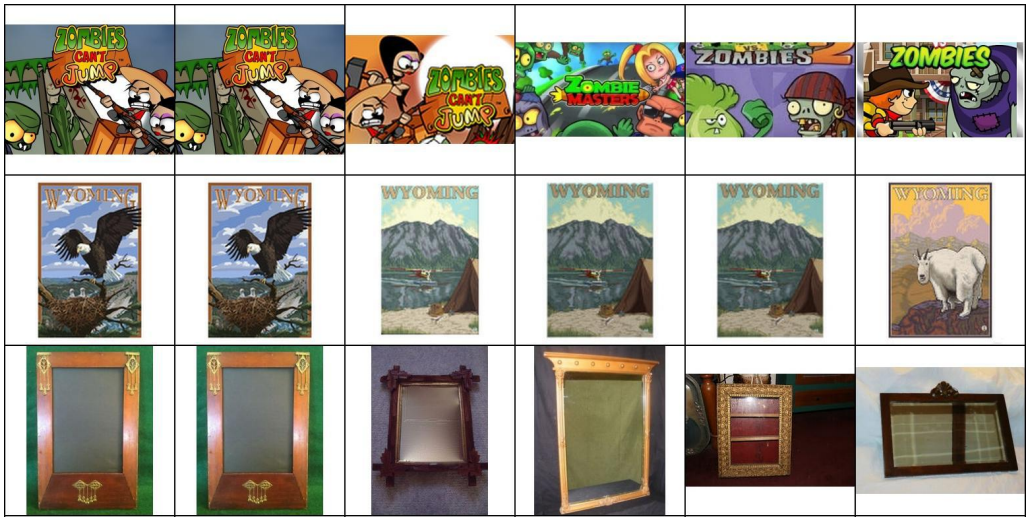
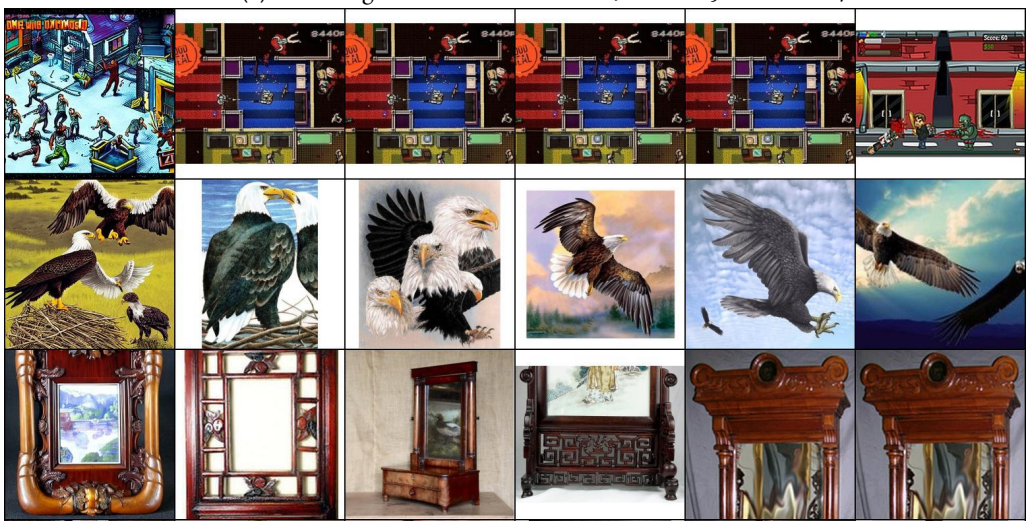



Figure 7..1: Classification as duplicates or non duplicates by thresholding MSE or CLIP distances. For each image pairs a red/green bar on the left (resp. right) indicates an MSE (resp. CLIP) non-duplicate/duplicate. Examples in the leftmost two columns of the figure are typical of cases where the decisions based on both metrics coincide (the situation is also clear for a human as well). On the rightmost columns, the two decisions differ because the pair is rather a near duplicate and one metric is less sensitive to the variations between the two images (CLIP is less sensitive to crops and zoom while MSE is less sensitive to resolution / compression artifacts).

**Visualization of retrieval** We view some visual results for retrieval in Fig. 7..2. Even highly compressed, these indices still retrieve relevant samples from L400M.



(a) Real images retrieved with a SNIP, ViT-H-15 index on L400M.



(b) Images generated by stable diffusion retrieved with a SNIP, ViT-H-15 index on L400M.

Figure 7..2: Retrieval with SNIP indices on L400M. Left column is query and remaining are retrieved.

Chapter 8

# Conclusion

This manuscript investigated the evaluation of image generation, with a focus on quality assessment, overfitting, membership inference, and parameter efficient learning. We summarize here some of the main contributions of this thesis and then review the remaining challenges.

**Generative Evaluation** We started by examining the evaluation of generated images via precision recall curves, identifying the limitations of the Fréchet Inception Distance in measuring model quality. We also provided theoretical insight into precision recall curves introduced in [148], namely their relation to the error rates of binary classifiers. For several practical settings, our method better reflected the real discrepancies found between real and generated datasets.

We gave an alternative perspective to GAN evaluation by examining memorization in Chapter 3, demonstrating how latent recovery could identify instances of memorized training points. We presented several applications using latent recovery, such as image inpainting and super resolution. Future work could further explore applications, for instance by doing latent recovery on recent text-to-image GAN systems [77], using one of the new latent recovery methods in the literature [4].

A promising future direction could be made by making a link between these two chapters. Namely, the first chapter takes a statistical view of generated and real distributions. Our latent recovery procedure only optimizes for reconstruction error. Indeed, with a perfect recovery procedure, images may be recovered but not with high likelihood. Thus, it we be natural to use a gradient of a likelihood classifier to recover points that are both high density and with low recovery error. In addition, one could also guide recovery by promoting points that are also high density in latent space.

**Membership Inference** Our study of memorization naturally led to the study of membership attacks. We studied membership inference starting from Chapter 4 to Chapter 5. In Chapter 4, we investigated a variety of training setups, and showed that many GAN generators are naturally robust versus membership inference. We used this fact to generate surrogate data, which due to the high quality of generation still retained utility as a drop in for training on several vision tasks. In Chapter 6, we introduced a new type of membership attack, capable of inferring identities used during training under challenging conditions not covered by previous state of the art.

**Parameter Efficient Learning** In Chapter 5, we explored transfer learning with few parameters and few training samples. Our width wise sharing procedure was more efficient than several other methods in the literature, in part due to its ability to spread capacity across the generator, rather than in specific layers. Finally, we drew ties with Chapter 4, by showing that training with few parameters offers robustness to membership attacks in the low data setting, whilst retaining decent quality.

**State of the Art Extraction Attack** As our final and perhaps strongest contribution, we presented a state-of-the-art extraction attack versus several widely used diffusion models. As a side contribution, we

are the first to demonstrate an extremely high level of duplication on LAION-2B. Specifically, we found roughly 700M duplicated images. We used this de-duplication to perform our extraction attack, and we demonstrated an attack on par with a recent one, whilst using thousands of times less compute.

**Future Directions** We recall the challenges we laid out at the beginning of the thesis, namely construction of more efficient membership attacks (C1), models that are robust versus said attacks (C2) and finally learning in a parameter and sample efficient way (C3). We reflect on several of our contributions, with these challenges in mind, to highlight promising future directions.

In terms of creating more general membership inference (C1), we made progress in Chapter 6 by constructing an attack that can infer properties of training data, rather than just data points. A promising future direction here is to study other facets of training data which systems like stable diffusion generate. Similar to how [60] studies whether CLIP contains identity information, a similar study could be performed with generative models. As we referenced in the introduction, generative models are largely used to generate living artists styles. Thus, an attack which is able to discern artistic styles used during training would be an interesting future direction; this would serve as a way to verify if your content was used for training, when the exact training set is unknown.

For training with low data (C3), we see potential in building systems that provide attribution. If efficient enough, a general purpose model which is trained on a large corpus of copyright free or non-sensitive data could be fine tuned, using either the methods presented in Chapter 5, or some of the more popular techniques like LoRA[61]. Then, when generation with a specific style, or with respect to a set of copyright images is desired, the large and small models could be used in tandem to generate the desired content. The attribution would then come from the training set of the domain specific weights.

Our extraction attack also makes progress towards (C1). Namely, the previous best attack required thousands of times more network evaluations whilst performing similarly to ours. We also unveiled a new type of vulnerability, namely template verbatims, which pose largely the same problems as exactly copied samples. Still, the attack has major short comings which prevents study of a more complete picture of overfitting for diffusion models. Namely, we witnessed copied training patches via inspection, similar to the observations in [161], and many samples which were slightly distorted or re-scaled and thus could not be automatically labeled. A potentially promising future direction is to design a similar attack, but labeling generations as copied if they contain patches, in any image location and with potential scale changes or other non-semantic transformations. Of course, this would be a technical challenge, as it would require object detection and segmentation to rule out trivial collisions like white backgrounds. Finally, Chapter 7 makes progress towards building more robust models (C2), as we showed which characteristics of the data lead to vulnerabilities to attack.

As generative models become larger, more general and more widely used, they have a large potential to cause harm in society. We underlined privacy and copyright as two primary areas where generative models can potentially have negative impacts. This manuscript made progress towards diagnosing and remedying these issues for building safer and ultimately more useful generation systems.

**Evaluating Deep Image Generation through the Lens of Utility and Privacy**

**Keywords:** Image Generation, Artificial Intelligence, Generative Adversarial Networks, Diffusion Models, Privacy in Machine Learning, Membership Inference Attacks.

**Summary:**
Modern image generators, such as stable diffusion or Midjourney, have become large scale, complex and general systems. As the wide spread application and use of these systems grow, so do their potential problem areas. In this thesis we investigate how generative models can leak information about their training data and the problems that poses to both the systems and the users. Systems such as Midjourney are trained with data gathered from the web and protected content can appear during generation without attribution. As generative models also have widespread application in the medical domain, it's imperative for the utility of the generative model to not generate data with strict privacy protections. We present the automatic evaluation of generative models, with a focus on these issues. We first present several statistical measures that can measure the image quality of deep generators, the diversity of generated images and finally measure their ability to overfit training samples. For the rest of the thesis, we study the problem of membership inference. We investigate a diverse set of factors that lead to vulnerability to membership attacks. On the other hand, we also observe many training setups which lead to robustness and empirical privacy. We present several new membership attacks that made improvements over the state of the art. Finally, we present a state of the art data extraction attack, capable of reconstructing training images from the most widely used generation systems.

**Évaluation de la Génération Profonde d'Images à travers le Prisme de l'Utilité et de la Confidentialité**

**Mots clefs:** Génération d'Images, Intelligence Artificielle, Réseaux Antagonistes Génératifs, Modèles de Diffusion, Confidentialité en Apprentissage Automatique, Attaques d'Inférence d'Appartenance.

**Résumé:**
Les générateurs d'images modernes, tels que Stable Diffusion ou Midjourney, sont devenus des systèmes à grande échelle, complexes et généraux. À mesure que l'application et l'utilisation de ces systèmes se généralisent, leurs éventuels problèmes se multiplient. Dans cette thèse, nous étudions comment les modèles génératifs peuvent fuiter des informations sur leurs données d'entraînement et les problèmes que cela pose à la fois aux systèmes et aux utilisateurs. Des systèmes comme Midjourney sont entrainés avec des données collectées sur le web et des contenus protégés peuvent apparaître pendant la génération sans notification d'attribution. Comme les modèles génératifs ont également une application répandue dans le domaine médical, il est impératif pour l'utilité du modèle génératif de ne pas générer de données sous protection stricte de la vie privée. Nous présentons l'évaluation automatique des modèles génératifs, avec un accent sur ces problèmes. Nous présentons d'abord plusieurs mesures statistiques qui peuvent mesurer la qualité des images produites par de tels générateurs profonds, leur diversité et enfin mesurer leur capacité à surapprendre les échantillons d'entraînement. Pour le reste de la thèse, nous étudions le problème de l'inférence d'appartenance. Nous étudions un ensemble divers de facteurs qui conduisent à la vulnérabilité aux attaques d'appartenance. D'un autre côté, nous observons également de nombreuses configurations d'entraînement qui assurent empiriquement la robustesse et la confidentialité. Nous présentons plusieurs nouvelles attaques d'appartenance permettant des améliorations par rapport à l'état de l'art. Enfin, nous présentons une attaque de pointe pour l'extraction de données, capable de reconstruire des images d'entraînement à partir des systèmes de génération les plus largement utilisés.

# Bibliography


[1] A. Abbas, K. Tirumala, D. Simig, S. Ganguli, and A. S. Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication, 2023.

[2] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space?, 2019.

[3] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan++: How to edit the embedded images?, 2020.

[4] Y. Alaluf, O. Tov, R. Mokady, R. Gal, and A. H. Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing, 2022.

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[6] R. Anadol. Refik anadol on ai, algorithms, and the machine as witness. https://www.moma.org/magazine/articles/821, 2022.

[7] AnthropicAI. Core views on ai safety: When, why, what, and how. https://www.anthropic.com/index/core-views-on-ai-safety, 2022.

[8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[9] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[10] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 224–232. JMLR.org, 2017.

[11] S. Arora, A. Risteski, and Y. Zhang. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.

[12] A. Babenko and V. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2055–2063, 2016.

[13] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG)*, 28(3):24, 2009.

[14] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. In *Proceedings of the International Conference Computer Vision (ICCV)*, 2019.

[15] D. Berthelot, T. Schumm, and L. Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[16] M. Bertran, N. Martinez, A. Papadaki, Q. Qiu, M. Rodrigues, and G. Sapiro. Learning to Collaborate for User-Controlled Privacy. *arXiv:1805.07410 [cs, stat]*, May 2018.

[17] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam. Optimizing the latent space of generative networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 600–609, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[18] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. Optimizing the latent space of generative networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 599–608, 2018.

[19] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed sensing using generative models. *Thirty-fifth International Conference on Machine Learning (ICML)*, 2018.

[20] A. Borji. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018.

[21] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[22] M. Byeon, B. Park, H. Kim, S. Lee, W. Baek, and S. Kim. Coyo-700m: Image-text pair dataset. https://github.com/kakaobrain/coyo-dataset, 2022.

[23] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[24] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramèr, B. Balle, D. Ippolito, and E. Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

[25] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 267–284, 2019.

[26] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models, 2021.

[27] A. Chen. Three threats posed by deepfakes that technology won't solve. *MIT Technology Review*, October 2019.

[28] D. Chen, N. Yu, Y. Zhang, and M. Fritz. GAN-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2020.

[29] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[30] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[31] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot. Label-only membership inference attacks. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1964–1974. PMLR, 18–24 Jul 2021.

[32] CivitAI. CivitAI, 2023.

[33] Y. Cong, M. Zhao, J. Li, S. Wang, and L. Carin. Gan memory with no forgetting. *arXiv preprint arXiv:2006.07543*, 2020.

[34] R. Dahl, M. Norouzi, and J. Shlens. Pixel recursive super resolution. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5449–5458, Oct 2017.

[35] G. Daras, J. Dean, A. Jalal, and A. G. Dimakis. Intermediate layer optimization for inverse problems using deep generative models, 2021.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[37] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021.

[38] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

[39] J. Duan, F. Kong, S. Wang, X. Shi, and K. Xu. Are diffusion models vulnerable to membership inference attacks?, 2023.

[40] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. *ICLR*, 2017.

[41] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. 2014.

[42] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision (ICCV'99)*, pages 1033–1038, Corfu, Greece, September 1999. IEEE.

[43] B. D. et al. DALL-E Mini.

[44] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[45] Q. Feng, C. Guo, F. Benitez-Quiroz, and A. M. Martinez. When do gans replicate? on the choice of dataset size. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6701–6710, October 2021.

[46] G. Franceschelli and M. Musolesi. Copyright in generative deep learning, 2021.

[47] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[48] M. Fredrikson, S. Jha, and T. Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pages 1322–1333, Denver, Colorado, USA, 2015. ACM Press.

[49] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *ArXiv*, abs/2208.01618, 2022.

[50] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):744–755, 2013.

[51] S. Ghalebikesabi, L. Berrada, S. Gowal, I. Ktena, R. Stanforth, J. Hayes, S. De, S. L. Smith, O. Wiles, and B. Balle. Differentially private diffusion models generate useful synthetic images, 2023.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[53] U. governement. U.s. copyright office fair use index. https://www.copyright.gov/fair-use/, 2023.

[54] I. Gulrajani, C. Raffel, and L. Metz. Towards GAN benchmarks which require generalization. In *International Conference on Learning Representations*, 2019.

[55] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro. Logan: Membership inference attacks against generative models, 2018.

[56] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro. LOGAN: Membership Inference Attacks Against Generative Models. *PoPETs*, 2019(1):133–152, 2019.

[57] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019.

[58] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 229–238. ACM, 1995.

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

[60] D. Hintersdorf, L. Struppek, M. Brack, F. Friedrich, P. Schramowski, and K. Kersting. Does clip know my face?, 2023.

[61] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.

[62] H. Hu and J. Pang. Membership inference of diffusion models, 2023.

[63] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[64] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018.

[65] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017.

[66] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, 36(4):107:1–107:14, 2017.

[67] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt. Openclip repository. July 2021.

[68] D. J. Im, A. H. Ma, G. W. Taylor, and K. Branson. Quantitatively evaluating GANs with divergences proposed for training. In *International Conference on Learning Representations*, 2018.

[69] D. J. Im, H. Ma, G. Taylor, and K. Branson. Quantitatively evaluating gans with divergences proposed for training. *arXiv preprint arXiv:1803.01045*, 2018.

[70] O. Jafari, P. Maurya, P. Nagarkar, K. M. Islam, and C. Crushev. A survey on locality sensitive hashing algorithms and their applications. *arXiv preprint arXiv:2102.08942*, 2021.

[71] M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd?, 2020.

[72] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[73] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[74] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[75] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7:535–547, 2021.

[76] J. Jordon, J. Yoon, and M. van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[77] M. Kang, J.-Y. Zhu, R. Zhang, J. Park, E. Shechtman, S. Paris, and T. Park. Scaling up gans for text-to-image synthesis, 2023.

[78] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[79] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[80] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33, 2020.

[81] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020.

[82] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[83] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

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

[85] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882, 2016.

[86] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

[87] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[88] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.

[89] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. *arXiv preprint arXiv:1904.06991*, 2019.

[90] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*, 2018.

[91] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[92] X. Li, L. Chang, and X. Liu. CE-Dedup: Cost-effective convolutional neural nets training based on image deduplication. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), New York City, NY, USA, September 30 - Oct. 3, 2021*, pages 11–18. IEEE, 2021.

[93] X. Li, L. Chang, and X. Liu. QHash: An efficient hashing algorithm for low-variance image deduplication. In *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 9–15, 2021.

[94] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[95] Z. Lin, A. Khetan, G. Fanti, and S. Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1498–1507, 2018.

[96] Z. C. Lipton and S. Tripathi. Precise recovery of latent vectors from generative adversarial networks. *ICLR*, 2017.

[97] K. S. Liu, B. Li, and J. Gao. Performing Co-Membership Attacks Against Deep Generative Models. *arXiv:1805.09898 [cs, stat]*, May 2018.

[98] Y. Liu, Z. Zhao, M. Backes, and Y. Zhang. Membership inference attacks by exploiting loss trajectory, 2022.

[99] Z. Liu, P. Luo, X. Wang, and X. Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15:2018, 2018.

[100] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen. Understanding Membership Inferences on Well-Generalized Learning Models. *arXiv:1802.04889 [cs, stat]*, Feb. 2018.

[101] D. Lopez-Paz and M. Oquab. Revisiting classifier two-sample tests. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[102] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 698–707. Curran Associates, Inc., 2018.

[103] M. Lučić, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[104] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[105] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[106] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[107] T. Matsumoto, T. Miura, and N. Yanai. Membership inference attacks against diffusion models, 2023.

[108] C. Meehan, K. Chaudhuri, and S. Dasgupta. A non-parametric test to detect data-copying in generative models, 2020.

[109] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Inference Attacks Against Collaborative Learning. *arXiv:1805.04049 [cs]*, May 2018.

[110] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *arXiv:1805.04049 [Cs]*, 2019.

[111] A. Menon and C. S. Ong. Linking losses for density ratio and class-probability estimation. In *International Conference on Machine Learning*, pages 304–313, 2016.

[112] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge?, 2018.

[113] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, pages 3478–3487, 2018.

[114] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. 2017.

[115] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *ICLR*, 2017.

[116] Midjourney. Midjourney, 2023.

[117] V. Mirjalili, S. Raschka, A. M. Namboodiri, and A. Ross. Semi-adversarial Networks: Convolutional Autoencoders for Imparting Privacy to Face Images. In *2018 International Conference on Biometrics, ICB 2018, Gold Coast, Australia, February 20-23, 2018*, pages 82–89, 2018.

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

[119] S. Mo, M. Cho, and J. Shin. Freeze discriminator: A simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

[120] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019.

[121] M. Nasr, R. Shokri, and A. Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 634–646, 2018.

[122] M. Nasr, R. Shokri, and A. Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 634–646. ACM, 2018.

[123] N. W. Netanel. Making sense of fair use. *Lewis & Clark Law Review*, 15, 2011.

[124] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[125] M. Novak. Getty images sues ai company over hideous frankenphotos. https://www.forbes.com/sites/mattnovak/2023/02/06/getty-images-sues-ai-company-over-hideous-frankenphotos/?sh=5da9fe6240b2, 2023.

[126] OpenAI. Pre-training Mitigations for DALLE2.

[127] Opensea. Opensea nft marketplace. https://opensea.io/.

[128] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[129] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman. Towards the science of security and privacy in machine learning. *3rd IEEE European Symposium on Security and Privacy*, 2018.

[130] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Úlfar Erlingsson. Scalable private learning with pate, 2018.

[131] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[132] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[133] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. 2015.

[134] S. Pascual, A. Bonafonte, and J. Serra. Segan: Speech enhancement generative adversarial network. *InterSpeech*, 2017.

[135] B. Piccoli, F. Rossi, and M. Tournus. A norm for signed measures. application to non local transport equation with source term. 2017.

[136] E. Pizzi, S. D. Roy, S. N. Ravindra, P. Goyal, and M. Douze. A self-supervised descriptor for image copy detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14532–14542, 2022.

[137] J. Portilla and E. P. Simoncelli. Parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[138] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

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

[140] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[141] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[142] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[143] A. Rezaei, C. Xiao, J. Gao, and B. Li. Protecting Sensitive Attributes via Generative Adversarial Networks. *arXiv:1812.10193 [cs, stat]*, Dec. 2018.

[144] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[145] K. Roose. An a.i.-generated picture won an art prize. artists aren't happy. https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html, 2022.

[146] A. Sablayrolles, M. Douze, Y. Ollivier, C. Schmid, and H. Jégou. White-box vs Black-box: Bayes Optimal Strategies for Membership Inference. In *ICML 2019 - 36th International Conference on Machine Learning*, Long Beach, United States, June 2019.

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

[148] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5234–5243. Curran Associates, Inc., 2018.

[149] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[150] M. Schneider and S.-F. Chang. A robust content based digital signature for image authentication. In *Proceedings 1996 International Conference on Image Processing*, pages 227–230, Lausanne, Switzerland, 1996. IEEE Computer Society.

[151] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[152] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[153] Y. Shen, P. Luo, J. Yan, X. Wang, and X. Tang. Faceid-gan: Learning a symmetry three-player gan for identity-preserving face synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2018.

[154] R. Shokri and V. Shmatikov. Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1310–1321, 2015.

[155] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models, 2017.

[156] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 3–18, 2017.

[157] A. Shonenkov et al. Deep image floyd. https://github.com/deep-floyd/IF, 2023.

[158] L. Simon, R. Webster, and J. Rabin. Revisiting precision recall definition for generative modeling. In *International Conference on Machine Learning*, pages 5799–5808, 2019.

[159] R. Siry, R. Webster, L. Simon, and J. Rabin. On the theoretical equivalence of several trade-off curves assessing statistical proximity, 2022.

[160] J. Sokolic, Q. Qiu, M. R. D. Rodrigues, and G. Sapiro. Learning to Succeed while Teaching to Fail: Privacy in Closed Machine Learning Systems. *arXiv:1705.08197 [cs, stat]*, May 2017.

[161] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*, 2022.

[162] C. Song, T. Ristenpart, and V. Shmatikov. Machine Learning Models that Remember Too Much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 587–601, 2017.

[163] StabilityAI. Stable diffusion 2.0. https://github.com/Stability-AI/stablediffusion, 2022.

[164] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[165] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[166] T. Tate. Defintion of appropriation. https://www.tate.org.uk/art/art-terms/a/appropriation, 2023.

[167] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[168] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016.

[169] A. Torfi, E. A. Fox, and C. K. Reddy. Differentially private synthetic medical data generation using convolutional gans. *Information Sciences*, 586:485–500, 2022.

[170] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or. Designing an encoder for stylegan image manipulation. *arXiv preprint arXiv:2102.02766*, 2021.

[171] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2018.

[172] H. Uzunova, J. Ehrhardt, F. Jacob, A. Frydrychowicz, and H. Handels. Multi-scale gans for memory-efficient generation of high resolution medical images, 2019.

[173] T. van Erven and P. Harremoës. Rényi divergence and majorization, 2010.

[174] M. Wang and W. Deng. Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655*, 2018.

[175] Y. Wang, A. Gonzalez-Garcia, D. Berga, L. Herranz, F. S. Khan, and J. v. d. Weijer. Minegan: Effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[176] R. Webster, J. Rabin, L. Simon, and F. Jurie. Detecting Overfitting of Deep Generative Networks via Latent Recovery. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[177] R. Webster, J. Rabin, L. Simon, and F. Jurie. Detecting overfitting of deep generative networks via latent recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11273–11282, 2019.

[178] R. Webster, J. Rabin, L. Simon, and F. Jurie. On the de-duplication of laion-2b, 2023.

[179] K. Wiggers. Vcs continue to pour dollars into generative ai. https://techcrunch.com/2023/03/28/generative-ai-venture-capital, 2023.

[180] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems*, 31:5962–5972, 2018.

[181] Z. Wu, Z. Wang, Z. Wang, and H. Jin. Towards Privacy-Preserving Visual Recognition via Adversarial Training: A Pilot Study. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, pages 627–645, 2018.

[182] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[183] C. Yang, Y. Shen, Y. Xu, and B. Zhou. Data-efficient instance generation from instance discrimination, 2021.

[184] R. A. Yeh, C. Chen, T.-Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. In *CVPR*, volume 2, page 4, 2017.

[185] X. Yi, E. Walia, and P. Babyn. Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58:101552, 2019.

[186] J. Yoon, J. Jordon, and M. van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

[187] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018.

[188] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[189] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv:1611.03530 [cs]*, Nov. 2016.

[190] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[191] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[192] Y. Zhang, A. Azad, and Z. Hu. Fastsv: A distributed-memory connected component algorithm with fast convergence, 2020.

[193] Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[194] M. Zhao, Y. Cong, and L. Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning*, pages 11340–11351. PMLR, 2020.

[195] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[196] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33, 2020.

[197] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.