ABSTRACT

Title of Dissertation: HUMAN-CENTRIC
DEEP GENERATIVE MODELS:
THE BLESSING AND THE CURSE

Ning Yu
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Larry Davis
Department of Computer Science

Over the past years, deep neural networks have achieved significant progress in a wide range of real-world applications. In particular, my research puts a focused lens in deep generative models, a neural network solution that proves effective in visual (re)creation. But is generative modeling a niche topic that should be researched on its own? My answer is critically no. In the thesis, I present the two sides of deep generative models, their blessing and their curse to human beings. Regarding what can deep generative models do for us, I demonstrate the improvement in **performance** and **steerability** of visual (re)creation. Regarding what can we do for deep generative models, my answer is to mitigate the **security** concerns of DeepFakes and improve minority **inclusion** of deep generative models.

For the **performance** of deep generative models, I probe on applying attention modules and dual contrastive loss to generative adversarial networks (GANs), which pushes photorealistic image generation to a new state of the art. For the **steerability**, I introduce Texture Mixer, a simple yet effective approach to achieve steerable texture

synthesis and blending. For the **security**, my research spans over a series of GAN fingerprinting solutions that enable the detection and attribution of GAN-generated image misuse. For the **inclusion**, I investigate the biased misbehavior of generative models and present my solution in enhancing the minority inclusion of GAN models over underrepresented image attributes. All in all, I propose to project actionable insights to the applications of deep generative models, and finally contribute to human-generator interaction.

HUMAN-CENTRIC DEEP GENERATIVE MODELS: THE BLESSING AND THE CURSE

by

Ning Yu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Larry Davis, Chair/Advisor
Professor Joseph JaJa, Dean's Representative
Professor David Jacobs
Professor Matthias Zwicker
Professor Abhinav Shrivastava

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost, I would like to thank my advisors, Larry Davis at the University of Maryland (UMD) and Mario Fritz at Max Planck Institute for Informatics (MPI), for giving me an invaluable opportunity to work on challenging and visionary projects over the past years. They have always made themselves available for advice and recommendations. Without their support, I would not be who I am today.

I would like to thank Joseph JaJa, David Jacobs, Matthias Zwicker, and Abhinav Shrivastava for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript and attending my presentation.

I would also like to thank my internship mentors at NVIDIA, Guilin Liu, Aysegul Dundar, Andrew Tao, and Bryan Catanzaro, as well as my internship mentors at Adobe, Connelly Barnes, Eli Shechtman, Sohrab Amirghodsi, Michal Lukáč, Xiaohui Shen, Zhe Lin, and Radomír Měch, for their insights and expertise that have inspired my research ideas and have calibrated my experimental designs. They have ignited my passion to transfer techniques from theoretical publications to practical products.

My publication collaborators have enriched my academic achievements and have helped complete my understanding of the domain knowledge. They deserve a special

mention: Ke Li, Peng Zhou, Jitendra Malik, Dingfan Chen, Yang Zhang, Saurabh Sharma, Bernt Schiele, Hui-Po Wang, Yang He, Margret Keuper, Zuxuan Wu, Ser-Nam Lim, Vladislav Skripniuk, Sahar Abdelnabi, and more. It has been a pleasure to work with and learn from such extraordinary individuals.

I would like to acknowledge financial support, coding sharing, and constructive advice for all my research projects. They are articulated in each of the following chapters.

I owe my deepest thanks to my family - my mother, father, and my girlfriend Siyi Wang, who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Since the revolutionary technique of generative adversarial networks (GANs) [9] was invented seven years ago, its successive breakthroughs have demonstrated stunning performance in generating photorealistic images [10, 11, 12, 13, 14, 15, 16, 17]. The progress is mainly driven by large-scale datasets [16, 18, 19, 20, 21, 22], architectural tuning [16, 17, 23, 24], and loss designs [12, 13, 25, 26, 27, 28]. GAN techniques have been popularized into extensive computer vision applications, including but not limited to image translation [22, 29, 30, 31, 32, 33, 34, 35, 36], postprocessing [37, 38, 39, 40, 41, 42, 43], image manipulation [44, 45, 46, 47], texture synthesis [48, 49], image inpainting [5, 50, 51, 52], and text-to-image generation [53, 54, 55, 56]. The recent Adobe Neural Filter library[1] pioneers the commercialization of GANs.

We regard these as the **blessing** of GANs. Yet generated images are still easy to spot especially on datasets with high variance (e.g. bedroom, church). Therefore, in Chapter 2 we propose various improvements to further push the boundaries in image generation. Specifically, we propose a novel dual contrastive loss and show that, with this loss, discriminator learns more generalized and distinguishable representations to incentivize generation. In addition, we revisit attention and extensively experiment with

[1] `https://helpx.adobe.com/photoshop/using/neural-filters.html`

different attention blocks in the generator. We find attention to be still an important module for successful image generation even though it was not used in the recent state-of-the-art models. Lastly, we study different attention architectures in the discriminator, and propose a reference attention mechanism. By combining the strengths of these remedies, we improve the compelling state-of-the-art Fréchet Inception Distance (FID) by at least 17.5% on several benchmark datasets. We obtain even more significant improvements on compositional synthetic scenes (up to 47.5% in FID).

With the improvement of GAN performance witnessed, we specify in Chapter 3 its steerability in Texture Mixer, a controllable texture interpolation pipeline. This study addresses the problem of interpolating visual textures. We formulate this problem by requiring (1) by-example controllability and (2) realistic and smooth interpolation among an arbitrary number of texture samples. To solve it we propose a neural network trained simultaneously on a reconstruction task and a generative adversarial task, which can project texture examples onto a latent space where they can be linearly interpolated and projected back onto the image domain, thus ensuring both intuitive control and realistic results. We show our method outperforms several baselines according to a comprehensive suite of metrics as well as a user study. We further show several applications based on our technique, which include texture brush, texture dissolve, and animal hybridization. Demos, videos, code, data, models, and supplemental material are available at GitHub[2].

However, a coin has two sides. Despite plenty of use cases of the GAN technique, a flood of strong concerns arise from its **curse**. Given the level of realism and diversity that GANs can achieve today, detecting generated media, well known as deepfakes, attributing

[2] `https://github.com/ningyu1991/TextureMixer.git`

their sources, and tracing their legal responsibilities become infeasible to human beings. Moreover, the misuse of deepfakes has been permeating to each corner of social media, ranging from misinformation of political campaigns[3] to fake journalism[4] [5]. Therefore, it is critical to look into effective visual forensics against threats from GANs. As responses, we propose a series of GAN fingerprinting solutions that enable the detection and attribution of GAN-generated image misuse.

In Chapter 4, We present the first study of learning GAN fingerprints towards image attribution and using them to classify an image as real or GAN-generated. For GAN-generated images, we further identify their sources. Our experiments show that (1) GANs carry distinct model fingerprints and leave stable fingerprints in their generated images, which support image attribution; (2) even minor differences in GAN training can result in different fingerprints, which enables fine-grained model authentication; (3) fingerprints persist across different image frequencies and patches and are not biased by GAN artifacts; (4) fingerprint finetuning is effective in immunizing against five types of adversarial image perturbations; and (5) comparisons also show our learned fingerprints consistently outperform several baselines in a variety of setups. Code, data, models, and supplementary material are available at GitHub[6].

While the above work on deepfake detection demonstrates high accuracy, it is subject to advances in generation techniques and adversarial iterations on detection countermeasure techniques. Thus, in Chapter 5 we seek a proactive and sustainable solution on deepfake

[3] https://www.technologyreview.com/2020/02/19/868173/an-indian-politician-is-using-deepfakes-to-try-and-win-voters

[4] https://www.theverge.com/2020/7/7/21315861

[5] https://futurism.com/the-byte/deepfake-fake-journalist

[6] https://github.com/ningyu1991/GANFingerprints.git

detection, that is agnostic to the evolution of generative models, by introducing artificial fingerprints into the models. Our approach is simple and effective. We first embed artificial fingerprints into training data, then validate a surprising discovery on the transferability of such fingerprints from training data to generative models, which in turn appears in the generated deepfakes. Experiments show that our fingerprinting solution (1) holds for a variety of cutting-edge generative models, (2) leads to a negligible side effect on generation quality, (3) stays robust against image-level and model-level perturbations, (4) stays hard to be detected by adversaries, and (5) converts deepfake detection and attribution into trivial tasks and outperforms the recent state-of-the-art baselines. Our solution closes the responsibility loop between publishing pre-trained generative model inventions and their possible misuses, which makes it independent of the current arms race.

In addition, we propose to improve the efficiency and scalability of proactive GAN fingerprinting in Chapter 6. Our technique achieves this by ad-hoc generating a large population of models with distinct fingerprints. Our recommended operation point uses a 128-bit fingerprint which in principle results in more than 10^{36} identifiable models. Experiments show that our method fulfills key properties of a fingerprinting mechanism and achieves effectiveness in deep fake detection and attribution. As a result, our work enables a responsible disclosure of state-of-the-art generative models, that allows researchers and companies to fingerprint their models, so that the generated samples containing a fingerprint can be accurately detected and attributed to a source.

Another thread of concerns on the **curse** of GANs comes from the fact that there could be potential biases in the learned model against underrepresented data subgroups [57,

58, 59, 60, 61]. The biases are rooted in the inevitable imbalance in the dataset [62], which are even exacerbated by the GANs [58]. This is because GANs can implicitly disregard infrequent images due to the well-established problem of mode collapse [63, 64], thereby further introducing model biases on top of data biases. This issue is particularly acute from the perspective of minority inclusion, because training data associated with minority subgroups by definition do not form dominant modes. Consequently, data from minority groups are rare to begin with, and would not be capable of being produced by the generative model at all due to mode collapse.

As a response, in Chapter 7 we propose Inclusive GAN, the first study to formalize the problem of minority inclusion in GANs as one of data coverage, and then propose to improve data coverage by harmonizing adversarial training with reconstructive generation. The experiments show that our method outperforms the existing state-of-the-art methods in terms of data coverage on both seen and unseen data. We develop an extension that allows explicit control over the minority subgroups that the model should ensure to include, and validate its effectiveness at little compromise from the overall performance on the entire dataset. Code, models, and supplemental videos are available at GitHub[7].

[7] https://github.com/ningyu1991/InclusiveGAN.git

Chapter 2: Dual Contrastive Loss and Attention for GANs

2.1 Introduction

Photorealistic image generation has increasingly become reality, benefiting from the invention of generative adversarial networks (GANs) [9] and its successive breakthroughs [10, 13, 14, 15, 16, 17, 65, 66]. The progress is mainly driven by large-scale datasets [16, 18, 19, 20, 21, 22], architectural tuning [16, 17, 23, 24, 67], and loss designs [13, 25, 26, 27, 28, 65, 66, 68, 69, 70, 71]. GAN techniques have been popularized into extensive computer vision applications, including but not limited to image translation [22, 29, 30, 31, 32, 33, 34, 35, 36], postprocessing [37, 38, 39, 40, 41, 42, 43], image manipulation [44, 45, 46, 47], texture synthesis [48, 49, 72], image inpainting [5, 50, 51, 52], and text-to-image generation [53, 54, 55, 56].

Yet, behind the seemingly saturated performance of the state-of-the-art StyleGAN2 [17], there still persists open issues of GANs that make generated images surprisingly obvious to spot [1, 73, 74]. Hence, it is still necessary to revisit the fundamental generation power when other concurrent deep learning techniques keep advancing and creating space for GAN improvements.

We investigate methods to improve GANs in two dimensions. In the first dimension, we work on the loss function. As the discriminator aims to model the intractable real

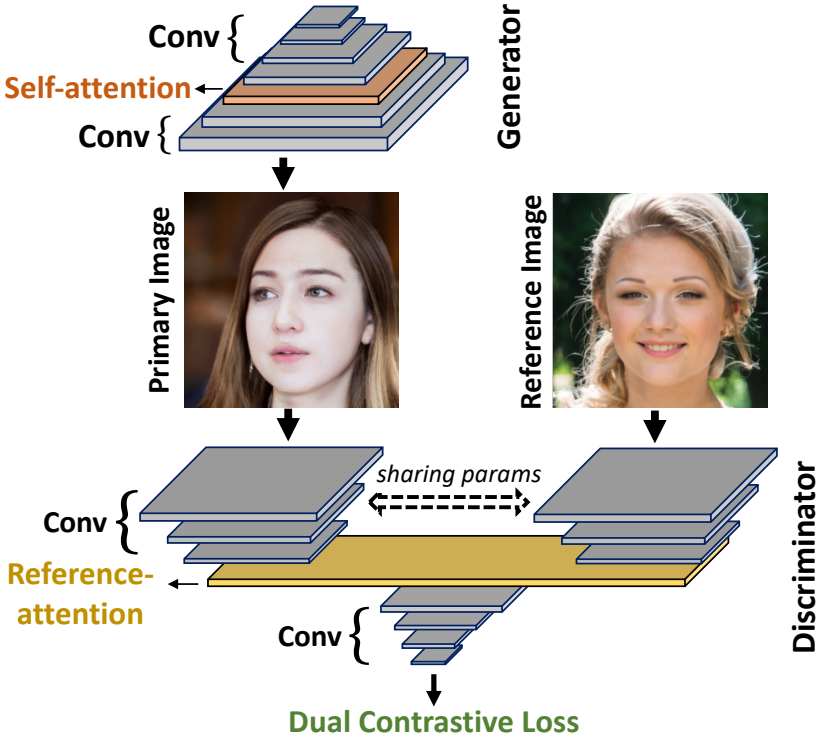


Figure 2.1: The diagram of our GAN framework using three key components: self-attention in the generator, reference-attention in the discriminator, and a novel dual contrastive loss. Technical diagrams are in Fig. 2.2 and 2.4.

data distribution via a workaround of real/fake binary classification, a more effective discriminator can back-propagate more meaningful signals for the generator to compete against. However, the feature representations of discriminators are often not generalized enough to incentivize the adversarially evolving generator and are prone to forgetting previous tasks [75] or previous data modes [63, 64]. This often leads to the generated samples with discontinued semantic structures [23, 76] or the generated distribution with mode collapse [28, 63]. To mitigate this issue, we propose to synergize generative modeling with the advancements in contrastive learning [77, 78]. In this direction, for the first time, we replace the logistic loss of StyleGAN2 with a newly designed dual contrastive loss.

In the second dimension, we revisit the architecture of both generator and discriminator networks. Specifically, many GAN-based image generators rely on convolutional layers to encode features. In such design, long-range dependencies across pixels (e.g., large-size semantically correlated layouts) can only be formulated with a deep stack of convolutional layers. This, however, does not favor the stability of GAN training because of the challenge to coordinate multiple layers desirably. The minimax formulation and the alternating gradient ascent-descent in the GAN framework further exacerbate such instability. To circumvent this issue, attention mechanisms that support long-range modeling across image regions are incorporated into GAN models [14, 23]. After that, however, StyleGAN2 claimed the state of the art with a novel architectural design without any attention mechanisms. Therefore, it turns not clear whether attention still improves results, which of the popular attention mechanisms [4, 79, 80, 81] improves the most, and in return of how many additional parameters. To answer these questions, we extensively study the role of attention in the current state-of-the-art generator, and during this study improve the results significantly.

In the discriminator, we again explore the role of attention as shown in Fig. 7.1. We design a novel reference attention mechanism in the discriminator where we allow two irrelevant images as the inputs at the same time: one input is sampled from real data as a reference, and the other input is switched between a real sample and a generated sample. The two inputs are encoded through two Siamese branches [82, 83, 84, 85] and fused by a reference-attention module. In this way, we achieve to guide real/fake classification under the attention of the real world. Contributions are summarized as follow:

- We propose a novel dual contrastive loss in adversarial training that generalizes representation to more effectively distinguish between real and fake, and further incentivize the image generation quality.
- We investigate variants of the attention mechanism in GAN architecture to mitigate the local and stationary issues of convolutions.
- We design a novel reference-attention discriminator architecture that substantially benefits limited-scale datasets.
- We conduct extensive experiments on large-scale datasets and their smaller subsets. We show that our improvements on the loss function and on the generator hold in both scenarios. On the other hand, we find discriminator to behave differently based on the number of available images, and the reference-attention-based discriminator to be only improving on limited-scale datasets.
- We redefine the state of the art by improving FID scores by at least 17.5% on several large-scale benchmark datasets. We also achieve more realistic generation on the CLEVR dataset [21] which poses different challenges from the other datasets: compositional

scenes with occlusions, shadows, reflections, and mirror surfaces. It comes with 47.5% FID improvement.

2.2 Related work

Generative adversarial networks (GANs). Since the invention of GANs [9], there have been rapid progress to achieve photorealistic image generation [10, 13, 14, 15, 16, 17, 65, 66, 66]. Significant improvements are obtained by careful architectural designs for generators [16, 17, 23, 24, 67], discriminators [33, 86] and new regularization techniques [13, 25, 26, 27, 28, 65, 66, 69, 70, 71]. Architectural evolution in generators started from a multi-layer perceptron (MLP) [9] and moved to deep convolutional neural networks (DCNN) [10], to models with residual blocks [13], and recently style-based [16, 17] and attention-based [14, 23] models. Similarly, discriminators evolved from MLP to DCNN [10], however, their design has not been studied as aggressively. In this paper, we propose changes in both generators and discriminators, and for the loss function.

Contrastive learning. Contrastive learning targets a transformation of inputs into an embedding where associated signals are brought together, and they are distanced from the other samples in the dataset [78, 87, 88, 89]. The same intuition behind contrastive learning has also been the base of Siamese networks [82, 83, 84, 85]. Contrastive learning is shown to be an effective tool for unsupervised learning [77, 90, 91], conditional image synthesis [36, 69, 70], and domain adaptation [92]. In this work, we study its effectiveness when it is closely coupled with the adversarial training framework and replaces the conventional adversarial loss for unconditional image generation. It is orthogonal to the works [69, 70,

71] where the contrastive losses serve only as an incremental auxiliary to the conventional adversarial loss and require expensive class annotations or augmentation for generation.

Attention models. Attention models have dominated the language modeling [93, 94, 95, 96, 97], and became popular among various computer vision problems from image recognition [80, 98, 99, 100, 101, 102, 103, 104] to image captioning [105, 106, 107] to video prediction [79, 81]. They are proposed in various forms: spatial attention that reweights the convolution activations [23, 81, 108], in different channels [99, 100, 101], or a combination of them [107, 109, 110]. Attention models with their reweighting mechanisms provide a possibility for long-range modeling across distant image regions. As attention models outperform others in various computer vision tasks, researchers were quick to incorporate them into unconditional image generation [14, 23, 67, 111], semantic-based image generation [86, 112], and text-guided image manipulation models [113, 114]. Even though attention models have already benefited the image generation tasks, we believe the results can be further improved by empowering the state-of-the-art image synthesis models [17] (attention not involved) with the most recent achievements in the attention modules [4]. In addition, we design a novel reference-attention architecture for the discriminator and show a further boost on limited-scale datasets.

2.3 Approach

Our improvements for GANs include a novel dual contrastive loss and variants of the attention mechanisms. For each improvement, we organize the context in a combination between method formulation and experimental investigation. After validating our optimal

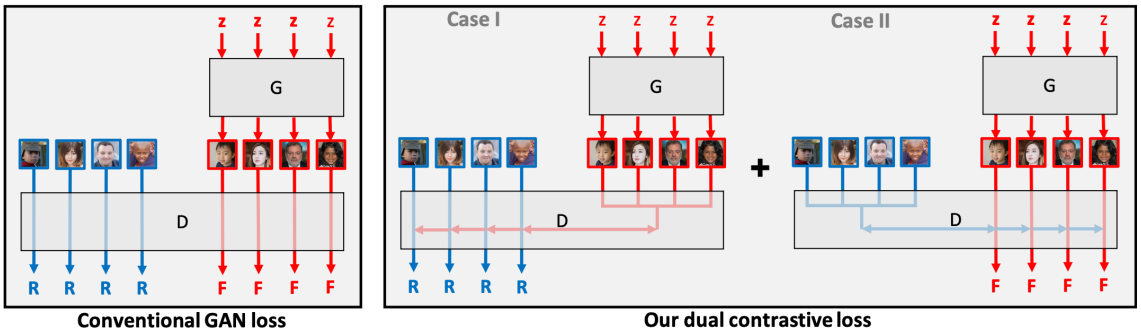


Figure 2.2: Comparisons between the diagram of conventional GAN loss and diagram of our dual contrastive loss. Our contrastive loss in Case I aims at teaching the discriminator to disassociate a single real image (R) against a batch of generated images (F). Dually in Case II, the discriminator learns to disassociate a single generated image against a batch of real images.

configuration, we compare it to the state of the art in Section 5.5.

2.3.1 Dual contrastive loss

Adversarial training relies on the discriminator's ability on real vs. fake classification. As in other classification tasks, discriminators are also prone to overfitting when the dataset size is limited [115]. On larger datasets, on the other hand, there is no study showing that disciminators overfit but we hypothesize that adversarial training can still benefit from novel loss functions which encourage the distinguishability power of the discriminator representations for their real vs. fake classification task.

We put another lens on the representation power of the discriminator by incentivizing generation via contrastive learning. Contrastive learning associates data points and their positive examples and disassociates the other points within the dataset which are referred to as negative examples. It is recently re-popularized by various unsupervised learning works [77, 78, 87, 88, 89] and generation works [36, 69, 70]. Among these works,

contrastive learning is used as an auxiliary task. For example in image to image translation task, a translator learns to output a zebra image given a horse image via adversarial loss and in addition learns to align the input horse image and the generated zebra image via contrastive loss function [36]. Contrastive loss in that work is utilized such that given a patch showing the legs of an output zebra should be strongly associated with the corresponding legs of the input horse, more so than the other patches randomly extracted from the horse image.

In this work, different from the previous ones, we do not use contrastive learning as an auxiliary task but directly couple it in the main adversarial training by a novel loss function formulation. We, to the best of our knowledge, for the first time train an unconditional GAN by solely relying on contrastive learning. As shown in Fig. 2.2 Right Case I, our contrastive loss function aims at teaching the discriminator to disassociate a single real image against a batch of generated images. Dually in Case II, the discriminator learns to disassociate a single generated image against a batch of real images. The generator adversarially learns to minimize such dual contrasts. Mathematically, we derive this loss function by extending the binary classification used in [9, 17] to a noise contrastive estimation framework [77], a one-against-a-batch classification in the softmax cross-entropy formulation. The novel formulation is as follows:

In Case I:

$$
\begin{aligned}
L_{real}^{contr}(G, D) &= \mathop{\mathbb{E}}_{\mathbf{x}\sim p(\mathbf{x})} \left[\log \frac{e^{D(\mathbf{x})}}{e^{D(\mathbf{x})} + \sum\limits_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)} e^{D(G(\mathbf{z}))}} \right] \\
&= - \mathop{\mathbb{E}}_{\mathbf{x}\sim p(\mathbf{x})} \left[\log \left(1 + \sum_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)} e^{D(G(\mathbf{z}))-D(\mathbf{x})} \right) \right]
\end{aligned}
\tag{2.1}
$$

In Case II:

$$
\begin{aligned}
L_{fake}^{contr}(G, D) &= \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)} \left[\log \frac{e^{-D(G(\mathbf{z}))}}{e^{-D(G(\mathbf{z}))} + \sum\limits_{\mathbf{x}\sim p(\mathbf{x})} e^{-D(\mathbf{x})}} \right] \\
&= - \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)} \left[\log \left(1 + \sum_{\mathbf{x}\sim p(\mathbf{x})} e^{D(G(\mathbf{z}))-D(\mathbf{x})} \right) \right]
\end{aligned}
\tag{2.2}
$$

Comparing between Eq. 2.1 and 2.2, the duality is formulated by switching the order of real/fake sampling while keeping the other calculation unchanged. Comparing to the logistic loss [9, 17], contrastive loss enriches the softplus formulation $\log(1 + e^{D(\cdot)})$ with a batch of inner terms and using discriminator logit contrasts between real and fake samples. Finally, our adversarial objective is:

$$
\min_G \max_D \; L_{real}^{contr}(G, D) + L_{fake}^{contr}(G, D) \tag{2.3}
$$

Investigation on loss designs. We extensively validate the effectiveness of dual contrastive loss compared to other loss functions as presented in Table 2.1. We replace the loss used in StyleGAN2 [17], non-saturating default loss, with other popular GAN losses while keeping all the other parameters the same. As shown in Table 2.1, dual contrastive

	FFHQ	Bedroom	Church	Horse	CLEVR
Non-saturating [9] (default)	4.86	4.01	4.54	3.91	9.62
Saturating [9]	5.16	4.26	4.80	5.90	10.46
Wasserstein [66]	7.99	6.05	6.28	7.23	**5.82**
Hinge [116]	4.14	4.92	4.39	5.27	14.87
Dual contrastive (**ours**)	**3.98**	**3.86**	**3.73**	**3.70**	6.06

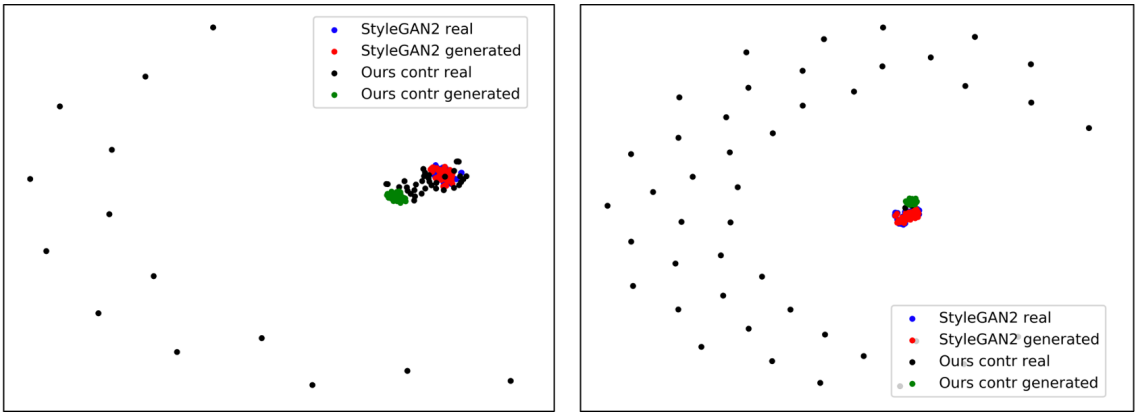
Table 2.1: Comparisons in FID among different GAN losses. Based on StyleGAN2 config E backbone, it shows our contrastive loss outperforms a variety of other losses on four out of five large-scale datasets. Wasserstein loss is better than ours on CLEVR, but are the worst on the other datasets.

Loss	FFHQ	Bedroom	Church	Horse	CLEVR
Non-saturating [9] (default)	245.	332.	517.	1285.	199.
Dual contrastive (**ours**)	**377.**	**580.**	**856.**	**1645.**	**513.**

Table 2.2: Comparisons in FDDF between StyleGAN2 default loss and our loss. A larger value is more desirable, indicating the learned discriminator features are more distinguishable between real and fake.

loss is the only loss that significantly improves upon the default loss of StyleGAN2 consistently on all the five datasets. Wasserstein loss is better than ours on CLEVR dataset, but is the worst among all the loss functions on the other datasets. We reason the success of the dual loss to its formulation that explicitly learns an unbiased representation between real and generated distributions.

The distinguishability of contrastive representation. Motivated by the consistent improvement from our dual contrastive loss, we delve deeper to investigate if and by how much our contrastive representation is more distinguishable than the original discriminator representation. We measure the representation distinguishability by the Fréchet distance of the discriminator features in the last layer (FDDF) between 50K real and generated samples. A larger value indicates more distinguishable features between real and fake.



Bedroom 256×256 **Church 256×256**

Figure 2.3: The tSNE plots for the distributions of discriminator features. The distinguishability of features based on our contrastive loss is much more significant than that based on the default non-saturating loss in StyleGAN2 baseline. Our loss learns to associate fake features to a "core" clique (green) while pushing real features in the wild outwards as "satellites" (black). The baseline loss fails to differentiate features from the two sources (red v.s. blue) with a clear margin.

We find our dual contrastive features to be consistently more distinguishable than the original discriminator features as shown in Table 2.2 and Fig. 2.3, which back-propagates more effective gradients to incentivize our generator.

2.3.2 Self-attention in the generator

The majority of the GAN-based image generators rely solely on convolutional layers to extract features [10, 13, 15, 16, 17, 65, 66], even though the local and stationary convolution primitive in the generator can not model the long-range dependencies in an image. Among recent GAN-based models, SAGAN [23] uses the self-attention block [81] and demonstrates improved results. BigGAN [14] also follows this choice and uses a similar attention module for better performance. After that, however, StyleGAN [16] and StyleGAN2 [17] redefine the state of the art with various modifications in the generator

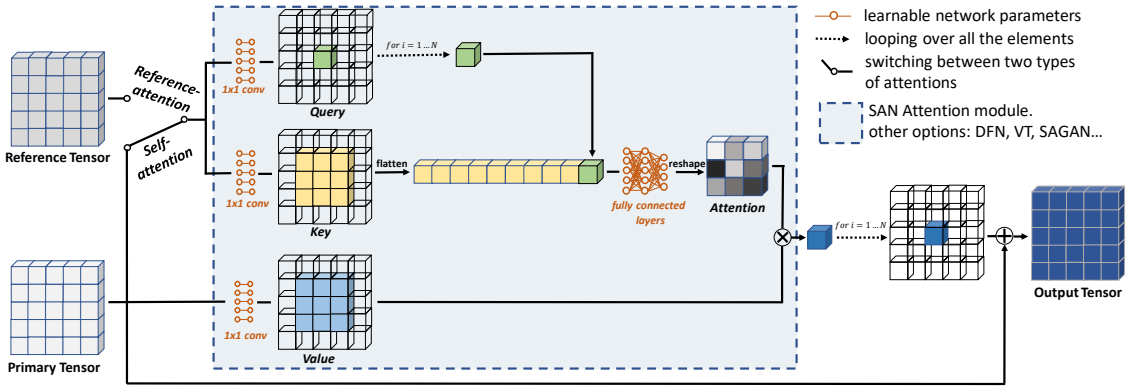


Figure 2.4: The diagram of self-attention and reference-attention schemes. The attention module is instantiated by SAN [4] but is agnostic to network backbone. It can flexibly switch to other options and be plug-and-play. We switch between the sources that are used to calculate the Key and Query tensors, so as to implement self-attention and reference-attention respectively.

architecture which do not include any attention mechanisms. StyleGAN2 also shows that generation results can be improved by larger networks with an increased number of convolution filters. Therefore, it is now not clear what the role of attention is in the state-of-the-art image generation models. Does attention still improve the network performance? Which attention mechanism benefits the most and in the trade of how many additional parameters? To answer these questions, we experiment with previously proposed self-attention modules: Dynamic Filter Networks (DFN) [79], Visual Transformers (VT) [80], Self-Attention GANs (SAGAN) [23], as well as the state-of-the-art patch-based spatially-adaptive self-attention module, SAN [4].

All the above self-attention modules are benefited from their adaptive data-dependent parameter space while they have their own hand-crafted architecture designs and interpretability. DFN [79] keeps the convolution primitive but makes the convolutional filter condition to its input tensor. VT [80] compresses input tensor to a set of 1D feature vectors, interprets them as semantic tokens, and leverages language transformer [93] for tensor propagation.

SAN [4] generalizes the self-attention block [81] (as used in SAGAN [23]) by replacing the point-wise softmax attention with a patch-wise fully-connected transformation.

We show the diagram of self-attention in Figure 2.4, with a specific instantiation from SAN [4] due to its generalized and state-of-the-art design. Note that the attention module is agnostic to network backbone and can be switched to other options for fair comparisons. For conceptual and technical completeness, we formulate our SAN-based self-attention below.

In details, let $\mathbf{T} \in \mathbb{R}^{h \times w \times c}$ be the input tensor to a convolutional layer in the original architecture. Following the mainstream protocol of self-attention calculation [23, 81, 111], we obtain the corresponding key, query, and value tensors $\mathbf{K}(\mathbf{T}), \mathbf{Q}(\mathbf{T}), \mathbf{V}(\mathbf{T}) \in \mathbb{R}^{h \times w \times c}$ separately using 1×1 convolutional kernel followed by bias and leaky ReLU. For each location (i, j) within the tensor spatial dimensions, we extract a large patch with size s from $\mathbf{K}$ centered at (i, j), denoted as $\mathbf{k} \in \mathbb{R}^{s \times s \times c}$. We then flatten the patch and concatenate it along the channel dimension with $\mathbf{q} \in \mathbb{R}^{1 \times 1 \times c}$, the query vector at (i, j), to obtain $\mathbf{p} \in \mathbb{R}^{1 \times 1 \times (s^2 c + c)}$:

$$\begin{aligned} \mathbf{k} &= \mathbf{K}_{(i-\frac{s}{2}:i+\frac{s}{2}+1,\, j-\frac{s}{2}:j+\frac{s}{2}+1)} \\ \mathbf{q} &= \mathbf{Q}_{(i,j)} \\ \mathbf{p} &= \mathsf{concat}\left(\mathsf{flatten}(\mathbf{k}), \mathbf{q}\right) \end{aligned} \tag{2.4}$$

In order to cooperate between the key and query, we feed $\mathbf{p}$ through two fully-connected layers followed by bias and leaky ReLU and obtain a vector with size $\tilde{\mathbf{w}} \in$

$\mathbb{R}^{1\times1\times s^2c}$:

$$\begin{aligned}\hat{\mathbf{w}} &= \mathsf{leakyReLU}(\mathbf{p}\mathbf{M}_{w1} + \mathbf{b}_{w1}) \\ \tilde{\mathbf{w}} &= \hat{\mathbf{w}}\mathbf{M}_{w2} + \mathbf{b}_{w2}\end{aligned} \tag{2.5}$$

$\mathbf{M}_{w1} \in \mathbb{R}^{(s^2c+c)\times s^2c}$, $\mathbf{M}_{w2} \in \mathbb{R}^{s^2c\times s^2c}$, and $\mathbf{b}_{w1}, \mathbf{b}_{w2} \in \mathbb{R}^{1\times1\times s^2c}$ are the learnable parameters in the fully connected layers and biases.

On one hand we reshape $\tilde{\mathbf{w}}$ back to the patch size $\mathbf{w} \in \mathbb{R}^{s\times s\times c}$; on the other hand we extract a patch with the same size from $\mathbf{V}$ centered at (i, j), denoted as $\mathbf{v} \in \mathbb{R}^{s\times s\times c}$. Next, we aggregate $\mathbf{v}$ over spatial dimensions with the correponding weights from $\mathbf{w}$ to derive an output vector $\mathbf{o} \in \mathbb{R}^{1\times1\times c}$:

$$\begin{aligned}\mathbf{w} &= \mathsf{reshape}(\tilde{\mathbf{w}}) \\ \mathbf{v} &= \mathbf{V}_{(i-\frac{s}{2}:i+\frac{s}{2}+1,\, j-\frac{s}{2}:j+\frac{s}{2}+1)} \\ \mathbf{o}(i,j) &= \sum_{m,n=1}^{s} \mathbf{w}_{(m,n)}\mathbf{v}_{(m,n)}\end{aligned} \tag{2.6}$$

We loop over all the (i, j) to constitute an output tensor $\bar{\mathbf{O}}^{self} \in \mathbb{R}^{h\times w\times c}$ and define it as the self-attention output. Finally, we replace the original convolution output with $\mathbf{O}^{self} \in \mathbb{R}^{h\times w\times c}$, a residual version of this self-attention output.

$$\begin{aligned}\bar{\mathbf{O}}^{self}_{(i,j)} &= \mathbf{o}(i,j), \quad \forall i = 1,\ldots,h,\ j = 1,\ldots,w \\ \bar{\mathbf{O}}^{self} &\doteq \mathsf{attn}\left(\mathbf{K}(\mathbf{T}), \mathbf{Q}(\mathbf{T}), \mathbf{V}(\mathbf{T})\right) \\ \mathbf{O}^{self} &= \bar{\mathbf{O}}^{self} + \mathbf{T}\end{aligned} \tag{2.7}$$

	CelebA	Animal Face	Bedroom	Church
StyleGAN2 [17]	9.84	36.55	19.33	11.02
+ DFN [79]	8.41	35.10	26.86	11.31
+ VT [80]	9.18	34.70	16.85	10.64
+ SAGAN [23]	9.35	34.83	17.94	10.65
+ SAN [4]	**8.60**	**32.72**	**16.36**	**9.62**

Table 2.3: Comparisons in FID among different attention modules in the generator. StyleGAN2 config E which does not include an attention module is used as a backbone. For computationally efficient comparisons, we use the 30k subset of each dataset at 128×128 resolution.

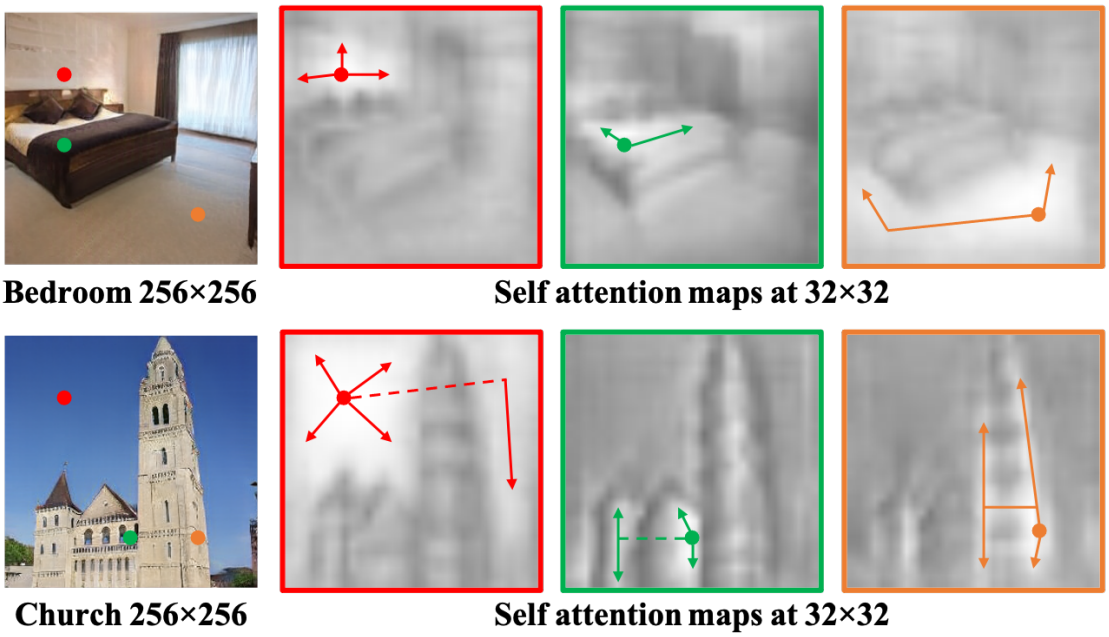


Figure 2.5: StyleGAN2 + SAN generated samples and their self-attention maps in the generator for the corresponding dot positions. Considering there is an attention weight kernel $\mathbf{w} \in \mathbb{R}^{s \times s \times c}$ for each position, we visualize the norm for each spatial position of $\mathbf{w}$. The attention maps strongly align to the semantic layout and structures of the generated images, which enable long-range dependencies across objects.

It is worth noting that $\mathbf{w}$ plays a conceptually equivalent role as the softmax attention map of the traditional key-query aggregation [23, 81, 111], except it is not identical across channels anymore but rather generalized to optimize for each channel. $\mathbf{w}$ also aligns in spirit with the concept of DFN [79], except the spatial size $s \times s$ is empirically set much larger than 3×3, and more importantly, $\mathbf{w}$ is not "sliding" anymore but rather generalized to optimize at each location.

Investigations on self-attention modules. In Table 2.3 we extensively compare among a variety of self-attention modules by replacing the default convolution in the 32×32-resolution layer in StyleGAN2 [17] config E backbone with one of them. We justify that SAN [4] significantly improves over the StyleGAN2 baseline and consistently improves for various datasets outperforming other attention variants with a clear margin.

We visualize the attention map examples of the best performing generator (StyleGAN2 + SAN) in Fig. 2.5. We find attention maps to strongly correspond to the semantic layout and structures of the generated images.

Complexity of self-attention modules. We also compare in Table 2.4 the time and space complexity of these self-attention modules. We observe that DFN [79] and VT [80] moderately improve the generation quality yet in the trade of undesirable $> 3.6\times$ complexity. On the contrary, the improvements from SAGAN [23] or SAN [4] are not at the cost of complexity, but rather benefited from the more representative attention designs. They use a fewer number of convolution channels and the multi-head trick [81] to control their complexity. These results show that the improved performance does not come from any additional parameters but rather the attention structure itself.

Method	FLOPS (G)	#parameters (M)
StyleGAN2 [17]	1.08	48.77
+ DFN [79]	4.20	177.60
+ VT [80]	7.39	240.09
+ SAGAN [23]	0.99	44.99
+ SAN [4]	1.08	48.43

Table 2.4: Time complexity in FLOPS and space complexity in the number of parameters for each method.

2.3.3 Reference-attention in the discriminator

First, we apply SAN [4], the best attention mechanism we validated in the generator, to the discriminator. However, we do not see a benefit of such design as shown in Table 2.5. Then, we explore an advanced attention scheme given that two classes of input (real vs. fake) are fed to the discriminator. We allow the discriminator to take two image inputs at the same time: the *reference* image and the *primary* image where we set the reference image to always be a real sample while the primary image to be either a real or generated sample. The reference image is encoded to represent one part of the attention components. These components are learned to guide the other part of the attention components, which are encoded from the *primary* image. There are three insights in this advancement. (1) An effective discriminator encodes real images and generated images differently, so that reference-attention is capable of learning positive feedback given both images from the real class and negative feedback given two images from different classes. Such a scheme amplifies the representation difference between real and fake, and in turn potentially strengthens the power of the discriminator. (2) Reference-attention enables distribution estimation in the discriminator feature level beyond the discriminator logit

level in the original GAN framework, which guides generation more strictly towards the real distribution. (3) Reference-attention learns to cooperate real and generated images explicitly in one round of back-propagation, instead of individually classifying them and trivially averaging the gradients over one batch. Pairing up images mitigates discriminator from overfitting, similar to the spirit of data augmentation.

In detail, we first encode the reference image and the primary image through the original discriminator layers prior to the convolution at a certain resolution. To align feature embeddings, we apply the Siamese architecture [82, 83] to share layer parameters as shown in Fig. 7.1. We then apply the same attention scheme as used in the generator, except we use the tensor $\mathbf{T}_{ref} \in \mathbb{R}^{h \times w \times c}$ from the reference branch to calculate the key and query tensors, and use the tensor $\mathbf{T}_{pri} \in \mathbb{R}^{h \times w \times c}$ from the primary branch to calculate the value tensor and the residual shortcut. Finally, we replace the original convolution output with our reference-attention output:

$$\mathbf{O}^{ref} \doteq \mathsf{attn}\left(\mathbf{K}(\mathbf{T}_{ref}), \mathbf{Q}(\mathbf{T}_{ref}), \mathbf{V}(\mathbf{T}_{pri})\right) + \mathbf{T}_{pri} \tag{2.8}$$

After the reference-attention layer, the two Siamese branches fuse into one and are followed by the remaining discriminator layers to obtain the classification logit. We show in Fig. 2.4 the diagram of reference-attention. Eq. 2.8 provides the flexibility how to cooperate between reference and primary images.

From Table 2.5 we validate reference-attention mechanism (ref attn) to improve the results whereas self-attention to be barely benefiting for the discriminator. Encouraged with these findings, we run the proposed reference-attention on full-scale datasets but do

	CelebA	Animal Face	Bedroom	Church
StyleGAN2 [17]	9.84	36.55	19.33	11.02
+ self attn in D	10.49	42.41	17.22	11.06
+ ref attn in D	**7.48**	**31.08**	**8.32**	**7.86**

Table 2.5: Comparisons in FID among different attention configurations in the discriminator. StyleGAN2 config E which does not include any attention module is used as a backbone. For computationally efficient comparisons, we use the 30k subset of each dataset at 128×128 resolution.

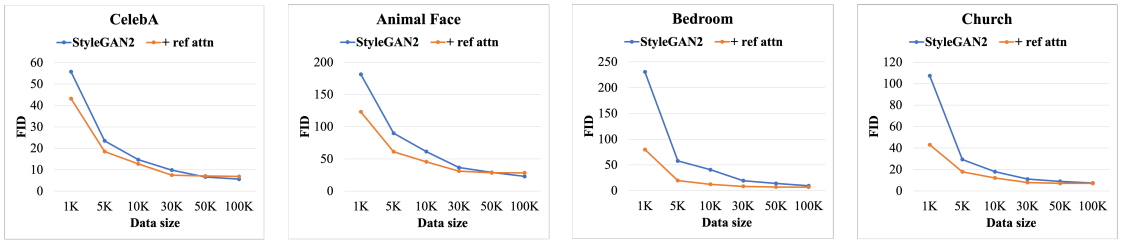


Figure 2.6: Comparisons in FID between StyleGAN2 config E baseline (blue) and that with our reference-attention in the discriminator (orange). Our method consistently improves the baseline when dataset size varies between 1k and 30k images. For computationally efficient comparisons, we use each dataset at 128×128 resolution.

not see any improvements. Therefore, we dive deep into reference-attentions behavior in the discriminator with respect to the dataset size as given in Fig. 2.6. We find that the reference-attention in the discriminator consistently improves the performance when dataset size varies between 1k and 30k images, and on contrary slightly deteriorates the performance when dataset sizes increase further. We reason that the arbitrary pair-up of the reference and primary image inputs to prevent overfitting when data size is small but causing underfitting with the increase of data size Even though in this paper our main scope is GANs on large-scale datasets, we believe these findings to be very interesting for researchers to design their networks for limited-scale datasets.

2.4 Comparisons to the state of the art

Implementation details. All our models are built upon the most recent state-of-the-art unconditional StyleGAN2 [17] config E for its high performance and reasonable speed. We leverage the plug-and-play advantages of all our improvement proposals to strictly follow StyleGAN2 official setup and training protocol, which facilitates reproducibility and fair comparisons. For dual contrastive loss, we first warm up training with the default non-saturating loss for about 20 epochs, and then switch to train with our loss.

Datasets. We use several benchmark datasets, 70K FFHQ face dataset [16], 3M LSUN Bedroom dataset [20], 120K LSUN Church dataset [20], 2M LSUN Horse dataset [20], CelebA face dataset [19] and Animal Face dataset [117], and 70K CLEVR [21] dataset which contains rendered images with random compositions of 3D shapes, uniform materials, uniform colors, point lighting, and a plain background. It poses different challenges from the other common datasets: compositional scenes with occlusions, shadows, reflections, and mirror surfaces. We use 256×256 resolution images for each of these datasets except the CelebA and Animal Face datasets which are used in 128×128 resolutions. We do not experiment with 1024×1024 resolution of FFHQ as it takes 9 days to train StyleGAN2 base model. Instead, we run extensive experiments on the mentioned various datasets. If not otherwise noted, we use the whole dataset.

Evaluation. FID [118] is regarded as the golden standard to quantitatively evaluate generation quality. We follow the protocol in StyleGAN2 [17] to report the FID between 50K generated images and 50K real testing images. The smaller the more desirable.

Comparisons. Besides StyleGAN2 [17], we also compare to a parallel state-of-the-

Method	Loss	FFHQ	Bedroom	Church	Horse	CLEVR
BigGAN [14]	Adv	11.4	-	-	-	-
U-Net GAN [24]	Adv	7.48	17.6	11.7	20.2	33.3
StyleGAN2 [17]	Adv	4.86	4.01	4.54	3.91	9.62
StyleGAN2 w/ attn	Adv	5.13	<u>3.48</u>	4.38	<u>3.59</u>	8.96
StyleGAN2	Contr	**3.98**	3.86	<u>3.73</u>	3.70	<u>6.06</u>
StyleGAN2 w/ attn	Contr	<u>4.63</u>	**3.31**	**3.39**	**2.97**	**5.05**

Table 2.6: Comparisons in FID to the state-of-the-art GANs on the large-scale datasets. We highlight the best in **bold** and second best with <u>underline</u>. "w/ attn" indicates using the self-attention in the generator. "Contr" indicates using our dual contrastive loss instead of conventional GAN loss.

art study, U-Net GAN [24], which was build upon and improved on BigGAN [14]. We train U-Net by adapting it to the better backbone of StyleGAN2 [17] for fair comparison, and obtain better results than their official release on non-FFHQ datasets. As shown in Table 2.6, our self-attention generator improves on four out of five large-scale datasets, up to 13.3% relative improvement on Bedroom dataset. This highlights the benefits of attention to details and to long-range dependencies on complex scenes. However, self-attention does not improve on the extensively studied FFHQ dataset. We reason that the image pre-processing of facial landmark alignment compensates for the lack of attention schemes, which makes previous works also overlook them on other datasets.

Our dual contrastive loss improves effectively on all the datasets, up to 37% improvement on CLEVR dataset. This highlights the benefits of contrastive learning on generalized representation, especially on aligned datasets, e.g. FFHQ and CLEVR, that can easily make a traditional discriminator overfit. The synergy effective between self-attention and contrastive learning is significant and consistent, resulting in at least 17.5% and up to 47.5% relative improvement on CLEVR. Especially for CLEVR, our generator handles

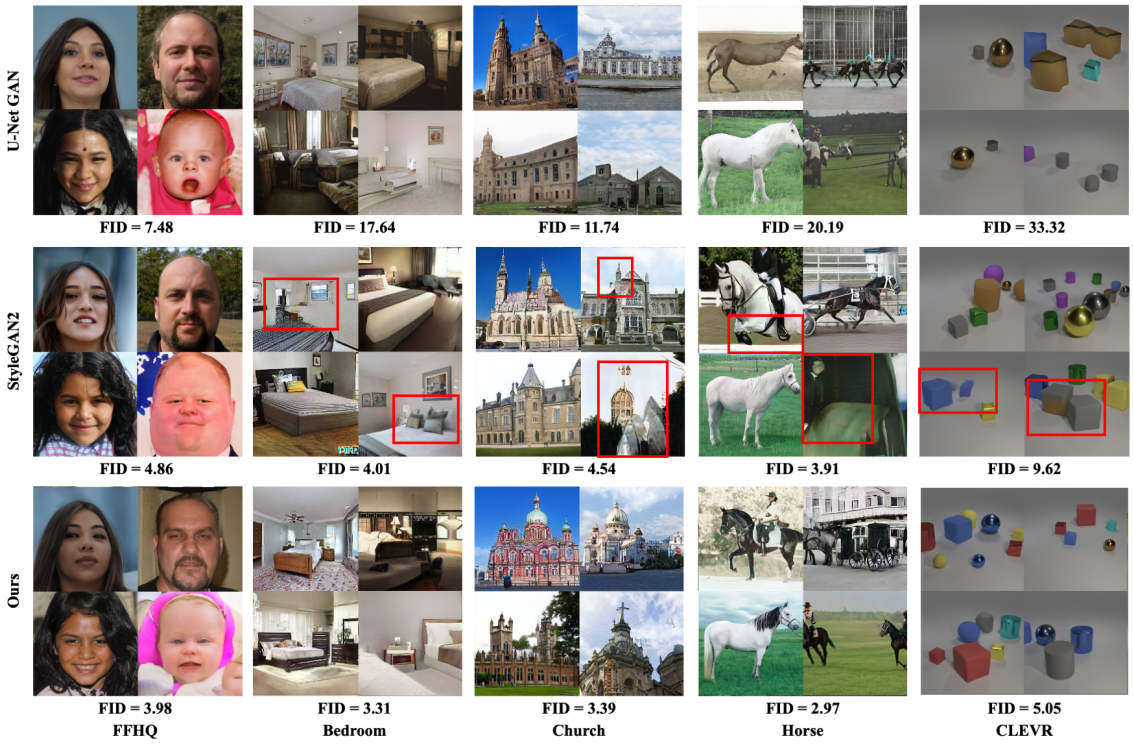


Figure 2.7: Uncurated generated samples. To align the comparisons, we use the same real query images for pre-trained generators to reconstruct. Artifacts from StyleGAN2 are highlighted with red boxes. Zoom in for details. In particular, our generation significantly outperforms the baselines on CLEVR images which strongly rely on long-range dependencies (occlusions, shadows, reflections, etc) and consistency (consistent shadow directions, consistent specularity, regular shapes, uniform colors, etc).

more realistically for occlusions, shadows, reflections, and mirror surfaces. As shown in Fig. 6.3, our method suppresses artifacts that were previously visible in StyleGAN2 baseline outputs, with red boxes, e.g., the artifacts on the wall in Bedroom images, discontinuities in the structure in Church images, as well as color leakage between objects in CLEVR images.

2.5 Conclusion

The advancements in attention schemes and contrastive learning generate opportunities for new designs of GANs. Our attention schemes serve as a beneficial replacement for local and stationary convolutions, so as to equip generation and discriminator representation with long-range adaptive dependencies. In particular, our reference-attention discriminator cooperates between real reference images and primary images, mitigates discriminator overfitting, and leads to further boost on limited-scale datasets. Additionally, our novel contrastive loss generalizes discriminator representations, makes them more distinguishable between real and fake, and in turn incentivizes better generation quality.

2.6 Acknowledgement

Ning Yu is partially supported by Twitch Research Fellowship. We thank Tero Karras, Xun Huang, and Tobias Ritschel for constructive advice in general.

Chapter 3: Texture Mixer: A Network for Controllable Synthesis and Interpolation of Texture

3.1 Introduction

Many materials exhibit variation in local appearance, as well as complex transitions between different materials. Editing materials in an image, however, can be highly challenging due to the rich, spatially-varying material combinations as we see in the natural world. One general research challenge then is to attempt to enable these kinds of edits. In particular, in this paper, we focus on textures. We define "texture" as being an image-space representation of a statistically homogeneous material, captured from a top-down view. We further focus on allowing a user to both be able to accurately control the placement of textures, as well as create plausible transitions between them.

Because of the complex appearance of textures, creating transitions by interpolating between them on the pixel domain is difficult. Doing so naïvely results in unpleasant artifacts such as ghosting, visible seams, and obvious repetitions. Researchers in texture synthesis have therefore developed sophisticated algorithms to address this problem. These may be divided to two families: non-parametric methods such as patch-based synthesis (e.g. [119, 120, 121]) and parametric methods (e.g. [122, 123]), including neural network

synthesis approaches (e.g. [124, 125, 126, 127, 128]). Previously, researchers used sophisticated patch-based interpolation methods [129, 130] with carefully crafted objective functions. However, such approaches are extremely slow. Moreover, due to the hand-crafted nature of their objectives, they cannot learn from a large variety of textures in the natural world, and as we show in our comparisons are often brittle and frequently result in less pleasing transitions. Further, we are not aware of any existing feedforward neural network approaches that offer both fine-grained controllable synthesis and interpolation between multiple textures. User-controllable texture interpolation is substantially more challenging than ordinary texture synthesis, because it needs to incorporate adherence to user-provided boundary conditions and a smooth transition for the interpolated texture.

In our paper, we develop a neural network approach that we call "Texture Mixer" which allows for both user control and interpolation of texture. We define the *interpolation* of texture as a broad term, encompassing any combination of: (1) Either gradual or rapid *spatial transitions* between two or more different textures, as shown in the palette, the letters, and the background in Figure 3.1, and (2) *Texture dissolve*, where we can imagine putting two textures in different layers, and cross-dissolving them according to a user-controlled transparency, as we show in our video. Previous neural methods can create interpolations similar to our dissolves by changing the latent variable [128, 131, 132, 133, 134]. Thus, in this paper we focus primarily on high-quality spatial interpolation: this requires textures to coexist in the same image plane without visible seams or spatial repetitions, which is more difficult to achieve. Our feedforward network is trained on a large dataset of textures and runs at interactive rates.

Our approach addresses the difficulty of interpolating between textures on the image

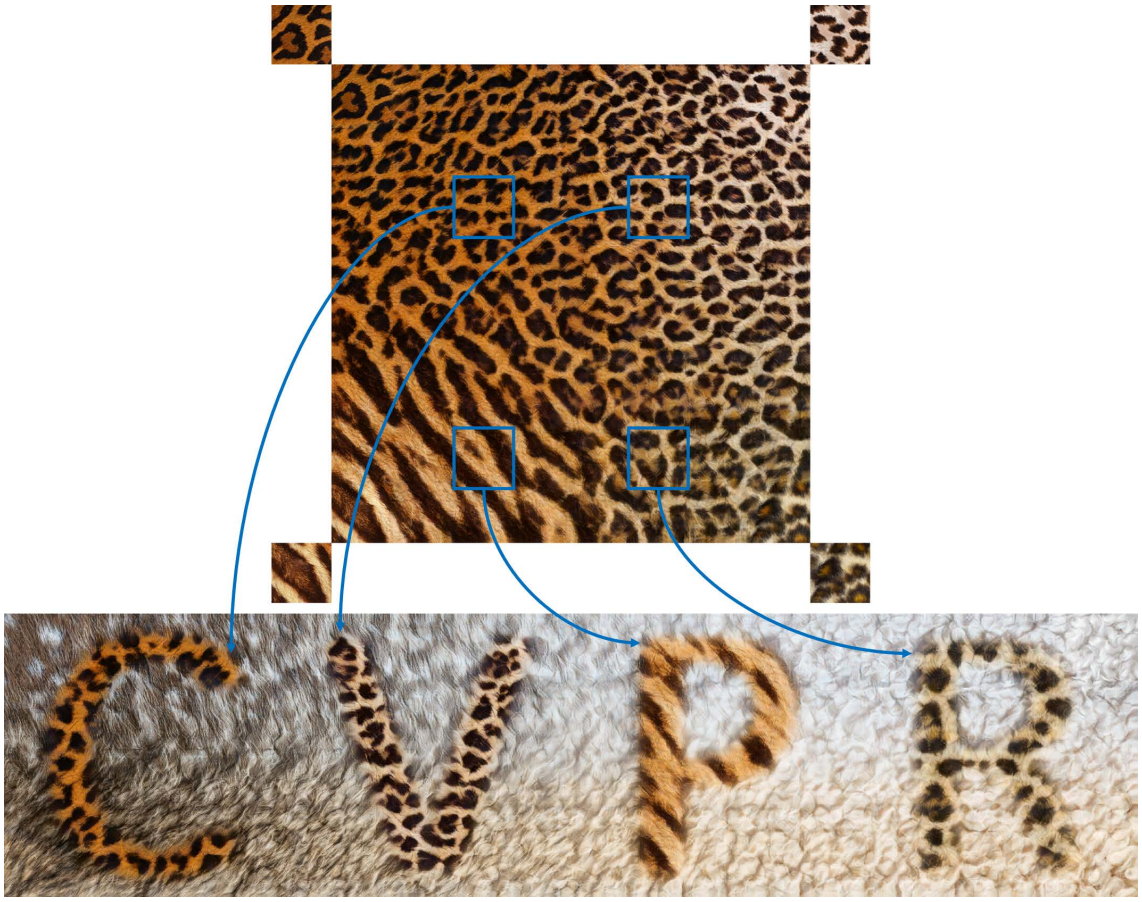


Figure 3.1: Texture interpolation and texture painting using our network on the *animal texture* dataset. The top part shows a 1024×1024 palette created by interpolating four source textures at the corners outside the palette. The bottom part shows a 512×2048 painting of letters with different textures sampled from the palette. The letters are interpolated by our method with the background, also generated by our interpolation.

domain by projecting these textures onto a latent domain where they may be linearly interpolated, and then decoding them back into the image domain to obtain the desired result. In order to satisfy the two goals of *controllability* and *visual realism*, we train our network simultaneously for both tasks. A *reconstruction task* ensures that when a texture is passed through an encoder and then a decoder (an autoencoder), the result will be similar to the input. This allows the user to specify texture at any given point of the output by example. An *interpolation task* uses a discriminator to ensure that linear interpolations of latent tensors also decode into plausible textures, so that the regions of the output not directly specified by the user are realistic and artifact-free. For this task, we can view our network as a conditional Generative Adversarial Network (GAN). In effect, we thus train an autoencoder and a conditional GAN at the same time, using shared weights and a shared latent space.

To perform the interpolation task, we take texture samples that user specifies, and project them into latent space using a learned encoder. Given these latent tensors, our network then uses three intuitive latent-space operations: tiling, interpolation, and shuffling. The tiling operation extends a texture spatially to any arbitrary size. The interpolation operation uses weighted combinations of two or more textures in latent domain. The shuffling operation swaps adjacent small squares within the latent tensor to reduce repetitions. These new latent tensors are then decoded to obtain the interpolated result.

Our main contributions are: (1) a novel interactive technique that allows both user control and interpolation of texture; (2) several practical and creative applications based on our technique; (3) a new suite of metrics that evaluate user controllability, interpolation smoothness, and interpolation realism; and (4) the state-of-the-art performance superior

to previous work both based on these metrics, and based on a user study if we consider them holistically.

3.2 Related Work

The problem of user-controllable texture interpolation has so far been under-explored. It is however closely related to several other problems, most significantly texture synthesis, inpainting, and stylization.

Texture synthesis algorithms can be divided into two families. The first one is parametric, with a generative texture model. These algorithms include older, non-neural methods [122, 123], and also more recent deep learning-based methods that are based on optimization [124, 135, 136, 137] or trained feedforward models [125, 126, 127, 128]. Where the underlying model allows spatially varying weights for combination, it may be used to cross-dissolve textures. However, we are not aware of any existing texture synthesis techniques in this family that enables spatial transition between different textures.

The second family of texture synthesis algorithms is non-parametric, in which the algorithm produces output that is optimized to be as close as possible to the input under some appearance measure [119, 120, 121, 129, 138, 139, 140, 141, 142, 143, 144]. These can be formulated to accept two different inputs and spatially vary which is being compared to, facilitating interpolation [129, 130]. As we mentioned before, such approaches are slow, and due to the hand-crafted nature of their objectives, they tend to be brittle.

Recently, generative adversarial networks (GANs) [7, 65, 66, 145] have shown improved realism in image synthesis and translation tasks [29, 31, 146]. GANs have also been used directly for texture synthesis [127, 147, 148], however, they were limited to a single texture they were trained on. A recent approach dubbed PSGAN [149] learns to synthesize a collection of textures present in a single photograph, making it more general and applicable to texture interpolation; it is not, however, designed for our problem as it cannot interpolate existing images. We show comparisons with PSGAN and it cannot reconstruct many input textures, even after running a sophisticated optimization or jointly associating PSGAN with an encoder. Moreover, PSGAN can suffer from mode collapse.

Texture synthesis and image inpainting algorithms are often closely related. A good hole filling algorithm needs to be able to produce some sort of transition between textures on opposite ends of the hole, and so may be used in a texture interpolation task. A few recent deep learning-based methods showed promising results [5, 51, 150, 151].

Finally, some neural stylization approaches [127, 131, 133, 135] based on separating images into content and style components have shown that, by stylizing a noise content image, they can effectively synthesize texture [124]. By spatially varying the style component, texture interpolation may thus be achieved.

3.3 Our network: Texture Mixer

In this section, we explain how our network works. We first explain in Section 3.3.1 how our method is trained. We then show how our training losses are set up in Section 3.3.2. Finally, we explain in Section 3.3.3 how our method can be either tested or used by an

end user.

3.3.1 Training setup

We aim to train our network simultaneously for two tasks: *reconstruction* and *interpolation*. The *reconstruction task* ensures that every input texture after being encoded and then decoded results in a similar texture. Meanwhile, the *interpolation task* ensures that interpolations of latent tensors are also decoded into plausible textures.

Our method can be viewed as a way of training a network containing both encoders and a generator, such that the generator is effectively a portion of a GAN. The network accepts a source texture S as input. A *global encoder* $E^g(S)$ encodes S into a latent vector z^g, which can also be viewed as a latent tensor with spatial size 1×1. A *local encoder* $E^l(S)$ encodes the source texture into a latent tensor z^l, which has a spatial size that is a factor m smaller than the size of the input texture: we use $m = 4$. The generator $G(z^l, z^g)$ concatenates z^l and z^g, and can decode these latent tensors back into a texture patch, so that ideally $G(E^l(S), E^g(S)) = S$, which encompasses the reconstruction task. Our generator is fully convolutional, so that it can generate output textures of arbitrary size: the output texture size is directly proportional to the size of the local tensor z^l. A discriminator D^{rec} is part of the reconstruction loss. An identical but separately trained discriminator D^{itp} evaluates the realism of interpolation.

Note that in practice, our generator network is implemented as taking a global tensor as input, which has the same spatial size as the local tensor. This is because, for some applications of texture interpolation, z^g can actually vary spatially. Thus, when we refer

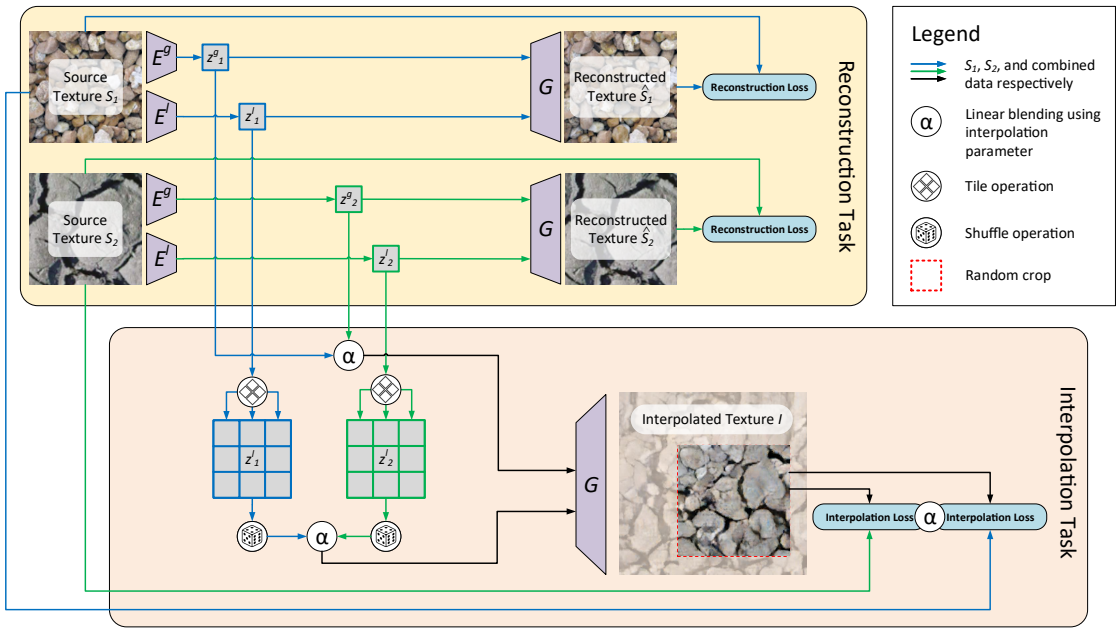


Figure 3.2: A diagram of our method. Background color highlights each of the tasks. Trapezoids represent trainable components that share weights if names match. Rounded rectangles represent the losses. Arrows and circles represent operations on tensor data.

to G taking a global latent vector z^g with spatial size 1×1 as input, what we mean is that this z^g vector is first repeated spatially to match the size of z^l, and the generator is run on the result.

We show the full training setup in Figure 3.2. We will also explain our setup in terms of formulas here. As is shown in the upper-left of Figure 3.2, the network is given two real source texture images S_1 and S_2 from the real texture dataset $\mathcal{S}$. Each local encoder E^l encodes S_i ($i \in \{1, 2\}$) to a local latent tensor $z^l_i = E^l(S_i)$. Meanwhile, each global encoder E^g encodes S_i to a global latent vector z^g_i, denoted as $z^g_i = E^g(S_i)$. These latent variables are shown in green and blue boxes in the upper-left of Figure 3.2.

For the *reconstruction task*, we then evaluate the reconstructed texture image $\hat{S}_i = G\big(z^l_i, z^g_i\big)$. These are shown in the upper center of Figure 3.2. For each reconstructed image $\hat{S}_i$, we then impose a weighted sum of three losses against the original texture S_i.

We describe these losses in more detail later in Section 3.3.2.

For the *interpolation task*, we pose the process of multiple texture interpolation as a problem of simultaneously (1) synthesizing a larger texture, and (2) interpolating between two different textures. In this manner, the network learns to perform well for both single and multiple texture synthesis. For single texture synthesis, we enlarge the generated images by a factor of 3×3. We do this by tiling z_i^l spatially by a factor of 3×3. We denote this tiling by $T(z_i^l)$, and indicate tiling by a tile icon in the lower-left of Figure 3.2. We chose the factor 3 because this is the smallest integer that can synthesize transitions over the four edges of z_i^l. Such a small tiling factor minimizes computational cost. The tiling operation can be beneficial for regular textures. However, in semiregular or stochastic textures, the tiling introduces two artifacts: undesired spatial repetitions, and undesired seams on borders between tiles.

We reduce these artifacts by applying a random shuffling to the tiled latent tensors $T(z_i^l)$. In Figure 3.2, this shuffling operation is indicated by a dice icon. Random shuffling in the latent space not only results in more varied decoded image appearance and thus reduces visual repetition, but also softens seams by spatially swapping "pixels" in the latent space across the border of two z_i^l tensors.

We implement the random shuffling by row and column swapping over several scales from coarse to fine. For this coarse to fine process, we use scales that are powers of two: $s_i = 2^i$ for $i = 0, 2, \ldots, n$. We set the coarsest scale n to give a scale s_n that is half the size of the local tensor z_i^l. For each scale s_i, we define a grid over the tiled latent tensor $T(z^l)$, where each grid cell has size $s_i \times s_i$. For each scale s_i, we then apply a random shuffling on cells of the grid for that scale: we denote this by P_i. This

shuffling proceeds through grid rows first in top-down and then bottom-up order: each row is randomly swapped with the succeeding row with probability 0.5. Similarly, this is repeated on grid columns, with column swapping from left to right and right to left. Thus, the entire shuffling operation is:

$$P\big(T(z_i^l)\big) = P_0 \circ P_1 \circ \cdots \circ P_n\big(T(z_i^l)\big) \tag{3.1}$$

We also want the synthesized texture to be able to transit smoothly between regions where there are user-specified texture constraints and regions where there are none. Thus, we override the original z_i^l without shuffling at the 4 corners of the tiled latent tensor. We denote such shuffling with corner overriding as $\tilde{P}\big(T(z_i^l)\big)$.

If we apply the fully convolutional generator G to a network trained using a single input texture and the above shuffling process, it will work for single texture synthesis. However, for multiple texture interpolation, we additionally apply interpolation in the latent space before calling G, as inspired by [128, 131, 149]. We randomly sample an interpolation parameter $\alpha \sim U[0, 1]$, and then interpolate the latent tensors using α. This is shown by the circles labeled with α in Figure 3.2. We linearly blend the shuffled local tensors $\tilde{P}\big(T(z_1^l)\big)$ and $\tilde{P}\big(T(z_2^l)\big)\big)$, which results in the final interpolated latent tensor Z^l:

$$Z^l = \alpha\tilde{P}\big(T(z_1^l)\big) + (1-\alpha)\tilde{P}\big(T(z_2^l)\big) \tag{3.2}$$

In the same way, we blend z_1^g and z_2^g to obtain

$$Z^g = \alpha z_1^g + (1 - \alpha) z_2^g \tag{3.3}$$

Finally, we feed the tiled and blended tensors into the generator G to obtain an interpolated texture image $I = G(Z^l, Z^g)$, which is shown on the right in Figure 3.2. From the interpolated texture, we take a random crop of the same size as the input textures. The crop is shown in the red dotted lines in Figure 3.2. The crop is then compared using appropriately α-weighted losses to each of the source textures. We use spatially uniform weights α at training time because all the real-world examples are *spatially homogeneous* and we do not want our adversarial discriminator to detect our synthesized texture due to it having spatial variation. In contrast, at testing time, we use spatially varying weights.

3.3.2 Training losses

For the *reconstruction task*, we use three losses. The first loss is a pixel-wise L_1 loss against each input S_i. The second loss is a Gram matrix loss against each input S_i, based on an ImageNet-pretrained VGG-19 model. We define the Gram loss L_{Gram} in the same manner as Johnson [126], and use the features relu$i$_1 for $i = 1, \ldots, 5$. The third loss is an adversarial loss L_{adv} based on WGAN-GP [66], where the reconstruction discriminator D^{rec} tries to classify whether the reconstructed image is from the real source texture set or generated by the network. The losses are:

$$L_{\text{pix}}^{\text{rec}} = \|\hat{S}_1 - S_1\|_1 + \|\hat{S}_2 - S_2\|_1 \tag{3.4}$$

$$L^{\text{rec}}_{\text{Gram}} = L_{\text{Gram}}(\hat{S}_1, S_1) + L_{\text{Gram}}(\hat{S}_2, S_2) \tag{3.5}$$

$$L^{\text{rec}}_{\text{adv}} = L_{\text{adv}}(\hat{S}_1, S_1 | D^{\text{rec}}) + L_{\text{adv}}(\hat{S}_2, S_2 | D^{\text{rec}}) \tag{3.6}$$

The L_{adv} term is defined from WGAN-GP [66] as:

$$L_{\text{adv}}(A, B|D) = D(A) - D(B) + GP(A, B|D) \tag{3.7}$$

Here A and B are a pair of input images, D is the adversarially trained discriminator, and $GP(\cdot)$ is the gradient penalty regularization term.

For the *interpolation task*, we expect the large interpolated texture image to be similar to some combination of the two input textures. Specifically, if $\alpha = 1$, the interpolated image should be similar to source texture S_1, and if $\alpha = 0$, it should be similar to S_2. However, we do not require pixel-wise similarity, because that would encourage ghosting. We thus impose only a Gram matrix and an adversarial loss. We select a random crop I_{crop} from the interpolated texture image. Then the Gram matrix loss for interpolation is defined as an α-weighted loss to each source texture:

$$L^{\text{itp}}_{\text{Gram}} = \alpha L_{\text{Gram}}(I_{\text{crop}}, S_1) + (1 - \alpha) L_{\text{Gram}}(I_{\text{crop}}, S_2) \tag{3.8}$$

Similarly, we adversarially train the interpolation discriminator D^{itp} for the interpolation task to classify whether its input image is from the real source texture set or whether it is

a synthetically generated interpolation:

$$L_{\text{adv}}^{\text{itp}} = \alpha L_{\text{adv}}(I_{\text{crop}}, S_1 | D^{\text{itp}}) + (1-\alpha) L_{\text{adv}}(I_{\text{crop}}, S_2 | D^{\text{itp}}) \tag{3.9}$$

Our final training objective is

$$\begin{aligned} \min_{E^l, E^g, G} \max_{D^{\text{rec}}, D^{\text{itp}}} \mathop{\mathbb{E}}_{S_1, S_2 \sim \mathcal{S}} (\lambda_1 L_{\text{pix}}^{\text{rec}} + \lambda_2 L_{\text{Gram}}^{\text{rec}} + \lambda_3 L_{\text{adv}}^{\text{rec}} \\ + \lambda_4 L_{\text{Gram}}^{\text{itp}} + \lambda_5 L_{\text{adv}}^{\text{itp}}) \end{aligned} \tag{3.10}$$

where $\lambda_1 = 100$, $\lambda_2 = \lambda_4 = 0.001$, and $\lambda_3 = \lambda_5 = 1$ are used to balance the order of magnitude of each loss term, which are not sensitive to dataset.

3.3.3 Testing and user interactions

At testing time, we can use our network in several different ways: we can interpolate sparsely placed textures, brush with textures, dissolve between textures, and hybridize different animal regions in one image. Each of these applications utilizes spatially varying interpolation weights.

Interpolation of sparsely placed textures. This option is shown in the palette and background in Figure 3.1. In this scenario, one or more textures are placed down by the user in the image domain. These textures are each encoded to latent domain.

In most cases, given input textures, our method is able to achieve inherent boundary matching and continuity. However, because of the trade-off between reconstruction and interpolation losses, there might be a *slight* mismatch in some cases. To make the textures

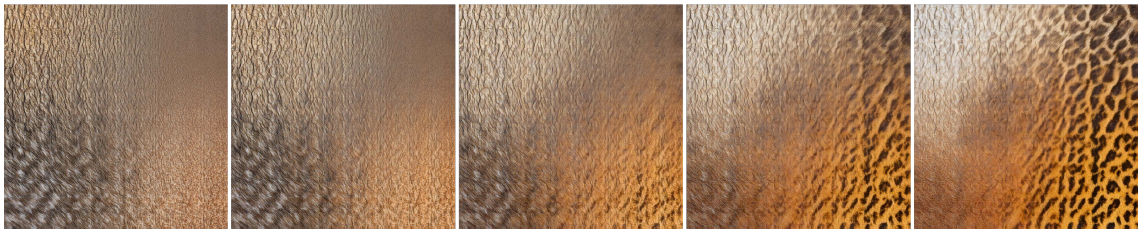


Figure 3.3: A sequence of dissolve video frame samples with size 1024×1024 on the *animal texture* dataset, where each frame is also with effect of interpolation.

better agree at boundary conditions, we postprocess our images as follows. Suppose that the user places a source textured region as a boundary condition. We first replace the reconstructed regions with the source texture. Then, within the source texture, we use graph cuts [139] to determine an optimal seam where we can cut between the source texture and the reconstruction. Finally, we use Poisson blending [152] to minimize the visibility of this seam.

Texture brush. We can allow the user to brush with texture as follows. We assume that there is a textured background region, which we have encoded to latent space. The user can select any texture to brush with, by first encoding the brush texture and then brushing in the latent space. For example, in Figure 3.1 we show an example of selecting a texture from a palette created by interpolating four sparsely created textures. We find the brush texture's latent domain tensors, and apply them using a Gaussian-weighted brush. Here full weight in the brush causes the background latent tensors to be replaced entirely, and other weights cause a proportionately decreased effect. The brush can easily be placed spatially because the latent and image domains are aligned with a resizing factor m related to the architecture.

Texture dissolve. We can create a cross-dissolve effect between any two textures

Figure 3.4: An animal hybridization example of size 1260×1260 between a dog and a bear. Our interpolation between the two animal furs is smoother, has less ghosting, and is more realistic than that of the Naïve α-blending.

by encoding them both to latent domain and then blending between them using blending weights that are spatially uniform. This effect is best visualized in a video, where time controls the dissolve effect. Figure 3.3 shows a sequence of video frame samples with gradually varying weights.

Animal hybridization. We generalize texture interpolation into a more practical and creative application - animal hybridization. Figure 3.4 shows an example. Given two aligned animal regions in one image and a hole over the transition region, we can sample source texture patches adjacent to the hole and conduct spatial interpolation among those textures. We fill the hole using our interpolated texture. Finally, we use graph cuts [139] and Poisson blending [152] to postprocess the boundaries.

3.4 Experiments

In this section, we demonstrate experimental comparisons. We first introduce our own datasets in Section 3.4.1. We then present in Section 3.4.2 a suite of evaluation

metrics for interpolation quality. In Section 3.4.3 we list and compare against several leading methods from different categories on the task of texture interpolation. In Section 3.4.4 we describe a user study as a holistic comparison. Finally, we conduct in Section 3.4.5 the ablation study by comparing against three simplified versions of our own method.

We propose to learn a model per texture category rather than a universal model because: (1) there are no real-world examples that depict interpolation between distinct texture categories; (2) there is no practical reason to interpolate across categories, e.g., fur and gravel; and (3) like with other GANs, a specific model per category performs better than a universal one due to the model's capacity limit.

3.4.1 Datasets

Training to interpolate frontal-parallel stationary textures of a particular category requires a dataset with a rich set of examples to represent the intra-variability of that category. Unfortunately, most existing texture datasets such as DTD [153] are intended for texture classification tasks, and do not have enough samples per category (only 120 in the case of DTD) to cover the texture appearance space with sufficient density.

Therefore, we collected two datasets of our own: (1) the *earth texture* dataset contains Creative Commons images from Flickr, which we randomly split into 896 training and 98 testing images; (2) the *animal texture* dataset contains images from Adobe Stock, randomly split into 866 training and 95 testing images. All textures are real-world RGB photos with arbitrary sizes larger than 512×512. Examples from both are shown in our figures throughout the paper.

We further augmented all our training and testing sets by applying: (1) color histogram matching with a random reference image in the same dataset; (2) random geometric transformations including horizontal and vertical mirroring, random in-plane rotation and downscaling (up to $\times 4$); and (3) randomly cropping a size of 128×128. In this way, we augmented $1,000$ samples for each training image and 100 samples for each testing image.

3.4.2 Evaluation

We will compare previous work with ours, and also do an ablation study on our own method. In order to fairly compare all methods, we use a horizontal interpolation task. Specifically, we randomly sampled two 128×128 squares from the test set. We call these the side textures. We placed them as constraints on either end of a 128×1024 canvas. We then used each method to produce the interpolation on the canvas, configuring each method to interpolate linearly where such option is available.

To the best of our knowledge, there is no standard method to quantitatively evaluate texture interpolation. We found existing generation evaluation techniques [7, 118, 132, 154] inadequate for our task. We, therefore, developed a suite of metrics that evaluate three aspects we consider crucial for our task: (1) user controllability, (2) interpolation smoothness, and (3) interpolation realism. We now discuss these.

User controllability. For interpolation to be considered controllable, it has to closely reproduce the user's chosen texture at the user's chosen locations. In our experiment, we measure this as the reconstruction quality for the side textures. We average the LPIPS

perceptual similarity measure [8] for the two side textures. We call this *Side Perceptual Distance (SPD)*.

We also would like the center of the interpolation to be similar to both side textures. To measure this, we consider the Gram matrix loss [126] between the central 128×128 crop of the interpolation and the side textures. We report the sum of distances from the center crop to the two side textures, normalized by the Gram distance between the two. We call this measure the *Center Gram Distance (CGD)*.

Interpolation smoothness. Ideally, we would like the interpolation to follow the shortest path between the two side textures. To measure this, we construct two difference vectors of Gram matrix features between the left side texture and the center crop, and between the center crop and the right side texture, and measure the cosine distance between the two vectors. We expect this *Centre Cosine distance (CCD)* to be minimized.

For smoothness, the appearance change should be gradual, without abrupt changes such as seams and cuts. To measure such, we train a *seam classifier* using real samples from the training set as negative examples, and where we create synthetic seams by concatenating two random textures as positive examples. We run this classifier on the center crop. We call this the *Center Seam Score (CSS)*. The architecture and training details of seam classifier are the same as those of D^{rec} and D^{itp}.

Interpolation realism. The texture should also look realistic, like the training set. To measure this, we chose the Inception Score [7] and Sliced Wasserstein Distance (SWD) [132], and apply them on the center crops. This gives *Center Inception Score (CIS)* and *Center SWD*, respectively. For *CIS*, we use the state-of-the-art *Inception-ResNet-v2* inception model architecture [155] finetuned with our two datasets separately.

We also found these metrics do not capture undesired repetitions, a common texture synthesis artifact. We, therefore, trained a *repetition classifier* for this purpose. We call this the *Center Repetition Score (CRS)*. The architecture and training details of repetition classifier are almost the same as those of the seam classifier except the input image size is 128×256 instead of 128×128, where the negative examples are random crops of size 128×256 from real datasets and the positive examples are horizontally tiled twice from random crops of size 128×128 from real datasets.

3.4.3 Comparisons

We compare against several leading methods from different categories on the task of texture interpolation. These include: naïve α-blending, Image Melding [129] as a representative of patch-based techniques, two neural stylization methods - AdaIN [131] and WCT [133], a recent deep hole-filling method called DeepFill [5], and PSGAN [149] which is the closest to ours but without user control. Most these had to be adapted for our task. Fig. 3.5 contains a qualitative comparison between the different methods. Note that in this example: (1) the overly sharp interpolation of DeepFill, (2) the undesired ghosting and repetition artifacts of naïve α-blending and ours (no shuffling), (3) the incorrect reconstruction and less relevant interpolation of AdaIN, WCT, and PSGAN, (4) the appearance mismatch between source and interpolation of Image Melding, (5) the lack of smoothness of ours (no z^g), and (6) the undesired fading of ours (no blending). We also report qualitative results, including the user study and the ablation experiments, in Table 3.1, that contains average values for the two datasets - *earth texture* and *animal*

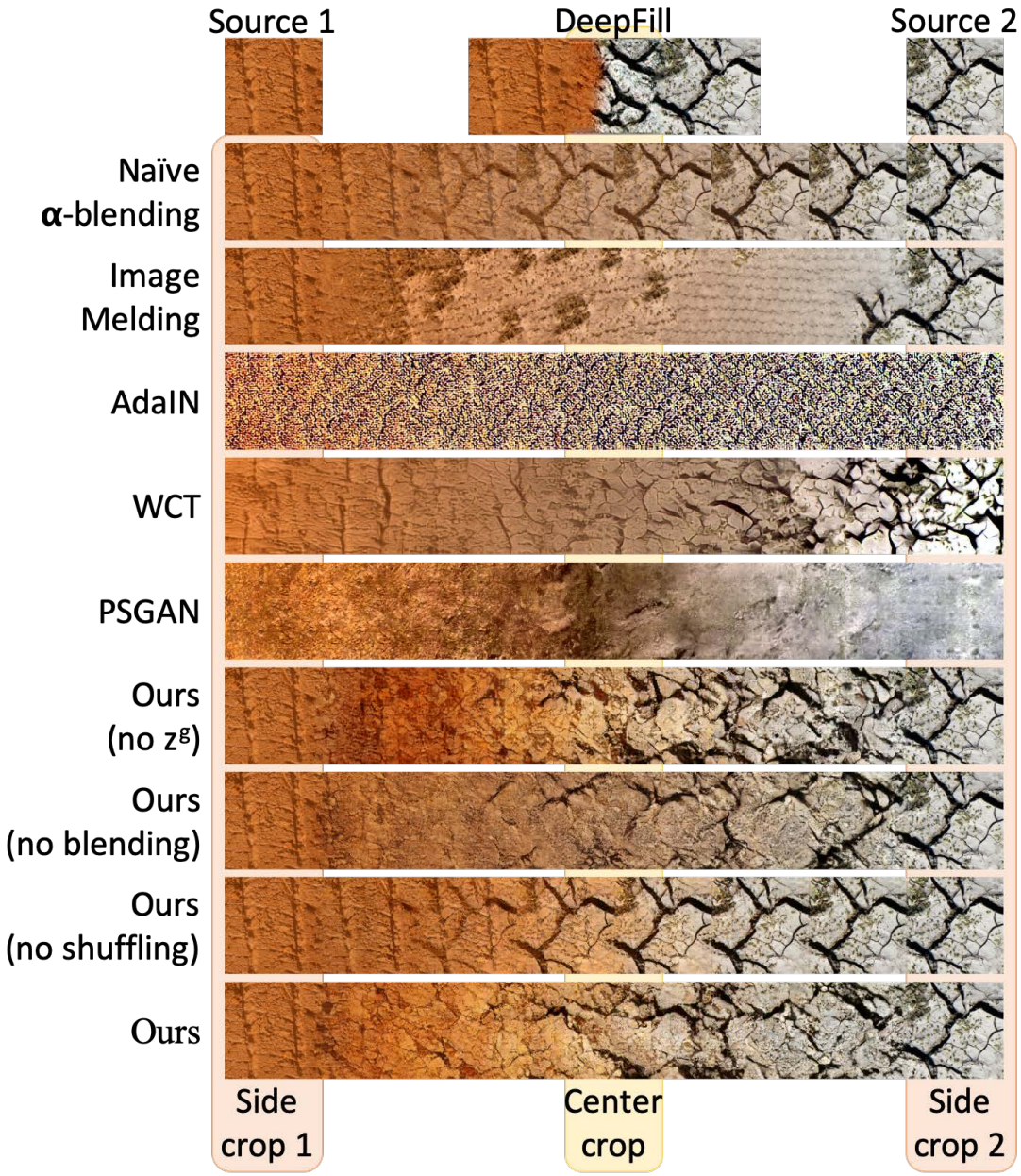


Figure 3.5: Qualitative demonstrations and comparisons of horizontal interpolation in the size of 128×1024 on the *earth texture* samples. We use the two side crops with the orange background for SPD measurement, and the center crop with the light yellow background for the other proposed quantitative evaluations. For the DeepFill [5] method, since the default design is not suitable for inpainting a wide hole due to lack of such ground truth, we instead test it on a shorter interpolation of size 128×384.

	Controllability		Smoothness		Realism			User study		Testing time
	SPD ⇓	CGD ⇓	CCD ⇓	CSS ⇓	CRS ⇓	CIS ⇑	CSWD ⇓	PR	p-value	
Naïve α-blending	0.0000	1.255	0.777	0.9953	0.4384	22.35	60.93	0.845	$< 10^{-6}$	0.02 s
Image Melding [129]	0.0111	1.289	0.865	0.0005	0.0004	**29.45**	47.09	0.672	$< 10^{-6}$	6 min
WCT [133]	0.8605	1.321	0.988	0.0020	0.0000	9.86	46.89	0.845	$< 10^{-6}$	7.5 s
PSGAN [149]	1.1537	1.535	1.156	0.0069	0.0005	26.81	35.90	0.967	$< 10^{-6}$	1.4 min
Ours (no z^g)	0.0112	1.207	0.680	0.0078	0.0010	21.04	21.54	-	-	-
Ours (no blending)	0.0103	1.272	0.817	0.0125	0.0009	22.24	52.29	-	-	-
Ours (no shuffling)	0.0107	**1.129**	**0.490**	0.0534	0.2386	26.78	**20.99**	-	-	-
Ours	0.0113	1.177	0.623	0.0066	0.0008	26.68	22.10	-	-	0.5 s

Table 3.1: Quantitative evaluation averaging over the *earth texture* and *animal texture* datasets. We highlighited the **best**, second best and very high values for each metric. We also indicate for each whether higher (⇑) or lower (⇓) values are more desirable.

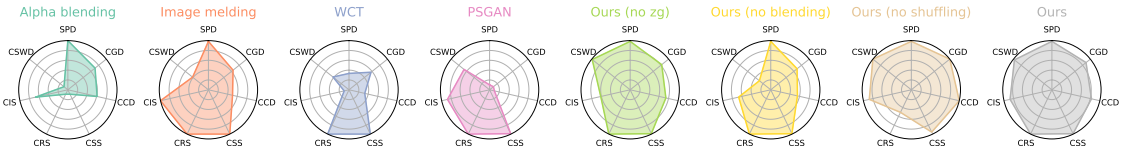


Figure 3.6: Radar charts visualizing Table 3.1. Values have been normalized to the unit range, and axes inverted so that higher value is always better. The first four are baseline methods and next three ablation candidates, with the last entry representing our full method. Our method scores near-top marks all around and shows balanced performance according to all metrics.

texture. Figure 3.6 summarizes the quantitative comparisons.

3.4.4 User study

We also conducted a user study on Amazon Mechanical Turk. We presented the users with a binary choice, asking them if they aesthetically prefer our method or one of the baseline methods on a random example from the horizontal interpolation task. For each method pair, we sampled 90 examples and collected 5 independent user responses per example. Tallying the user votes, we get 90 results per method pair. We assumed a null hypothesis that on average, our method will be preferred by 2.5 users for a given method pair. We used a one-sample permutation t-test to measure p-values, using 10^6

permutations, and found the p-values for the null hypothesis are all $< 10^{-6}$. This indicates that the users do prefer one method over another. To quantify this preference, we count for each method pair all the examples where at least 3 users agree in their preference, and report a *preference rate (PR)* which shows how many of the preferences were in our method's favor. Both PR and the p-values are listed in Table 3.1.

3.4.5 Ablation study

We also compare against simplified versions of our method. The qualitative results for this comparison are shown in Figure 3.5. We report quantitative result numbers in Table 3.1, and visualized them in Figure 3.6. We ablate the following components:

Remove $\mathbf{z^g}$. The only difference between z^g and z^l is in the tiling and shuffling for z^l. However, if we remove z^g, we find texture transitions are less smooth and gradual.

Remove texture blending during training. We modify our method so that the interpolation task during training is performed only upon two identical textures. This makes the interpolation discriminator D^{itp} not be aware of the realism of blended samples, so testing realism deteriorates.

Remove random shuffling. We skip the shuffling operation in latent space and only perform blending during training. This slightly improves realism and interpolation directness, but causes visually disturbing repetitions.

3.5 Conclusion

We presented a novel method for controllable interpolation of textures. We were able to satisfy the criteria of controllability, smoothness, and realism. Our method outperforms several baselines on our newly collected datasets. As we see in Figure 3.6, although some baseline method may achieve better results than ours on one of the evaluation criteria, they usually fail on the others. In contrast, our method has consistent high marks in all evaluation categories. The user study also shows the users overwhelmingly prefer our method to any of the baselines. We have also demonstrated several applications based on this technique and hope it may become a building block of more complex workflows.

3.6 Acknowledgement

The authors acknowledge the Maryland Advanced Research Computing Center for providing computing resources and acknowledge the photographers for licensing photos under Creative Commons or public domain.

Chapter 4: Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints

4.1 Introduction

In the last two decades, photorealistic image generation and manipulation techniques have rapidly evolved. Visual contents can now be easily created and edited without leaving obvious perceptual traces [156]. Recent breakthroughs in generative adversarial networks (GANs) [7, 65, 66, 145, 157, 158] have further improved the quality and photorealism of generated images. The adversarial framework of GANs can also be used in conditional scenarios for image translation [29, 30, 31] or manipulation in a given context [72, 159, 160, 161, 162], which diversifies media synthesis.

At the same time, however, the success of GANs has raised two challenges to the vision community: visual forensics and intellectual property protection.

GAN challenges to visual forensics. There is a widespread concern about the impact of this technology when used maliciously. This issue has also received increasing public attention, in terms of disruptive consequences to visual security, laws, politics, and society in general [163, 164, 165]. Therefore, it is critical to look into effective visual forensics against threats from GANs.

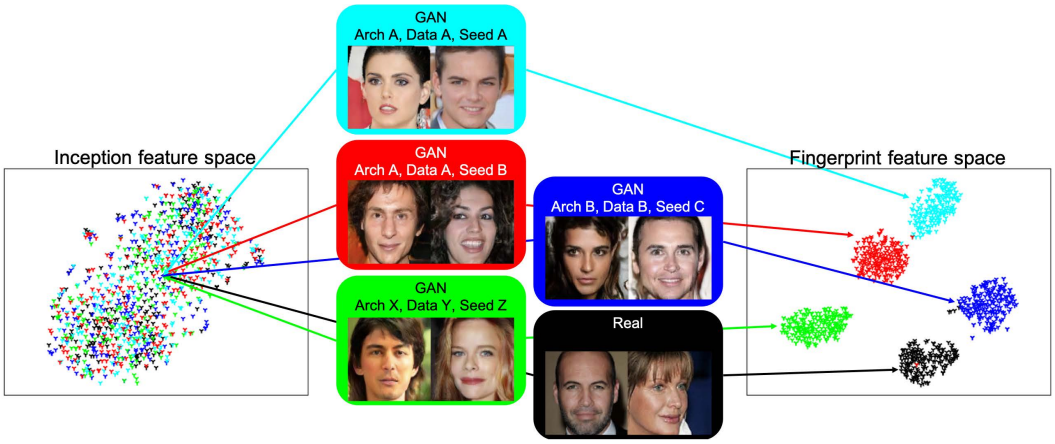


Figure 4.1: A t-SNE [6] visual comparison between our fingerprint features (right) and the baseline inception features [7] (left) for image attribution. Inception features are highly entangled, indicating the challenge to differentiate high-quality GAN-generated images from real ones. However, our result shows any single difference in GAN architectures, training sets, or even initialization seeds can result in distinct fingerprint features for effective attribution.

While recent state-of-the-art visual forensics techniques demonstrate impressive results for detecting fake visual media [166, 167, 168, 169, 170, 171, 172, 173, 174, 175], they have only focused on semantic, physical, or statistical inconsistency of specific forgery scenarios, e.g., copy-move manipulations[166, 175] or face swapping [173]. Forensics on GAN-generated images [176, 177, 178] shows good accuracy, but each method operates on only one GAN architecture by identifying its unique artifacts and results deteriorate when the GAN architecture is changed. It is still an open question of whether GANs leave stable marks that are commonly shared by their generated images. That motivates us to investigate an effective feature representation that differentiates GAN-generated images from real ones.

GAN challenges to intellectual property protection. Similar to other successful applications of deep learning technology to image recognition [179] or natural language processing [180], building a product based on GANs is non-trivial [181, 182, 183]. It

requires a large amount of training data, powerful computing resources, significant machine learning expertise, and numerous trial-and-error iterations for identifying optimal model architectures and their model hyper-parameters. As GAN services become widely deployed with commercial potential, they will become increasingly vulnerable to pirates. Such copyright plagiarism may jeopardize the intellectual property of model owners and take future market share from them. Therefore, methods for attributing GAN-generated image origins are highly desirable for protecting intellectual property.

Given the level of realism that GAN techniques already achieve today, attribution by human inspection is no longer feasible (see the mixture of Figure 4.4). The state-of-the-art digital identification techniques can be separated into two categories: digital watermarking and digital fingerprint detection. Neither of them is applicable to GAN attribution. Previous work on watermarking deep neural networks [184, 185] depends on an embedded security scheme during "white-box" model training, requires control of the input, and is impractical when only GAN-generated images are accessible in a "black-box" scenario. Previous work on digital fingerprints is limited to device fingerprints [186, 187] or in-camera post-processing fingerprints [188], which cannot be easily adapted to GAN-generated images. That motivates us to investigate GAN fingerprints that attribute different GAN-generated images to their sources.

We present the first study addressing the two GAN challenges simultaneously by learning GAN fingerprints for image attribution: We introduce GAN fingerprints and use them to classify an image as real or GAN-generated. For GAN-generated images, we further identify their sources. We approach this by training a neural network classifier and predicting the source of an image. Our experiments show that GANs carry distinct

model fingerprints and leave stable fingerprints in their generated images, which support image attribution.

We summarize our **contributions** as demonstrating the existence, uniqueness, persistence, immunizability, and visualization of GAN fingerprints. We address the following questions:

Existence and uniqueness: Which GAN parameters differentiate image attribution? We present experiments on GAN parameters including architecture, training data, as well as random initialization seed. We find that a difference in any one of these parameters results in a unique GAN fingerprint for image attribution. See Figure 7.1, Section 5.5.7 and 4.4.2.

Persistence: Which image components contain fingerprints for attribution? We investigate image components in different frequency bands and in different patch sizes. In order to eliminate possible bias from GAN artifact components, we apply a perceptual similarity metric to distill an artifact-free subset for attribution evaluation. We find that GAN fingerprints are persistent across different frequencies and patch sizes, and are not dominated by artifacts. See Section 4.3.1 and 4.4.3.

Immunizability: How robust is attribution to image perturbation attacks and how effective are the defenses? We investigate common attacks that aim at destroying image fingerprints. They include noise, blur, cropping, JPEG compression, relighting, and random combinations of them. We also defend against such attacks by finetuning our attribution classifier. See Section 4.4.4.

Visualization: How to expose GAN fingerprints? We propose an alternative classifier variant to explicitly visualize GAN fingerprints in the image domain, so as to better interpret the effectiveness of attribution. See Section 4.3.2 and 4.4.5.

Comparison to baselines. In terms of attribution accuracy, our method consistently outperforms three baseline methods (including a very recent one [189]) on two datasets under a variety of experimental conditions. In terms of feature representation, our fingerprints show superior distinguishability across image sources compared to inception features [7].

4.2 Related work

Generative Adversarial Networks (GANs). GANs [7, 65, 66, 145, 157, 158] have shown improved photorealism in image synthesis [127, 148, 190], translation [29, 30, 31], or manipulation [159, 160, 191]. We focus on unconditional GANs as the subject of our study. We choose the following four GAN models as representative candidates of the current state of the art: ProGAN [157], SNGAN [13], CramerGAN [192], and MMDGAN [154], considering their outstanding performances on face generation.

Visual forensics. Visual forensics targets detecting statistical or physics-based artifacts and then recognizing the authenticity of visual media without evidence from an embedded security mechanism [168, 193]. An example is a steganalysis-based method [194], which uses hand-crafted features plus a linear Support Vector Machine to detect forgeries. Recent CNN-based methods [169, 170, 171, 172, 173, 174, 175, 195, 196, 197] learn deep features and further improve tampering detection performance on images or videos. Rössler [198, 199] introduced a large-scale face manipulation dataset to benchmark forensics classification and segmentation tasks, and demonstrated superior performance when using additional domain-specific knowledge. For forensics on GAN-generated images, several existing works [176, 177, 178] show good accuracy. However, each

method considers only one GAN architecture and results do not generalize across architectures.

Digital fingerprints. Prior digital fingerprint techniques focus on detecting hand-crafted features for either device fingerprints or postprocessing fingerprints. The device fingerprints rely on the fact that individual devices, due to manufacturing imperfections, leave a unique and stable mark on each acquired image, i.e., the photo-response non-uniformity (PRNU) pattern [186, 187]. Likewise, postprocessing fingerprints come from the specific in-camera postprocessing suite (demosaicking, compression, etc.) during each image acquisition procedure [188]. Recently, Marra [189] visualize GAN fingerprints based on PRNU, and show their application to GAN source identification. We replace their hand-crafted fingerprint formulation with a learning-based one, decoupling model fingerprint from image fingerprint, and show superior performances in a variety of experimental conditions.

Digital watermarking. Digital watermarking is a complementary forensics technique for image authentication [200, 201, 202]. It involves embedding artificial watermarks in images. It can be used to reveal image source and ownership so as to protect their copyright. It has been shown that neural networks can also be actively watermarked during training [184, 185]. In such models, a characteristic pattern can be built into the learned representation but with a trade-off between watermarking accuracy and the original performance. However, such watermarking has not been studied for GANs. In contrast, we utilize inherent fingerprints for image attribution without any extra embedding burden or quality deterioration.

4.3 Fingerprint learning for image attribution

Inspired by the prior works on digital fingerprints [186, 188], we introduce the concepts of GAN model fingerprint and image fingerprint. Both are simultaneously learned from an image attribution task.

Model fingerprint. Each GAN model is characterized by many parameters: training dataset distribution, network architecture, loss design, optimization strategy, and hyper-parameter settings. Because of the non-convexity of the objective function and the instability of adversarial equilibrium between the generator and discriminator in GANs, the values of model weights are sensitive to their random initializations and do not converge to the same values during each training. This indicates that even though two well-trained GAN models may perform equivalently, they generate high-quality images differently. This suggests the existence and uniqueness of GAN fingerprints. We define the model fingerprint per GAN instance as a reference vector, such that it consistently interacts with all its generated images. In a specifically designed case, the model fingerprint can be an RGB image the same size as its generated images. See Section 4.3.2.

Image fingerprint. GAN-generated images are the outcomes of a large number of fixed filtering and non-linear processes, which generate common and stable patterns within the same GAN instances but are distinct across different GAN instances. That suggests the existence of image fingerprints and attributability towards their GAN sources. We introduce the fingerprint per image as a feature vector encoded from that image. In a specifically designed case, an image fingerprint can be an RGB image the same size as the original image. See Section 4.3.2.

Attribution network Similar to the authorship attribution task in natural language processing [203, 204], we train an attribution classifier that can predict the source of an image: real or from a GAN model.

We approach this using a deep convolutional neural network supervised by image-source pairs $\{(I, y)\}$ where $I \sim \mathbb{I}$ is sampled from an image set and $y \in \mathbb{Y}$ is the source ground truth belonging to a finite set. That set is composed of pre-trained GAN instances plus the real world. Figure 4.2(a) depicts an overview of our attribution network.

We implicitly represent image fingerprints as the final classifier features (the $1 \times 1 \times 512$ tensor before the final fully connected layer) and represent GAN model fingerprints as the corresponding classifier parameters (the $1 \times 1 \times 512$ weight tensor of the final fully connected layer).

Why is it necessary to use such an external classifier when GAN training already provides a discriminator? The discriminator learns a hyperplane in its own embedding space to distinguish generated images from real ones. Different embedding spaces are not aligned. In contrast, the proposed classifier necessarily learns a unified embedding space to distinguish generated images from different GAN instances or from real images.

Note that our motivation to investigate "white-box" GANs subject to known parameters is to validate the attributability along different GAN parameter dimensions. In practice, our method also applies to "black-box" GAN API services. The only required supervision is the source label of an image. We can simply query different services, collect their generated images, and label them by service indices. Our classifier would test image authenticity by predicting if an image is sampled from the desired service. We also test service authenticity by checking if most of their generated images have the desired source

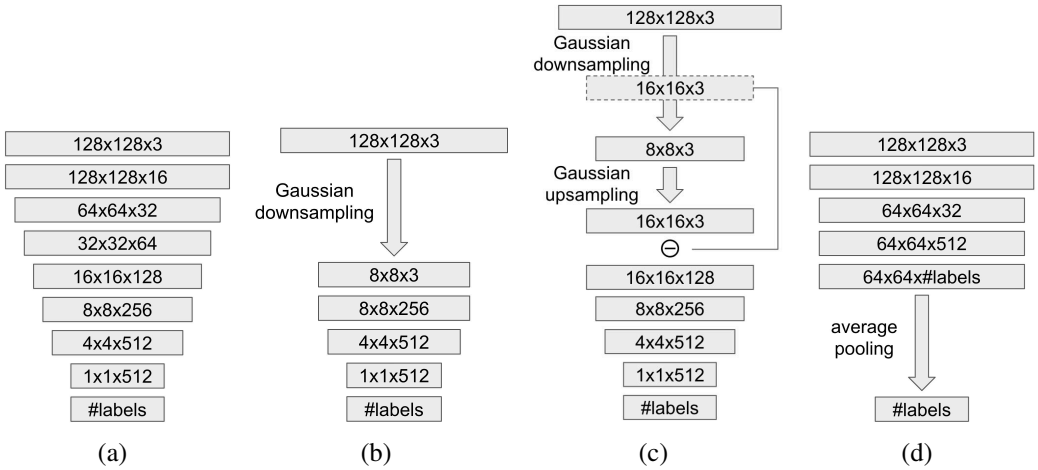


Figure 4.2: Different attribution network architectures. Tensor representation is specified by two spatial dimensions followed by the number of channels. The network is trained to minimize cross-entropy classification loss. (a) Attribution network. (b) Pre-downsampling network example that downsamples input image to 8×8 before convolution. (c) Pre-downsampling residual network example that extracts the residual component between 16×16 and 8×8 resolutions. (d) Post-pooling network example that starts average pooling at 64×64 resolution.

prediction.

4.3.1 Component analysis networks

In order to analyze which image components contain fingerprints, we propose three variants of the network.

Pre-downsampling network. We propose to test whether fingerprints and attribution can be derived from different frequency bands. We investigate attribution performance w.r.t. downsampling factor. Figure 4.2(b) shows an architecture example that extracts low-frequency bands. We replace the trainable convolution layers with our Gaussian downsampling layers from the input end and systematically control at which resolution we stop such replacement.

Pre-downsampling residual network. Complementary to extracting low-frequency

bands, Figure 4.2(c) shows an architecture example that extracts a residual high-frequency band between one resolution and its factor-2 downsampled resolution. It is reminiscent of a Laplacian Pyramid [205]. We systematically vary the resolution at which we extract such residual.

Post-pooling network. We propose to test whether fingerprints and attribution can be derived locally based on patch statistics. We investigate attribution performance w.r.t. patch size. Figure 4.2(d) shows an architecture example. Inspired by PatchGAN [29], we regard a "pixel" in a neural tensor as the feature representation of a local image patch covered by the receptive field of that "pixel". Therefore, post-pooling operations count for patch-based neural statistics. Earlier post-pooling corresponds to a smaller patch size. We systematically vary at which tensor resolution we start this pooling in order to switch between more local and more global patch statistics.

4.3.2 Fingerprint visualization

Alternatively to our attribution network in Section 5.5.7 where fingerprints are implicitly represented in the feature domain, we describe a model similar in spirit to Marra [189] to explicitly represent them in the image domain. But in contrast to their hand-crafted PRNU-based representation, we modify our attribution network architecture and learn fingerprint images from image-source pairs $(\{I, y\})$. We also decouple the representation of model fingerprints from image fingerprints. Figure 4.3 depicts the fingerprint visualization model.

Abstractly, we learn to map from input image to its fingerprint image. But without

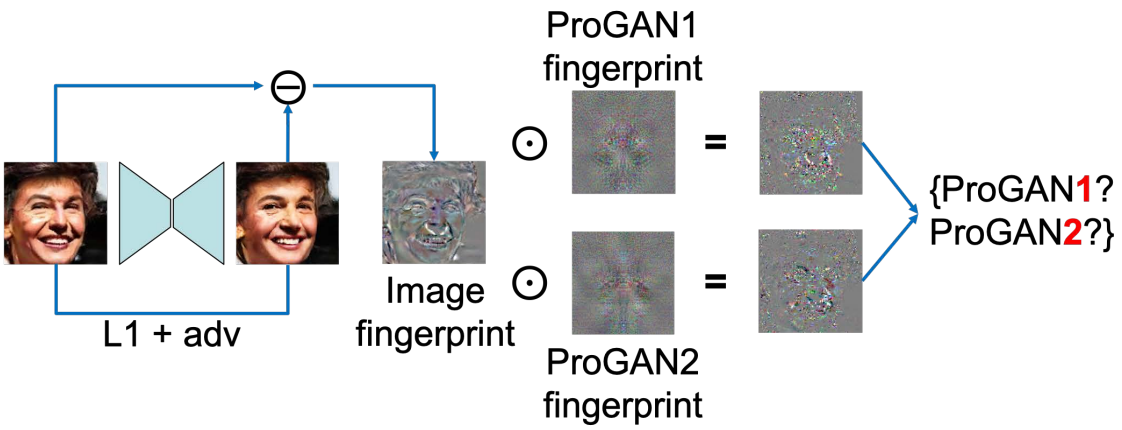


Figure 4.3: Fingerprint visualization diagram. We train an AutoEncoder and GAN fingerprints end-to-end. $\odot$ indicates pixel-wise multiplication of two normalized images.

fingerprint supervision, we choose to ground the mapping based on a reconstruction task with an AutoEncoder. We then define the reconstruction residual as the image fingerprint. We simultaneously learn a model fingerprint for each source (each GAN instance plus the real world), such that the correlation index between one image fingerprint and each model fingerprint serves as softmax logit for classification.

Mathematically, given an image-source pair (I, y) where $y \in \mathbb{Y}$ belongs to the finite set $\mathbb{Y}$ of GAN instances plus the real world, we formulate a reconstruction mapping R from I to $R(I)$. We ground our reconstruction based on pixel-wise L_1 loss plus adversarial loss:

$$L_{pix}(I) = ||R(I) - I||_1 \tag{4.1}$$

$$L_{adv}(I) = D_{rec}\big(R(I)\big) - D_{rec}\big(I\big) + GP\big(R(I), I|D_{rec}\big) \tag{4.2}$$

where D_{rec} is an adversarially trained discriminator, and $GP(\cdot)$ is the gradient penalty regularization term defined in [66].

We then explicitly define image fingerprint F^I_{im} as the reconstruction residual $F^I_{im} = R(I) - I$.

We further explicitly define model fingerprint F^y_{mod} as freely trainable parameters with the same size as F^I_{im}, such that $corr(F^I_{im}, F^y_{mod})$, the correlation index between F^I_{im} and F^y_{mod}, is maximized over $\mathbb{Y}$. This can be formulated as the softmax logit for the cross-entropy classification loss supervised by the source ground truth:

$$L_{cls}(I, y) = -\log \frac{corr(F^I_{im}, F^y_{mod})}{\sum_{\hat{y}\in\mathbb{Y}} corr(F^I_{im}, F^{\hat{y}}_{mod})} \tag{4.3}$$

where $corr(A, B) = \hat{A} \odot \hat{B}$, $\hat{A}$ and $\hat{B}$ are the zero-mean, unit-norm, and vectorized version of images A and B, and $\odot$ is the inner product operation.

Our final training objective is

$$\min_{R,\{F^{\tilde{y}}_{mod}|\tilde{y}\in\mathbb{Y}\}} \max_{D_{rec}} \mathop{\mathbb{E}}_{\{(I,y)\}} (\lambda_1 L_{pix} + \lambda_2 L_{adv} + \lambda_3 L_{cls}) \tag{4.4}$$

where $\lambda_1 = 20.0$, $\lambda_2 = 0.1$, and $\lambda_3 = 1.0$ are used to balance the order of magnitude of each loss term, which are not sensitive to dataset and are fixed.

Note that this network variant is used to better visualize and interpret the effectiveness of image attribution. However, it introduces extra training complexity and thus is not used if we only focus on attribution.

4.4 Experiments

We discuss the experimental setup in Section 7.4.1. From Section 4.4.2 to 4.4.5, we explore the four research questions discussed in the Introduction.

4.4.1 Setup

Datasets. We employ CelebA human face dataset [206] and LSUN bedroom scene dataset [207], both containing $20,000$ real-world RGB images.

GAN models. We consider four recent state-of-the-art GAN architectures: ProGAN [157], SNGAN [13], CramerGAN [192], and MMDGAN [154]. Each model is trained from scratch with their default settings except we fix the number of training epochs to 240 and fix the output size of a generator to $128 \times 128 \times 3$.

Baseline methods. Given real-world datasets and four pre-trained GAN models, we compare with three baseline classification methods: k-nearest-neighbor (kNN) on raw pixels, Eigenface [208], and the very recent PRNU-based fingerprint method from Marra [189].

Evaluation. We use classification accuracy to evaluate image attribution performance.

In addition, we use the ratio of inter-class and intra-class Fréchet Distance [209], denoted as FD ratio, to evaluate the distinguishability of a feature representation across classes. The larger the ratio, the more distinguishable the feature representation across sources. We compare our fingerprint features to image inception features [7]. The FD of inception features is also known as FID for GAN evaluation [118]. Therefore, the FD ratio of inception features can serve as a reference to show how challenging it is to

(a) CelebA real data

(b) ProGAN (c) SNGAN (d) CramerGAN (e) MMDGAN

Figure 4.4: Face samples from difference sources.

attribute high-quality GAN-generated images manually or without fingerprint learning.

4.4.2 Existence and uniqueness: which GAN parameters differentiate image attribution?

We consider GAN architecture, training set, and initialization seed respectively by varying one type of parameter and keeping the other two fixed.

Different architectures. First, we leverage all the real images to train ProGAN, SNGAN, CramerGAN, and MMDGAN separately. For the classification task, we configure training and testing sets with 5 classes: $\{$*real*, *ProGAN*, *SNGAN*, *CramerGAN*, *MMDGAN*$\}$. We randomly collect $100,000$ images from each source for classification training and another $10,000$ images from each source for testing. We show face samples from each source in Figure 4.4. Table 4.1 shows that we can effectively differentiate GAN-generated

images from real ones and attribute generated images to their sources, just using a regular CNN classifier. There do exist unique fingerprints in images that differentiate GAN architectures, even though it is far more challenging to attribute those images manually or through inception features [7].

Different GAN training sets. We further narrow down the investigation to GAN training sets. From now we only focus on ProGAN plus real dataset. We first randomly select a base real subset containing $100,000$ images, denoted as *real_subset_diff_0*. We then randomly select 10 other real subsets also containing $100,000$ images, denoted as *real_subset_diff_#i*, where $i \in \{1,\ 10,\ 100,\ 1000,\ 10000,\ 20000,\ 40000,\ 60000,\ 80000,\ 100000\}$ indicates the number of images that are not from the base subset. We collect such sets of datasets to explore the relationship between attribution performance and GAN training set overlaps.

For each *real_subset_diff_#i*, we separately train a ProGAN model and query $100,000$ images for classifier training and another $10,000$ images for testing, labeled as *ProGAN_subset_diff_#i*. In this setup of $\{$*real*, *ProGAN_subset_diff_#i*$\}$, we show the performance evaluation in Table 4.2. Surprisingly, we find that attribution performance remains equally high regardless of the amount of GAN training set overlap. Even GAN training sets that differ in just one image can lead to distinct GAN instances. That indicates that one-image mismatch during GAN training results in a different optimization step in one iteration and finally results in distinct fingerprints. That motivates us to investigate the attribution performance among GAN instances that were trained with identical architecture and dataset but with different random initialization seeds.

Different initialization seeds. We next investigate the impact of GAN training

		CelebA	LSUN
Accuracy (%)	kNN	28.00	36.30
	Eigenface [208]	53.28	-
	PRNU [189]	86.61	67.84
	Ours	**99.43**	**98.58**
FD ratio	Inception [7]	2.36	5.27
	Our fingerprint	**454.76**	**226.59**

Table 4.1: Evaluation on {*real*, *ProGAN*, *SNGAN*, *CramerGAN*, *MMDGAN*}. The best performance is highlighted in **bold**.

		CelebA	LSUN
Accuracy (%)	kNN	11.46	10.72
	Eigenface [208]	27.98	-
	PRNU [189]	92.28	70.55
	Ours	**99.50**	**97.66**
FD ratio	Inception [7]	1.08	1.64
	Our fingerprint	**111.41**	**39.96**

Table 4.2: Evaluation on {*real*, *ProGAN_subset_diff_#i*}. The best performance is highlighted in **bold**.

initialization on image attributability. We train 10 ProGAN instances with the entire real dataset and with different initialization seeds. We sample $100,000$ images for classifier training and another $10,000$ images for testing. In this setup of {*real*, *ProGAN_seed_v#i*} where $i \in \{1, ..., 10\}$, we show the performance evaluation in Table 4.3. We conclude that it is the difference in optimization (e.g., caused by different randomness) that leads to attributable fingerprints. In order to verify our experimental setup, we ran sanity checks. For example, two identical ProGAN instances trained with the same seed remain indistinguishable and result in random-chance attribution performance.

		CelebA	LSUN
Accuracy (%)	kNN	10.88	10.58
	Eigenface [208]	23.12	-
	PRNU [189]	89.40	69.73
	Ours	**99.14**	**97.04**
	Our visNet	97.07	96.58
FD ratio	Inception [7]	1.10	1.29
	Our fingerprint	**80.28**	**36.48**

Table 4.3: Evaluation on {*real*, *ProGAN_seed_v#i*}. The best performance is highlighted in **bold**. "Our visNet" row indicates our fingerprint visualization network described in Section 4.3.2 and evaluated in Section 4.4.5.

4.4.3 Persistence: which image components contain fingerprints for attribution?

We systematically explore attribution performance w.r.t. image components in different frequency bands or with different patch sizes. We also investigate possible performance bias from GAN artifacts.

Different frequencies. We investigate if band-limited images carry effective fingerprints for attribution. We separately apply the proposed pre-downsampling network and pre-downsampling residual network for image attribution. Given the setup of {*real*, *ProGAN_seed_v#i*}, Table 4.4 shows the classification accuracy w.r.t. downsampling factors. We conclude that (1) a wider frequency band carries more fingerprint information for image attribution, (2) the low-frequency and high-frequency components (even at the resolution of 8×8) individually carry effective fingerprints and result in attribution performance better than random, and (3) at the same resolution, high-frequency components carry more fingerprint information than low-frequency components.

Different local patch sizes. We also investigate if local image patches carry effective fingerprints for attribution. We apply the post-pooling network for image attribution.

Downsample factor	Resolution	CelebA		LSUN	
		L-f	H-f	L-f	H-f
1	128^2	99.14	99.14	97.04	97.04
2	64^2	98.74	98.64	96.78	96.84
4	32^2	95.50	98.52	91.08	96.04
8	16^2	87.20	92.90	83.02	91.58
16	8^2	67.44	78.74	63.80	80.58
32	4^2	26.58	48.42	28.24	54.50

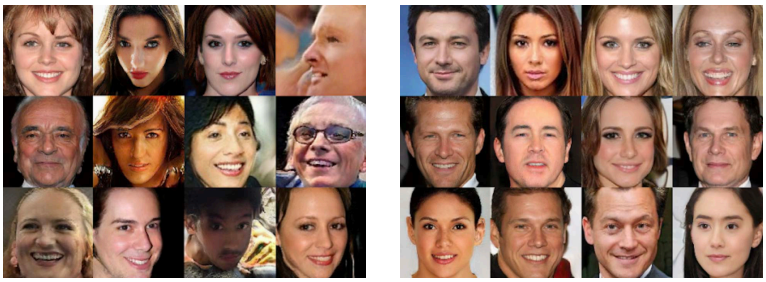
Table 4.4: Classification accuracy (%) of our network w.r.t. downsampling factor on low-frequency or high-frequency components of {*real*, *ProGAN_seed_v#i*}. "L-f" column indicates the low-frequency components and represents the performances from the pre-downsampling network. "H-f" column indicates the high-frequency components and represents the performances from the pre-downsampling residual network.

Pooling starts at	Patch size	CelebA	LSUN
4^2	128^2	99.34	97.44
8^2	108^2	99.32	96.30
16^2	52^2	99.30	95.94
32^2	24^2	99.24	88.36
64^2	10^2	89.60	18.26
128^2	3^2	13.42	17.10

Table 4.5: Classification accuracy (%) of our network w.r.t. patch size on {*real*, *ProGAN_seed_v#i*}.

Given the setup of {*real*, *ProGAN_seed_v#i*}, Table 4.5 shows the classification accuracy w.r.t. patch sizes. We conclude that for CelebA face dataset a patch of size 24×24 or larger carries sufficient fingerprint information for image attribution without deterioration; for LSUN, a patch of size 52×52 or larger carries a sufficient fingerprint.

Artifact-free subset. Throughout our experiments, the state-of-the-art GAN approaches are capable of generating high-quality images – but are also generating obvious artifacts in some cases. There is a concern that attribution might be biased by such artifacts. In order to eliminate this concern, we use Perceptual Similariy [8] to measure the 1-nearest-neighbor similarity between each testing generated image and the real-world dataset, and



(a) Non-selected samples (b) Selected samples

Figure 4.5: Visual comparisons between (a) arbitrary face samples and (b) selected samples with top 10% Perceptual Similarity [8] to CelebA real dataset. We notice the selected samples have higher quality and fewer artifacts. They are also more similar to each other, which challenge more on attribution.

then select the 10% with the highest similarity for attribution. We compare face samples between non-selected and selected sets in Figure 4.5. We notice this metric is visually effective in selecting samples of higher quality and with fewer artifacts.

Given the setup of 10% selected $\{real, ProGAN\_seed\_v\#i\}$, we show the performance evaluation in Table 4.6. All the FD ratio measures consistently decreased compared to Table 4.3. This indicates our selection also moves the image distributions from different GAN instances closer to the real dataset and consequently closer to each other. This makes the attribution task more challenging. Encouragingly, our classifier, pre-trained on non-selected images, can perform equally well on the selected high-quality images and is hence not biased by artifacts.

4.4.4 Immunizability: how robust is attribution to image perturbation attacks and how effective are the defenses?

Attacks. We apply five types of attacks that perturb testing images [210]: *noise*, *blur*, *cropping*, *JPEG compression*, *relighting*, and random combination of them. The

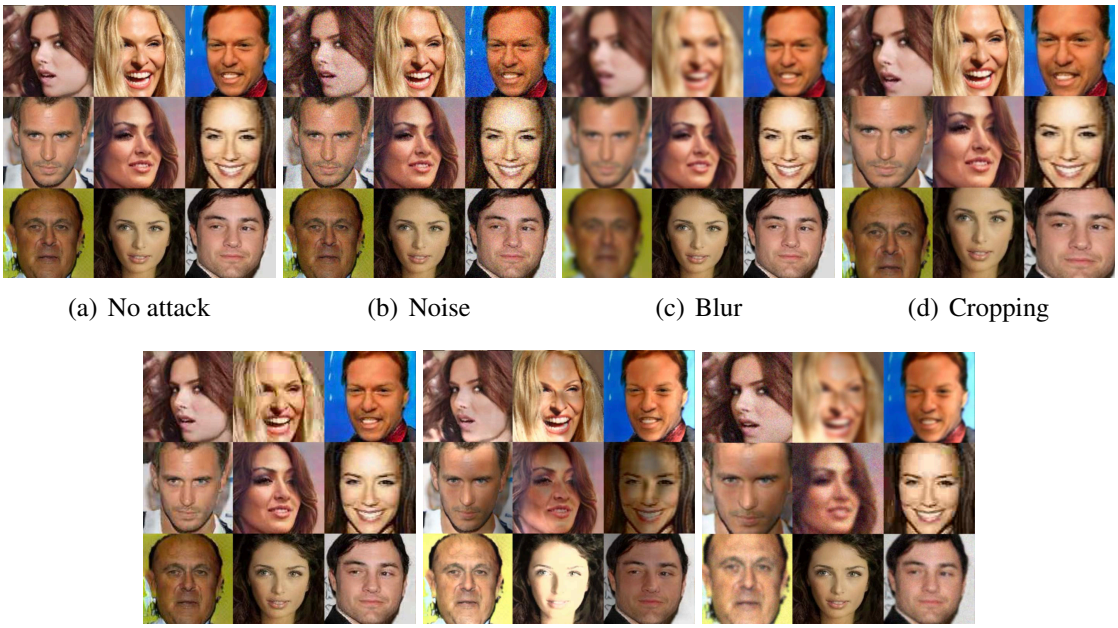
		CelebA	LSUN
Accuracy (%)	kNN	11.99	10.35
	Eigenface [208]	26.69	-
	PRNU [189]	93.50	74.49
	Ours	**99.93**	**98.16**
FD ratio	Inception [7]	1.04	1.22
	Our fingerprint	**15.63**	**6.27**

Table 4.6: Evaluation on the 10% selected images of {*real*, *ProGAN_seed_v#i*}. The best performance is highlighted in **bold**.

intention is to confuse the attribution network by destroying image fingerprints. Examples of the perturbations on face images are shown in Figure 4.6.

Noise adds i.i.d. Gaussian noise to testing images. The Gaussian variance is randomly sampled from $U[5.0, 20.0]$. *Blur* performs Gaussian filtering on testing images with kernel size randomly picked from $\{1,\ 3,\ 5,\ 7,\ 9\}$. *Cropping* crops testing images with a random offset between 5% and 20% of the image side lengths and then resizes back to the original. *JPEG compression* performs JPEG compression processing with quality factor randomly sampled from $U[10, 75]$. *Relighting* uses SfSNet [211] to replace the current image lighting condition with another random one from their lighting dataset. The combination performs each attack with a 50% probability in the order of *relighting*, *cropping*, *blur*, *JPEG compression*, and *noise*.

Given perturbed images and the setup of {*real*, *ProGAN_seed_v#i*}, we show the pre-trained classifier performances in the "Akt" columns in Table 4.7 and Table 4.8. All performances decrease due to attacks. In detail, the classifier completely fails to overcome *noise* and *JPEG compression* attacks. It still performs better than random when facing the other four types of attacks. The *relighting* attack is the least effective one because it only



(a) No attack (b) Noise (c) Blur (d) Cropping

(e) Compression (f) Relighting (g) Combination

Figure 4.6: Image samples for the attacks and defenses of our attribution network.

perturbs low-frequency image components. The barely unchanged fingerprints in high-frequency components enables reasonable attribution.

Defenses. In order to immunize our classifier against attacks, we finetune the classifier under the assumption that we know the attack category. Given perturbed images and the setup of {*real*, *ProGAN_seed_v#i*}, we show the finetuned classifier performance in the "Dfs" columns in Table 4.7 and Table 4.8. It turns out that the immunized classifier completely regains performance over *blur*, *cropping* and *relighting* attacks, and partially regains performance over the others. However, the recovery from *combination* attack is minimal due to its highest complexity. In addition, our method consistently outperforms the method of Marra [189] under each attack after immunization, while theirs does not effectively benefit from such immunization.

	CelebA											
	Noise		*Blur*		*Cropping*		*Compression*		*Relighting*		*Combination*	
	Atk	Dfs	Atk	Dfs	Atk	Dfs	Atk	Dfs	Atk	Dfs	Atk	Dfs
PRNU [189]	**57.88**	63.82	27.37	42.43	9.84	10.68	**26.15**	44.55	86.59	87.02	**19.93**	21.77
Ours	9.14	**93.02**	**49.64**	**97.20**	**46.80**	**98.28**	8.77	**88.02**	**94.02**	**98.66**	19.31	**72.64**

Table 4.7: Classification accuracy (%) of our network w.r.t. different perturbation attacks before or after immunization on CelebA {*real*, *ProGAN_seed_v#i*}. The best performance is highlighted in **bold**.

	LSUN											
	Noise		*Blur*		*Cropping*		*Compression*		*Relighting*		*Combination*	
	Atk	Dfs	Atk	Dfs	Atk	Dfs	Atk	Dfs	Atk	Dfs	Atk	Dfs
PRNU [189]	**39.59**	40.97	26.92	30.79	9.30	9.42	18.27	23.66	60.86	63.31	16.54	16.89
Ours	11.80	**95.30**	**74.48**	**96.68**	**86.20**	**97.30**	**24.73**	**92.40**	**62.21**	**97.36**	**24.44**	**83.42**

Table 4.8: Classification accuracy (%) of our network w.r.t. different perturbation attacks before or after immunization on LSUN bedroom {*real*, *ProGAN_seed_v#i*}. The best performance is highlighted in **bold**.

4.4.5 Fingerprint visualization

Given the setup of {*real*, *ProGAN_seed_v#i*}, we alternatively apply the fingerprint visualization network (Section 4.3.2) to attribute images. We show the attribution performance in the "Our visNet" row in Table 4.3, which are comparable to that of the attribution model. Figure 4.7 visualizes face fingerprints. It turns out that image fingerprints maximize responses only to their own model fingerprints, which supports effective attribution. To attribute the real-world image, it is sufficient for the fingerprint to focus only on the eyes. To attribute the other images, the fingerprints also consider clues from the background, which, compared to foreground faces, is more variant and harder for GANs to approximate realistically [212].

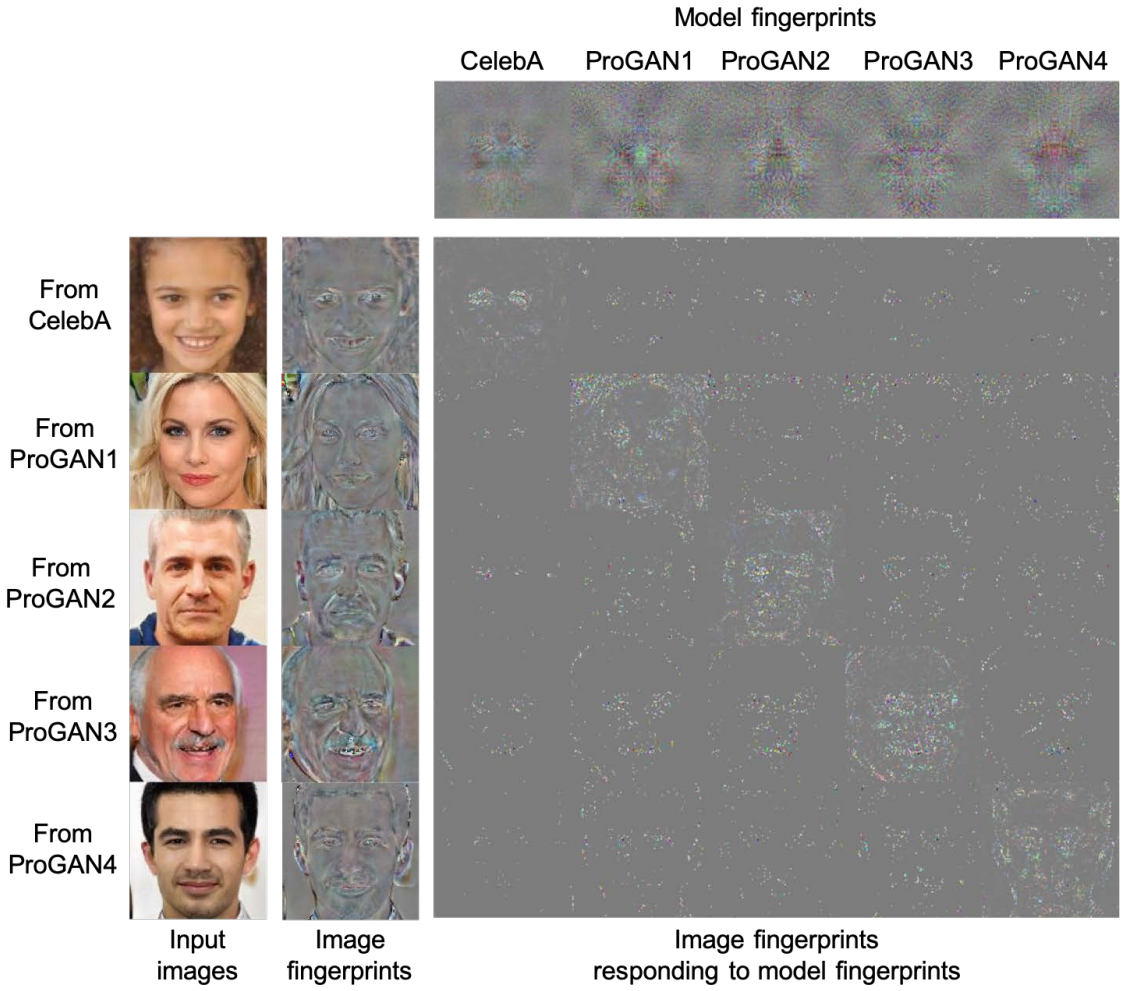


Figure 4.7: Visualization of model and image fingerprint samples. Their pairwise interactions are shown as the confusion matrix.

4.5 Conclusion

We have presented the first study of learning GAN fingerprints towards image attribution. Our experiments show that even a small difference in GAN training (e.g., the difference in initialization) can leave a distinct fingerprint that commonly exists over all its generated images. That enables fine-grained image attribution and model attribution. Further encouragingly, fingerprints are persistent across different frequencies and different patch sizes, and are not biased by GAN artifacts. Even though fingerprints can be deteriorated by several image perturbation attacks, they are effectively immunizable by simple finetuning. Comparisons also show that, in a variety of conditions, our learned fingerprints are consistently superior to the very recent baseline [189] for attribution, and consistently outperform inception features [7] for cross-source distinguishability.

4.6 Acknowledgement

This project was partially funded by DARPA MediFor program under cooperative agreement FA87501620191. We acknowledge the Maryland Advanced Research Computing Center for providing computing resources. We thank Hao Zhou for helping with the relighting experiments. We also thank Yaser Yacoob and Abhinav Shrivastava for constructive advice in general.

Chapter 5: Artificial Fingerprinting for Generative Models: Rooting Deepfake Attribution in Training Data

5.1 Introduction

In the past years, photorealistic image generation has been rapidly evolving, benefiting from the invention of generative adversarial networks (GANs) [9] and its successive breakthroughs [10, 12, 13, 14, 15, 16, 17]. Given the level of realism and diversity that generative models can achieve today, detecting generated media, well known as *deepfakes*, attributing their sources, and tracing their legal responsibilities become infeasible to human beings.

Moreover, the misuse of deepfakes has been permeating to each corner of social media, ranging from misinformation of political campaigns [213] to fake journalism [214, 215]. This motivates tremendous research efforts on deepfake detection [216] and source attribution [1, 2, 189]. These techniques aim to counter the widespread of malicious applications of deepfakes by automatically identifying and flagging generated visual contents and tracking their sources. Most of them rely on low-level visual patterns in GAN-generated images [1, 2, 189] or frequency mismatch [217, 218, 219]. However, these techniques are unable to sustainably and robustly prevent deepfake misuse in the long run;

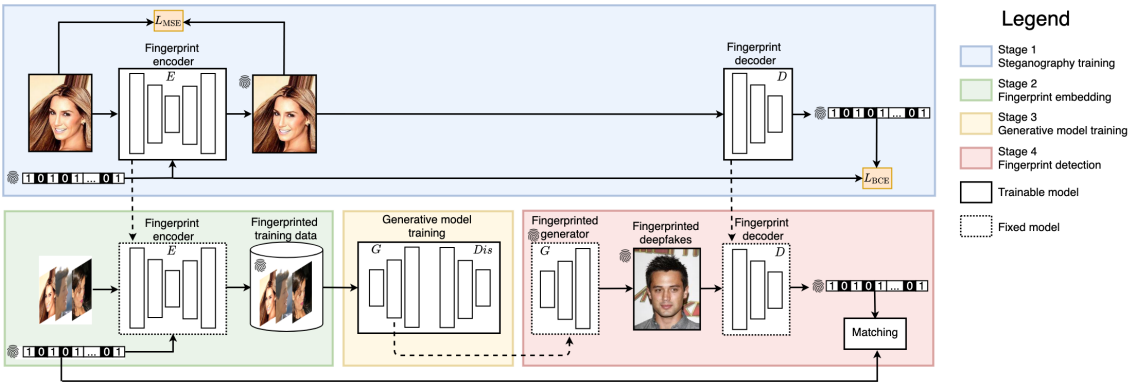


Figure 5.1: Our solution pipeline consists of four stages. We first train an image steganography encoder and decoder. Then we use the encoder to embed artificial fingerprints into the training data. After that, we train a generative model with its original protocol. Finally, we decode the fingerprints from the generated deepfakes.

as generative models evolve, they learn to better match the true distribution causing fewer artifacts [216]. Besides, detection countermeasures are also continuously evolving [216, 220, 221].

Motivated by this, we tackle deepfake detection and attribution through a different lens, and propose a ***proactive*** and sustainable solution for detection, which is simple and effective. In specific, we aim to introduce ***artificial fingerprints*** into generative models that enable identification and tracing. Figure 7.1 depicts our pipeline; we first embed artificial fingerprints into the training data using image steganography [222, 223]. The generative model is then trained with its original protocol without modification. This makes our solution agnostic and plug-and-play for arbitrary models. We then show a surprising discovery on the ***transferability*** of such fingerprints from training data to the model: the same fingerprint information that was encoded in the training data can be decoded from all generated images.

We achieve deepfake detection by classifying images with matched fingerprints

in our database as fake and images with random detected fingerprints as real. We also achieve deepfake attribution when we allocate different fingerprints for different generative models. Our solution thus closes the responsibility loop between generative model inventions and their possible misuses. It prevents the misuse of published pre-trained generative models by enabling inventors to proactively and responsibly embed artificial fingerprints into the models.

We summarize our contributions as follow:

(1) We synergize the two previously uncorrelated domains, image steganography and GANs, and propose the first ***proactive*** and sustainable solution for the third emerging domain, deepfake detection and attribution.

(2) This is the first study to demonstrate the ***transferability*** of artificial fingerprints from training data to generative models and then to all the generated deepfakes. Our discovery is non-trivial: only deep-learning-based fingerprinting techniques [222, 223] are transferable to generative models, while conventional steganography and watermarking techniques [224, 225] are not. See Section 5.5.2 for comparisons.

(3) We empirically validate several beneficial properties of our solution. ***Universality*** (Section 5.5.2): it holds for a variety of cutting-edge generative models [14, 15, 16, 17, 226]. ***Fidelity*** (Section 5.5.3): it has a negligible side effect on generation quality. ***Robustness*** (Section 5.5.4): it stays robust against many perturbations. ***Secrecy*** (Section 5.5.5): the artificial fingerprints are hard to be detected by adversaries. ***Anti-deepfake*** (Section 5.5.6 and 5.5.7): it converts deepfake detection and attribution into trivial tasks and outperforms the state-of-the-art baselines [1, 2].

5.2 Related Work

Generative adversarial networks (GANs). GANs [9] was first proposed as a workaround to model the intractable real data distribution. The iterative improvements push the generation realism to brand-new levels [10, 12, 13, 14, 15, 16, 17]. Successes have also spread to many other vision tasks (e.g. [5, 29, 30, 31, 34, 37, 226]). In Section 5.5, we focus on three categories of cutting-edge generative models: unconditional (ProGAN [15], StyleGAN [16], and StyleGAN2 [17]), class-conditional (BigGAN [14]), and image-conditional (image-to-image translation) (CUT [226]).

Image steganography and watermarking. Image steganography and watermarking hide information into carrier images [227]. Previous techniques rely on Fourier transform [228, 229], JPEG compression [224, 225], or least significant bits modification [230, 231, 232]. Recent works replace hand-crafted hiding procedures with neural network encoding [222, 223, 233, 234, 235, 236, 237]. We leverage recent deep-learning-based steganography methods [222, 223] to embed artificial fingerprints into training data, and validate their transferability to generative models. This is non-trivial because only deep-learning-based fingerprints are transferable to generative models, while conventional ones [224, 225] are not (Section 5.5.2). Besides, the stealthiness achieved by steganography allows preserving the original generation quality (Section 5.5.3) and fingerprint secrecy (Section 5.5.5).

Our fingerprinting is conceptually and functionally orthogonal to all of them. Instead of encoding information into pixels of individual images, our solution encodes information into generator parameters such that all the generated images are entangled with that information. Compared to the pipeline of a generator followed by a watermarking module,

our solution introduces zero generation overheads, and obstructs adversarial model surgery that targets to detach watermarking from image generation.

Network watermarking. Different from image watermarking, network watermarking targets to hide information into model parameters without affecting its original performances, similar in spirit to our goal. There are two categories of them: black-box trigger-set-based solutions [184, 238], and white-box feature-based solutions [185, 239, 240]. The former ones embed watermarks through a trigger set of input and decodes watermarks according to the input-output behavior of the model. The latter ones directly embed watermarks in the model parameter space with transformation matrices. It is worth noting that our solution renders conceptual and technical distinctions from network watermarking. In terms of concepts, the previous works target to only discriminative models (e.g., classification), while a solution for generative models is urgently lacking. In terms of techniques, to adapt to generator watermarking, we tune our solution to *indirectly* transfers fingerprints from training data to model parameters. This is because (1) unconditional generative models do not allow deterministic input so that a trigger set is not applicable, and (2) transformations in the parameter space are not agnostic to model configurations so that they are neither scalable nor sustainable along with the evolution of generative models.

Deepfake detection and attribution. Images generated by GAN models bear unique patterns. [189] shows that generative models leave unique noise residuals to generated samples, which allows deepfake detection. [2] moves one step further, using a neural network classifier to attribute different images to their sources. [1] also train a classifier and improve the generalization across different generation techniques. [217, 218, 220] point out that the high-frequency pattern mismatch can be used for deepfake

detection, so can the texture feature mismatch [241]. However, these cues are not sustainable due to the advancement of detection countermeasures. For example, spectral regularization [220] is proposed to narrow down the frequency mismatch and results in a significant detection deterioration. Also, detectors [1] are vulnerable to adversarial evasion attacks [221].

In contrast to the previous passive approaches, we propose a novel *proactive* solution for model fingerprinting and, thus, for deepfake detection. We differentiate between our term *artificial fingerprints* which refers to the information we deliberately and proactively embed into the model, and the term GAN fingerprints [2] which refers to the inherent cues and artifacts of different GAN models. Our work is also distinct from a follow-up proactive technique [242]. They focus on fingerprinting scalability and efficiency while we focus more fundamentally on its transferability and universality.

5.3 Problem Statement

Generation techniques can be misused to create misinformation at scale to achieve financial or political gains. Recently, there have been concerns about releasing generative models. For example, OpenAI employed a staged release to evaluate the potential risks of their GPT-2 model [243]. GPT-3 was later released as a black-box API only [244]. Face2Face [160] authors did not open their sources for real-time face capture and reenactment.

We design solution from the model inventors' side (e.g., OpenAI). Our solution introduces traceable artificial fingerprints in generative models. It enables deepfake detection and attribution by decoding the fingerprints from the generated images and matching them to the known fingerprints given to different models. This equips model inventors

with a means for a proactive and responsible disclosure when publishing their pre-trained models. This distinguishes our model fingerprinting solution from watermarking the generated images: we aim to defend against the misuse of published generative models rather than single deepfake media.

In practice, the training is done by the model inventor. Responsible model inventors, different from malicious deepfake users, should be *eager/willing* to adopt a proactive solution to fingerprint their generative models against potential deepfake misuses. The fingerprinting encoder and decoder, and the unique fingerprints given to different models, are privately maintained by the model inventor. Once a deepfake misuse happens, the inventor is able to verify if this is generated by one of their models. If so, they can further attribute by which model user. Then they can prohibit that user's accessibility to the model and/or seek legal regulations. Thus, they can claim responsible disclosure with a countermeasure against potential misuse when they publish their models.

5.4 Artificial Fingerprints

The goal of image attribution is to learn a mapping $D_0(\mathbf{x}) \mapsto y$ that traces the source $y \in \mathbb{Y} = \{\text{real}, \text{G}_1, \ldots, \text{G}_\text{N}\}$ of an image $\mathbf{x}$. If the domain $\mathbb{Y}$ is limited, predefined, and known to us, this is a closed-world scenario and the attribution can be simply formulated as a multi-label classification problem, each label corresponding to one source, as conducted in [2]. However, $\mathbb{Y}$ can be unlimited, undefined, continuously evolving, and agnostic to us. This open-world scenario is intractable using discriminative learning. To generalize our solution to being agnostic to the selection of generative models, we formulate the

attribution as a regression mapping $D(\mathbf{x}) \mapsto \mathbf{w}$, where $\mathbf{w} \in \{0,1\}^n$ is the source identity space and n is the dimension. We propose a pipeline to root the attribution down to the training dataset $\tilde{\mathbf{x}} \in \tilde{\mathbb{X}}$ and close the loop of the regression D. We describe the pipeline stages (depicted in Figure 7.1) below:

Steganography training. The source identity is represented by the artificial fingerprints $\mathbf{w}$. We use a steganography system [222, 223] to learn an encoder $E(\tilde{\mathbf{x}}, \mathbf{w}) \mapsto \tilde{\mathbf{x}}_{\mathbf{w}}$ that embeds an arbitrary fingerprint $\mathbf{w}$ (randomly sampled during training) into an arbitrary image $\tilde{\mathbf{x}}$. We couple E with a decoder $D(\tilde{\mathbf{x}}_{\mathbf{w}}) \mapsto \mathbf{w}$ to detect the fingerprint information from the image. E and D are formulated as convolutional neural networks with the following training losses:

$$\min_{E,D} \mathbb{E}_{\tilde{\mathbf{x}} \sim \tilde{\mathbb{X}}, \mathbf{w} \sim \{0,1\}^n} L_{\text{BCE}}(\tilde{\mathbf{x}}, \mathbf{w}; E, D) + \lambda L_{\text{MSE}}(\tilde{\mathbf{x}}, \mathbf{w}; E) \tag{5.1}$$

$$L_{\text{BCE}}(\tilde{\mathbf{x}}, \mathbf{w}; E, D) = \frac{1}{n} \sum_{k=1}^{n} \big(\mathbf{w}_k \log \hat{\mathbf{w}}_k + (1 - \mathbf{w}_k) \log(1 - \hat{\mathbf{w}}_k) \big) \tag{5.2}$$

$$L_{\text{MSE}}(\tilde{\mathbf{x}}, \mathbf{w}; E) = ||E(\tilde{\mathbf{x}}, \mathbf{w}) - \tilde{\mathbf{x}}||_2^2 \tag{5.3}$$

$$\hat{\mathbf{w}} = D\big(E(\tilde{\mathbf{x}}, \mathbf{w})\big) \tag{5.4}$$

where $\mathbf{w}_k$ and $\hat{\mathbf{w}}_k$ are the k^{th} bit of the input fingerprint and detected fingerprint separately; and λ is a hyper-parameter to balance the two objective terms. The binary cross-entropy term L_{BCE} guides the decoder to decode the fingerprint embedded by the encoder. The

mean squared error term L_{MSE} penalizes any deviation of the stego image $E(\tilde{\mathbf{x}}, \mathbf{w})$ from the original image $\tilde{\mathbf{x}}$.

Artificial fingerprint embedding. In this stage, we use the well trained E and D networks. We allocate each training dataset $\tilde{\mathbb{X}}$ a unique fingerprint $\mathbf{w}$. We apply the trained E to each training image $\tilde{\mathbf{x}}$ and collect a fingerprinted training dataset $\tilde{\mathbb{X}}_{\mathbf{w}} = \{E(\tilde{\mathbf{x}}, \mathbf{w}) | \tilde{\mathbf{x}} \in \tilde{\mathbb{X}}\}$.

Generative model training. In order to have a solution that is agnostic to the evolution of generative models, we intentionally do not intervene with their training. It makes our solution plug-and-play for arbitrary generation tasks without touching their implementations, and introduces zero overhead to model training. We simply replace $\tilde{\mathbb{X}}$ with $\tilde{\mathbb{X}}_{\mathbf{w}}$ to train the generative model in its original protocol.

Artificial fingerprint decoding. We hypothesize the *transferability* of our artificial fingerprints from training data to generative models: a well-trained generator $G_{\mathbf{w}}(\mathbf{z}) \mapsto \mathbf{x}_{\mathbf{w}}$ contains, in all generated images, the same fingerprint information $\mathbf{w}$ (as embedded in the training data $\tilde{\mathbf{x}}_{\mathbf{w}}$). We justify this hypothesis in Section 5.5.2. As a result, the artificial fingerprint can be recovered from a generated image $\mathbf{x}_{\mathbf{w}}$ using the decoder D: $D(\mathbf{x}_{\mathbf{w}}) \equiv \mathbf{w}$. Based on this transferability, we can formulate deepfake attribution as fingerprint matching using our decoder D.

Artificial fingerprint matching. To support robustness to post-generation modifications that could be applied to the generated images, we relax the matching of the decoded artificial fingerprints to a soft matching. We perform a null hypothesis test given the number of matching bits k between the decoded fingerprint $\tilde{\mathbf{w}}$ and the fingerprint $\mathbf{w}$ used in generative model training. The null hypothesis H_0 is getting this number of successes

(i.e. matching bits) by chance. Under the null hypothesis, the probability of matching bits (random variable X) follows a binomial distribution: the number of trials n is the number of bits in the fingerprint sequence, and k is the number of successes where each bit has a 0.5 probability of success. We can then measure the p-value of the hypothesis test by computing the probability of getting k or higher matching bits under the null hypothesis:

$$Pr(X > k|H_0) = \sum_{i=k}^{n} \binom{n}{i} 0.5^n \tag{5.5}$$

The fingerprint is verified, $\tilde{\mathbf{w}} \sim \mathbf{w}$, if the null hypothesis results in a very low probability (p-value). Usually, when the p-value is smaller than 0.05, we reject the null hypothesis and regard $1 - p$ as the verification confidence.

5.5 Experiments

We describe the experimental setup in Section 7.4.1. We first evaluate the required proprieties of our solution: the transferability and universality of our artificial fingerprint in Section 5.5.2, its fidelity in Section 5.5.3, its robustness in Section 5.5.4, and its secrecy in Section 5.5.5. The transferability in turn enables accurate deepfake detection and attribution, which is evaluated and compared in Section 5.5.6 and 5.5.7 respectively.

5.5.1 Setup

Generative models. As a proactive solution, it should be agnostic to genetative models. Without losing representativeness, we focus on three generation applications with their state-of-the-art models. For unconditional generation: ProGAN [15], StyleGAN [16],

and StyleGAN2 [17]; for class-conditional generation: BigGAN [14]; for image-conditional generation, i.e., image-to-image translation: CUT [226]. Each model is trained from scratch with the official implementation.

Datasets. Each generation application benchmarks its own datasets. For unconditional generation, we train/test on 150k/50k CelebA [206] at 128×128 resolution, 50k/50k LSUN *Bedroom* [207] at 128×128 resolution, and the most challenging one, 50k/50k LSUN *Cat* [207] at its original 256×256 resolution. For class-conditional generation, we experiment on the entire CIFAR-10 dataset [245] with the original training/testing split at the original 32×32 resolution. For image-conditional generation, we experiment on the entire *Horse→Zebra* dataset [30] and *Cat→Dog* [45] dataset with the original training/testing split at the original 256×256 resolution. We only need to fingerprint images from the target domains.

5.5.2 Transferability

The transferability means that the artificial fingerprints that are embedded in the training data also appear consistently in all the generated data. This is a non-trivial hypothesis in Section 5.4 and needs to be justified by the fingerprint detection accuracy.

Evaluation. Fingerprints are represented as binary vectors $\mathbf{w} \in \{0,1\}^n$. We use bitwise accuracy to evaluate the detection accuracy. We set $n = 100$ as suggested in [223]. We also report p-value for the confidence of detection.

Baselines. For comparison, we implement a straightforward baseline method. Instead of embedding fingerprints into training data, we enforce fingerprint generation jointly

with model training. That is, we train on clean data, and enforce generated images to not only approximate real training images but also contain a specific fingerprint. Mathematically,

$$\min_{G,D} \max_{Dis} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\tilde{\mathbf{x}}\sim\tilde{\mathbb{X}}} L_{\text{adv}}(\mathbf{z}, \tilde{\mathbf{x}}; G, Dis) + \eta \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{w}\sim\{0,1\}^n} L_{\text{BCE}}(\mathbf{z}, \mathbf{w}; G, D) \tag{5.6}$$

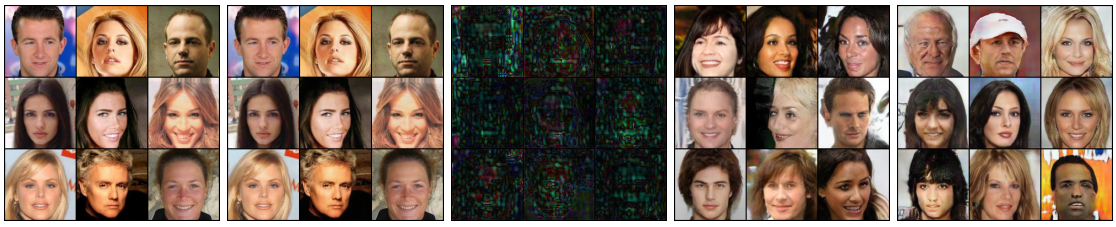
where G and Dis are the original generator and discriminator in the GAN framework, L_{adv} is the original GAN objective, and L_{BCE} is adapted from Eq. 5.2 where we replace $\hat{\mathbf{w}} = D(E(\tilde{\mathbf{x}}, \mathbf{w}))$ with $\hat{\mathbf{w}} = D(G(\mathbf{z}))$. η is set to 1.0 as a hyper-parameter to balance the two objective terms.

We also compare the deep-learning-based steganography technique used in our solution ([223]) to two well-established, non-deep learning steganographic methods [224, 225] that alter the frequency coefficients of JPEG compression.

Results. We report the fingerprint detection performance in Table 5.1 fourth and fifth columns. We observe:

(1) The "Data" row shows the detection accuracy on real testing images for sanity checks: it reaches the 100% saturated accuracy, indicating the effectiveness of the steganography technique by its nature.

(2) Our artificial fingerprints can be almost perfectly and confidently detected from generated images over a variety of applications, generative models, and datasets. The accuracy is ≥ 0.98 except for ProGAN on LSUN *Bedroom*, but its 0.93 accuracy and 10^{-19} p-value are far sufficient to verify the presence of fingerprints. Our hypothesis on the *transferability* from training data to generative models (i.e. generated data) is therefore justified. As a result, artificial fingerprints are qualified for deepfake detection



(a) (b) (c) (d) (e)

Figure 5.2: CelebA samples at 128×128 for Table 5.1 last two columns. (a) Original real training samples. (b) Fingerprinted real training samples. (c) The difference between (a) and (b), 10× magnified for easier visualization. (d) Samples from the non-fingerprinted ProGAN. (e) Samples from the fingerprinted ProGAN.

and attribution.

(3) The *universality* of fingerprint transferability over varying tasks and models validates our solution is agnostic to generative model techniques.

(4) The baseline of joint fingerprinting and generation training (first row) is also moderately effective in terms of fingerprint detection, but we show in Section 5.5.3 it leads to strong deterioration of generation quality.

(5) Conventional steganography methods [224, 225] (second and third rows) do not transfer hidden information into models, indicated by the random guess performance during decoding. We attribute this to the discrepancy between deep generation techniques and shallow steganography techniques. We reason that generative models leverage deep discriminators to approximate common image patterns including low-level fingerprints. Only comparably deep-learning-based fingerprinting techniques, e.g. [223], are compatible to hide and transfer fingerprints to the models, while hand-crafted image processing is not effective. Therefore, the transferability of our fingerprinting is non-trivial.

Dataset	Fgpt tech	Model	Bit acc ⇑	p-value	Orig FID	Fgpt FID ⇓
CelebA	Eq. 5.6	ProGAN	0.93	$< 10^{-19}$	14.09	60.28
	[224]	StyleGAN2	0.51	0.46	6.41	6.93
	[225]	StyleGAN2	0.53	0.31	6.41	6.82
	[223]	Data	1.00	-	-	1.15
	[223]	ProGAN	0.98	$< 10^{-26}$	14.09	14.38
	[223]	StyleGAN	0.99	$< 10^{-28}$	8.98	9.72
	[223]	StyleGAN2	0.99	$< 10^{-28}$	6.41	6.23
LSUN *Bedroom*	[223]	ProGAN	0.93	$< 10^{-19}$	29.16	32.58
	[223]	StyleGAN	0.98	$< 10^{-26}$	24.95	25.71
	[223]	StyleGAN2	0.99	$< 10^{-28}$	13.92	14.71
LSUN *Cat*	[223]	ProGAN	0.98	$< 10^{-26}$	45.22	48.97
	[223]	StyleGAN	0.99	$< 10^{-28}$	33.45	34.01
	[223]	StyleGAN2	0.99	$< 10^{-28}$	31.01	32.60
CIFAR-10	[223]	BigGAN	0.99	$< 10^{-28}$	6.25	6.80
Horse→Zebra	[223]	CUT	0.99	$< 10^{-28}$	22.98	23.43
Cat→Dog	[223]	CUT	0.99	$< 10^{-28}$	55.78	56.09

Table 5.1: Artificial fingerprint detection in bitwise accuracy (⇑ indicates higher is better) and generation quality in FID (⇓ indicates lower is better). The "Data" row corresponds to real testing images for a sanity check. The "Orig FID" column corresponds to the original (non-fingerprinted) models for references. The first three rows are the baselines.

5.5.3 Fidelity

The fidelity of generated images is as critical as the transferability. Fingerprinting should have a negligible side effect on the functionality of generative models. This preserves the original generation quality and avoids the adversary's suspect of the presence of fingerprints. The steganography technique we used should enable this, which we validate empirically.

Evaluation. We use Fréchet Inception Distance (FID) [118] to evaluate the generation quality; the lower, the more realistic. We measure FID between a set of 50k generated

images and a set of 50k real non-fingerprinted images, in order to evaluate the quality of the generated set. When calculating different FIDs for each dataset, the real set is unchanged.

Results. We compare the generation quality of original and fingerprinted generative models in Table 5.1 sixth and seventh columns. We observe:

(1) The "Data" rows are for sanity checks: embedding fingerprints into real images does not substantially deteriorate image quality: FID ≤ 1.15 is in an excellent realism range. This validates the secrecy of the steganographic technique and lays a valid foundation for high-quality model training.

(2) For a variety of settings, the performance of the fingerprinted generative models tightly sticks to the original limits of their non-fingerprinted baselines. The heaviest deterioration is as small as $+3.75$ FID happening for ProGAN on LSUN *Cat*. In practice, the generated fingerprints are imperceptibly hidden in the generated images and can only be perceived under $10\times$ magnification. See Figure 5.2 for demonstrations. Therefore, the fidelity of fingerprinted models is justified and it qualifies our solution for deepfake detection and attribution.

(3) The baseline of joint fingerprinting and generation training (first row) deteriorates generation quality remarkably. This indicates model fingerprinting is a non-trivial task: direct fingerprint reconstruction distracts adversarial training. In contrast, our solution leverages image steganography and fingerprint transferability, sidesteps this issue, and leads to better performance.

5.5.4 Robustness

Deepfake media and generative models may undergo post-processing or perturbations during broadcasts. We validate the robustness of our fingerprint detection given a variety of image and model perturbations, and investigate the corresponding working ranges.

Perturbations. We evaluate the robustness against four types of *image perturbation*: additive Gaussian noise, blurring with Gaussian kernel, JPEG compression, center cropping. We also evaluate the robustness against two types of *model perturbations*: model weight quantization and adding Gaussian noise to model weights. For quantization, we compress each model weight given a decimal precision. We vary the amount of perturbations, apply each to the generated images or to the model directly, and detect the fingerprint using the pre-trained decoder.

Results. We evaluate the artificial fingerprint detection over 50k images from a fingerprinted ProGAN. We plot the bitwise accuracy w.r.t. the amount of perturbations in Figure 5.3. We observe:

(1) For all the image perturbations, fingerprint detection accuracy drops monotonously as we increase the amount of perturbation, while for small perturbations accuracy drops rather slowly. We consider accepting accuracy $\geq 75\%$ as a threshold (p-value $= 2.8 \times 10^{-7}$). This results in the working range w.r.t. each perturbation: Gaussian noise standard deviation $\sim [0.0, 0.05]$, Gaussian blur kernel size $\sim [0, 5]$, JPEG compression quality $\sim [50, 100]$, center cropping size $\sim [86, 128]$, quantization decimal precision $\leq 10^{-1}$, and model noise standard deviation $\sim [0.0, 0.18]$, which are reasonably wide ranges in practice.

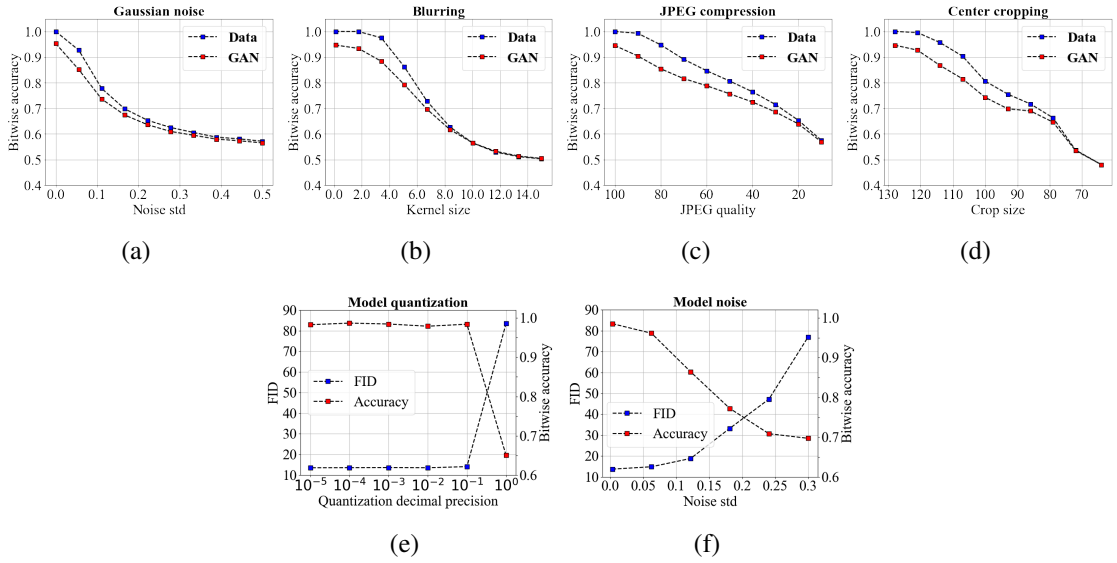


Figure 5.3: Red plots show the artificial fingerprint detection in bitwise accuracy w.r.t. the amount of perturbations over ProGAN trained on CelebA. In the left four plots (robustness against *image perturbations*), blue dots represent detection accuracy on the fingerprinted real training images, which serve as the upper bound references for the red dots. In the right two plots (robustness against *model perturbations*), blue dots represent the FID of generated images from the perturbed models.

(2) For image perturbations (the left four subplots) outside the above working ranges, the reference upper bounds drop even faster and the margins to the testing curves shrink quickly, indicating that the detection deterioration is irrelevant to model training but rather relevant to the heavy quality deterioration of training images.

(3) For model perturbations (the right two subplots) outside the above working ranges, image quality deteriorates faster than fingerprint accuracy: even before the accuracy gets lower than 75%, FID has already increased by $> 500\%$.

(4) As a result of (2) and (3), before fingerprint detection degenerates close to random guess ($\sim 50\%$ accuracy), image quality has been heavily deteriorated by strong perturbations (Figure 5.4), which indicates that our fingerprints are more robust than image functionality itself in the case of these studied perturbations.

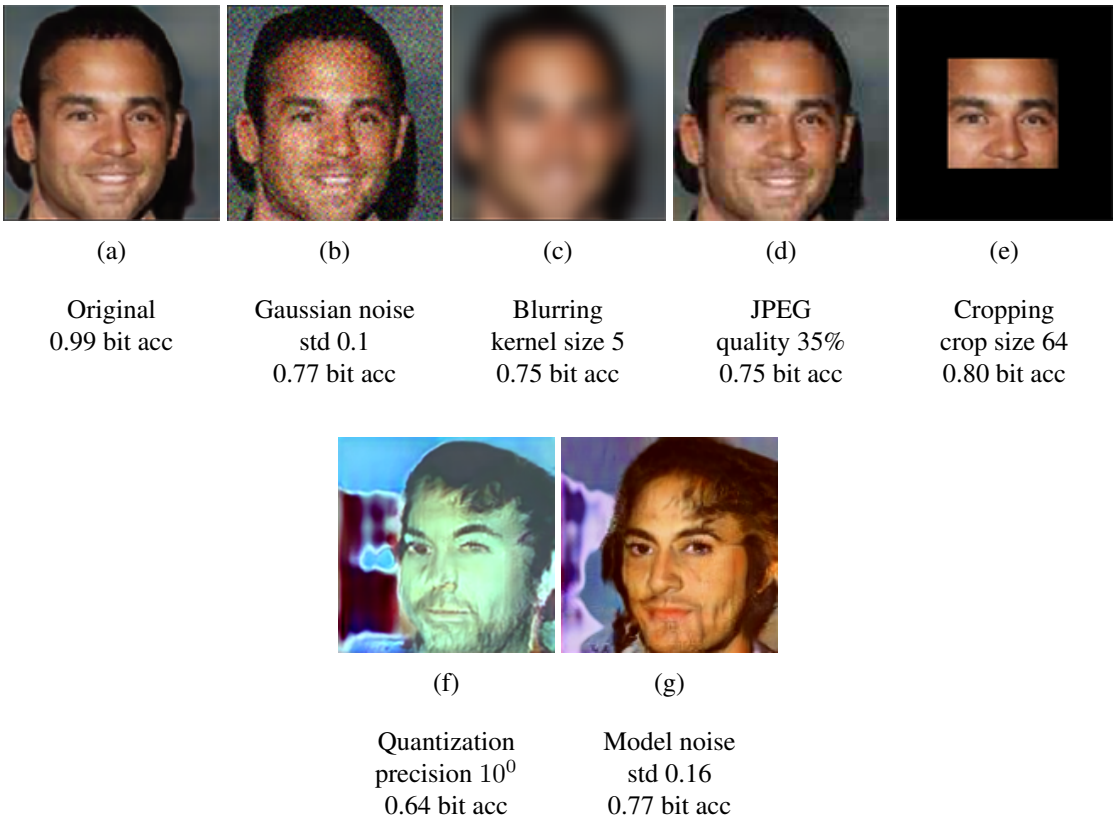


Figure 5.4: Perturbed image samples from the fingerprinted ProGAN and the corresponding fingerprint detection accuracy. The detection still performs robustly (bitwise accuracy ≥ 0.75) even when the image quality heavily deteriorates.

Discussion on attacks. Other attacks that require training counter models might be conceivable. For example, to train a model that removes the fingerprints from generated images (e.g. a denoising autoencoder). However, this would require attackers to have paired training images before and after the fingerprint embedding. In our scenario, we assume that the fingerprint encoder is not released which hinders this training data collection requirement.

5.5.5 Secrecy

The presence of a fingerprint embedded in a generative model should not be easily detected by a third party, otherwise, it would be potentially manipulated.

Attack. In order to quantify the secrecy, we formulate it as a binary classification problem: the presence of a fingerprint in an image. We follow the attack protocol in [235] to perform the Artificial Training Sets (ATS) attack [246]. We aim to classify between fingerprinted and non-fingerprinted test images, but we have no supervision. The intuition is to train another steganography encoder to expand the test set and establish an artificial setting with known labels which enables supervised training. The original test class space is now a subspace of the artificial training class space and is separable by the training task.

The attack is as follows: We independently train another steganography encoder. We consider the original test images as negative training samples. Then, we apply the encoder twice to the test set to obtain extra images fingerprinted 2 times (corresponding to originally non-fingerprinted images) or 3 times (corresponding to originally fingerprinted images), which are regarded as positive training samples. Then we train an SVM classifier [247] using such positive and negative samples, in order to separate between images fingerprinted 0-1 time (original set), and the ones fingerprinted 2-3 times (artificial training set).

During testing, we first apply the encoder once to the test images so that the originally non-fingerprinted images are now fingerprinted 1 time (belonging to 0-1 class), and the originally fingerprinted images are now fingerprinted 2 times (belonging to 2-3 class). Then we can use the classifier to separate them and propagate the predictions back to the original images. We evaluate the attack on a set of 250 fingerprinted deepfake images and

250 non-fingerprinted deepfake images.

Results. The binary classification accuracy on the existence of fingerprint is $\mathbf{0.502}$ according to the ATS attack, which is close to random guess. It indicates our fingerprinting is secret enough from being detected by adversaries who have no access to our encoder and decoder. We reason that the steganography encoder trained from different initialization uses different patterns to hide the fingerprint, and therefore does not couple well with the victim encoder. This also supports our previous discussion of the importance of keeping the encoder private to support both the secrecy and the robustness of the artificial fingerprints.

5.5.6 Deepfake Detection

In the previous sections, we showed that our fingerprinting solution is effective in transferring the fingerprints and meeting the other required criteria. We now discuss how to use it for deepfake detection and attribution.

Unlike existing methods that detect intrinsic differences between the real and deepfake classes [1, 2, 218, 220], we, *standing for model inventors*, propose a proactive solution by embedding artificial fingerprints into generative models and consequently into the generated images. In practice, responsible model inventors, different from malicious deepfake users, should be *eager/willing* to do so. Then we convert the problem to verifying if one decoded fingerprint is in our fingerprint regulation database or not. Even with a non-perfect detection accuracy, we can still use our solution based on the null hypothesis test in Section 5.4. We consider deepfake verification given $\geq 75\%$ (p-value $= 2.8 \times 10^{-7}$) bit

matching. This is feasible based on two assumptions: (1) The decoded fingerprint of a real image is random; and (2) the fingerprint capacity is large enough such that the random fingerprint from a real image is unlikely to collide with a regulated fingerprint in the database. The second condition is trivial to satisfy, considering we sample fingerprints $\mathbf{w} \in \{0,1\}^n$ and $n = 100$. 2^{100} is a large enough capacity. Then we validate the first assumption by the deepfake detection experiments below.

Baselines. We compare to two recent state-of-the-art CNN-based deepfake detectors [1, 2] as baselines. [2] is trained on 40k real images and 40k generated images equally from four generative models with distinct fingerprints. We consider the *open-world* scenario where disjoint generative models are used in training and testing, to challenge the classifier's generalization. For [1] we use the officially released model because they already claim improved generalization across different generation techniques.

Results. We compare our solution to the two baselines on a variety of generation applications, models, and datasets. We test on 4k real images and 4k generated images equally from four generative models with distinct fingerprints. We report deepfake detection accuracy in Table 6.3 fourth column. We observe:

(1) Our solution performs perfectly (100% accuracy) for all the cases, turning open-world deepfake detection into a trivial fingerprinting detection and matching problem.

(2) [2] deteriorates to random guess ($\sim 50\%$ accuracy) because of the curse of domain gap between training and testing models. In contrast, our solution benefits from being agnostic to generative models. It depends only on the presence of fingerprints rather than the discriminative cues that are overfitted during training.

(3) Our solution outperforms [1] with clear margins. In particular, [1] degenerates

Dataset	Model	Detector	Detection acc ⇑	Attribution acc ⇑
CelebA	ProGAN	[2]	0.508	0.235
		[1]	0.924	N/A
		Ours	1.000	1.000
	StyleGAN	[2]	0.497	0.168
		[1]	0.906	N/A
		Ours	1.000	1.000
	StyleGAN2	[2]	0.500	0.267
		[1]	0.895	N/A
		Ours	1.000	1.000
LSUN *Bedroom*	ProGAN	[2]	0.493	0.597
		[1]	0.952	N/A
		Ours	1.000	1.000
	StyleGAN	[2]	0.499	0.366
		[1]	0.956	N/A
		Ours	1.000	1.000
	StyleGAN2	[2]	0.491	0.267
		[1]	0.930	N/A
		Ours	1.000	1.000
LSUN *Cat*	ProGAN	[1]	0.951	N/A
		Ours	1.000	1.000
	StyleGAN	[1]	0.923	N/A
		Ours	1.000	1.000
	StyleGAN2	[1]	0.905	N/A
		Ours	1.000	1.000
CIFAR-10	BigGAN	[1]	0.815	N/A
		Ours	1.000	1.000
Horse→Zebra	CUT	[1]	0.836	N/A
		Ours	1.000	1.000
Cat→Dog	CUT	[1]	0.902	N/A
		Ours	1.000	1.000

Table 5.2: Deepfake detection and attribution accuracy (⇑ indicates higher is better). [1] is not applicable to the multi-source attribution scenarios in the last column.

when model techniques evolve to be more powerful (from ProGAN to StyleGAN2), or condition on some input guidance. On the contrary, our proactive solution synergizes

with this evolution with high fingerprint detection accuracy, and therefore, with perfect deepfake detection accuracy.

(4) In general, although [1] generalizes better than [2], it is still subject to future adversarial evolution of generative models, which were witnessed rapidly progressing over the last few years. For example, [1] was effectively evaded in [221] by extremely small perturbations. In contrast, our work offers higher sustainability in the long run by proactively enforcing a margin between real and generated images. This requires and enables responsible model inventors' disclosure against potential misuses of their models.

5.5.7 Deepfake Attribution

The goal of attribution is to trace the model source that generated a deepfake. It upgrades the binary classification in detection to multi-class classification. Our artificial fingerprint solution can be easily extended for attribution and enable us, standing for model inventors, to attribute responsibility to our users when misuses occur.

Baseline. [1] is not applicable to multi-source attribution. We only compare to [2] in the *open-world* scenario, i.e., the training and testing sets of generative models are not fully overlapping. Given 40k generated images equally from four generative models with distinct fingerprints, we use [2] to train four one-vs-all-the-others binary classifiers. During testing, all four classifiers are applied to an image. We assign the image to the class with the highest confidence if not all the classifiers reject that image. Otherwise, it is assigned to the unknown label.

Results. We compare our solution to [2] on CelebA and LSUN *Bedroom*. We test

on 4k/4k generated images equally from four model sources that are in/out of the training set of [2]. We report deepfake attribution accuracy in Table 6.3 last column. We obtain the same discoveries and conclusions as those of deepfake detection in Section 5.5.6. The open-world attribution deteriorates for the CNN classifier [2] while our fingerprinting solution maintains the perfect (100%) accuracy.

5.6 Conclusion

Detecting deepfakes is a complex problem due to the rapid development of generative models and the possible adversarial countermeasure techniques. For the sake of sustainability, we investigate a proactive solution on the model inventors' side to make deepfake detection agnostic to generative models. We root deepfake detection into training data, and demonstrate the transferability of artificial fingerprints from training data to a variety of generative models. Our empirical study shows several beneficial properties of fingerprints, including universality, fidelity, robustness, and secrecy. Experiments demonstrate our perfect detection and attribution accuracy that outperforms two recent state of the art. As there have been recent concerns about the release of powerful generative techniques, our solution closes the responsibility loop between publishing pre-trained generative model inventions and their possible misuses. It opens up possibilities for inventors' responsibility disclosure by allocating each model a unique fingerprint.

5.7 Acknowledgement

Ning Yu is partially supported by Twitch Research Fellowship. We thank David Jacobs, Matthias Zwicker, Abhinav Shrivastava, Yaser Yacoobfor, and Apratim Bhattacharyya for constructive discussion and advice.

Chapter 6: Responsible Disclosure of Generative Models Using Scalable Fingerprinting

6.1 Introduction

Over the recent five years, deep generative models have demonstrated stunning performance in generating photorealistic images [9, 64, 248, 249], and have delivered extensive applications ranging from low-level image postprocessing [37, 38, 39, 40, 41, 42, 43] to high-level semantic-conditioned image generation [29, 33, 34, 86, 250, 251] and attribute editing [44, 45, 46, 47]. These successes are considerably boosted by the revolutionary technique of generative adversarial networks (GANs) [9, 10, 12, 13, 14, 15, 16, 17], and are closing the gap of appearances between real images and fake ones.

Despite plenty of use cases of generative models, a flood of strong concerns arise [252, 253, 254]: how can these models be misused to spoof sensors, generate deep fakes, and enable misinformation at scale? Not only human beings have difficulties in distinguishing deep fakes, but dedicated research efforts on deep fake detection [216, 217, 218, 219] and attribution [1, 2, 189] are also unable to sustain longer against the evolution of generative models. For example, researchers delve into details on how deep fake detection works, and learn to improve generation that better fits the detection criteria [216, 220]. In

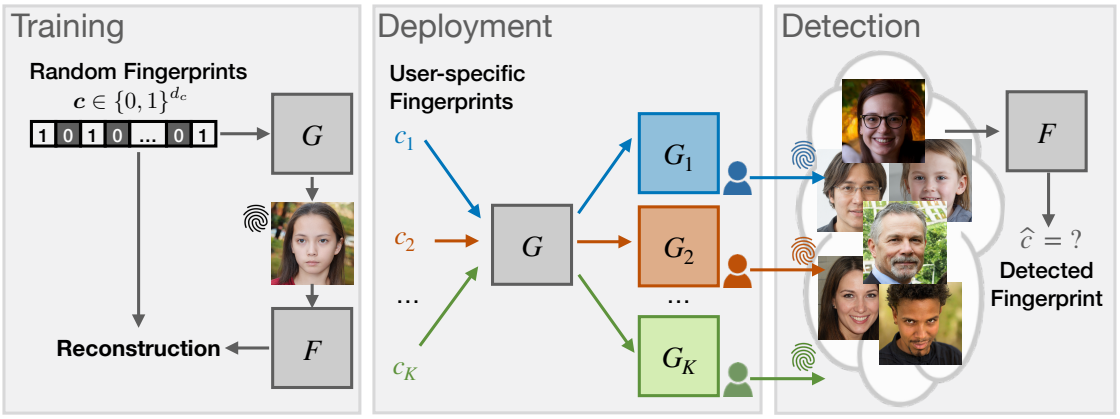


Figure 6.1: The diagram of our fingerprinting mechanism for generative models. Left: A responsible model owner trains fingerprinting networks in the image generation context. Middle: During deployment, the model owner can ad-hoc generate a large number of fingerprinted generator instances, each corresponding to a user download. Right: The model owner can detect fingerprints from generated images to verify and trace a user's deep fake misuse. This enables the owner's responsible disclosure.

principle, any successful detector can play an auxiliary role in augmenting the discriminator in the next iteration of GAN techniques, and consequently results in an even stronger generator.

The dark side of deep generative models makes its industrialization process not as straightforward as those of other artificial intelligence techniques. For example, when commercializing the GPT-2 [243] and GPT-3 [244] models, OpenAI hesitates to open-source the models but rather only release the black-box APIs[1]. They involve expensive human labor in the loop to monitor and prevent the malicious use of the APIs. Yet still, it is a challenging and industry-wide task on how to trace the responsibility of the downstream use cases in an open end.

To pioneer in this task, we propose a fingerprinting mechanism to enable responsible disclosure of deep generative models, that allows responsible researchers and companies

[1] https://openai.com/blog/openai-api/

to fingerprint their models. As a result, the generated samples contain fingerprints that can be accurately detected and attributed to their sources. This is achieved by an efficient and scalable ad-hoc generation of a large population of models with distinct fingerprints. See Figure 7.1 Middle.

Similar in the spirit of the dynamic filter networks [79] and style-based generator architectures [16, 17] where their network filters are not freely learned but conditioned on an input, we regulate to parameterize a unique fingerprint into the filters of each generator instance. The core gist is to incorporate a fingerprint auto-encoder into a GAN framework while preserving the original generation performance. See Figure 7.1 Left. In particular, given a GAN backbone with a generator and a discriminator, we use the fingerprint embedding from the encoder to modulate each convolutional filter of the generator (Figure 6.2(b)), and try to decode this fingerprint from the generated images. We jointly train the fingerprint auto-encoder and GAN with our fingerprint related losses and the original adversarial loss. See Figure 6.2(a) for the diagram, and 6.3.2 for details.

After training, the responsible model owner is capable of fingerprinting and releasing different generator instances to different user downloads, which are equipped with the same generation performance but with different fingerprints. Each user download corresponds to a unique fingerprint, which is maintained by the owner's database. As a result, when misuse of a model happens, the model owner can use the decoder to detect the fingerprint from the generated images, verify it to the database, and then trace the responsibility of the user. See Figure 7.1 Right. Based on this form of responsible disclosure, responsible model owners, like OpenAI, have a way to mitigate adverse side effects on society when releasing their powerful models, while at the same time should have an automatic way to

attribute misuse.

There are several key properties of our mechanism. The **efficiency** to instantiate a generator is inherently satisfied because, after training, the fingerprint encoding and filter modulation run with little overhead. We evaluate the **effectiveness** of our fingerprinting and obtain almost perfect detection accuracy. We also justify the **fidelity** with a negligible side effect on the original generation quality. See Section 6.4.2. Our recommended operation point uses a 128-bit fingerprint (Section 6.4.3) which in principle results in more than 10^{36} identifiable generator instances. The **scalability** benefits from the fact that fingerprints are randomly sampled on the fly during training so that fingerprint detection generalizes well for the entire fingerprint space. See Section 6.4.4 for validation. In addition, we validate in Section 6.4.5 the **robustness** and **immunizability** of our fingerprinting against perturbation on generated images.

To target the initial motivation, we test our mechanism, as a proactive method, in the deep fake detection and attribution tasks. We show in Section 6.4.6 saturated performance and advantages over two state-of-the-art discriminative methods [1, 2] especially in the open world. This is because, conditioned on user-specific fingerprint inputs, the presence of such fingerprints in generated images guarantees the margin between real and fake, and facilitates the attribution and responsibility tracing of deep fakes to their sources.

Our **contributions** are in four thrusts:

(1) We introduce the concept of fingerprinting for generative models that enables a responsible disclosure of state-of-the-art GAN models.

(2) We present a novel mechanism for efficient and scalable fingerprinting GAN models, i.e., only one generic GAN model is trained while more than 10^{36} identifiable

generator instances (each with a unique fingerprint) can be obtained with little overhead during deployment.

(3) We also justify several key properties of our fingerprinting, including effectiveness, fidelity, large capacity, scalability, robustness, and immunizability.

(4) Finally, for the deep fake detection and attribution tasks, we validate its saturated performance and advantages over previous learning-based discriminative methods. It makes our responsible disclosure independent of the evolution of GAN techniques.

6.2 Related work

Generative adversarial networks (GANs). GANs [9] were invented to model real data distribution via solving a real/fake binary classification problem, and demonstrated stunning realism. This has triggered rapid progresses towards generating photorealistic images in high resolution [10, 12, 13, 14, 15, 16, 17]. Many successful generative applications are established on them, e.g., image postprocessing [37, 38, 39, 40, 41, 42, 43], image translation [22, 29, 30, 31, 32, 33, 34, 35, 36], and image manipulation [44, 45, 46, 47]. They deliver generative modeling techniques to ordinary people and make deep fakes popular on social media, which urges mitigation against deep fake misuse. As a response, our fingerprinting mechanism enables responsible disclosure of generative models, and is agnostic to their evolution and categories. We demonstrate that our method can effectively fingerprint the state-of-the-art GAN models [17] without affecting generation quality.

Deep fake detection and attribution. These tasks come along with the increasing concerns on deep fake misuse [252, 253, 254]. Deep fake detection is a binary classification

problem to distinguish fake samples from real ones, while attribution further traces their sources. The findings of visually imperceptible but machine-distinguishable patterns in GAN-generated images make these tasks viable by noise pattern matching [189], deep classifiers [2, 196, 255], or deep Recurrent Neural Networks [256]. [1] follows up with a generalization of classification across different GAN techniques. [217, 218, 220, 241] observe that mismatches between real and fake in frequency domain or in texture representation can facilitate deep fake detection.

However, these *passive* detection methods heavily rely on the inherent clues in deep fakes. Therefore, they can barely sustain a long time against the adversarial iterations of GAN techniques. For example, [220] improves generation realism by closing the gap in generated high-frequency components. To handle this situation, artificial fingerprinting is proposed in [3] to *proactively* embed clues into generative models by rooting fingerprints into training data. This makes deep fake detection independent of GAN evolution. Yet, as *indirect* fingerprinting, [3] cannot scale up to a large number of fingerprints because they have to pre-process training data for each individual fingerprint and re-train a generator with each fingerprint. Our method is similar in spirit to [3], but possesses fundamental advantages by *directly* fingerprinting generative models: after training one generic fingerprinting model, we can instantiate a large number of generators ad-hoc with different fingerprints.

Image steganography and watermarking. Image steganography and watermarking represent a technique of hiding information into carrier images [227]. Previous techniques rely on Fourier transform [228, 229], JPEG compression [224, 225], or least significant bits modification [230, 231, 232]. Recent works substitute hand-crafted hiding procedures with neural network embedding [233, 234, 236] and/or generative modeling [222, 223,

235, 237, 257, 258, 259]. Our fingerprinting is conceptually and functionally orthogonal to steganography and watermarking. Instead of retouching pixels and encoding information into individual images, our solution is the first study to directly modify generator parameters and encode information into the model such that all the generated images contain the identical hidden information. Technically, compared to the pipeline of a generator followed by a watermarking module, our solution introduces zero generation overheads, and makes adversarial model surgery impossible that targets to detach watermarking from image generation.

Network watermarking. Network watermarking techniques [184, 185, 238, 239, 240] embed watermarks into network parameters rather than pixels while not deteriorating the original utility. Our solution shares motivations with the existing works but is substantially different in terms of concepts, motivations, and techniques. For concepts, the existing works are applicable to only image classification models, while a solution for generative models is missing altogether and therefore urgently needed. For motivations, the existing works target to fingerprint a single model, while we are motivated by the limitation of [3] to scale up the fingerprinting to as many as 10^{36} various generator instances within one-time training. For techniques, the existing works embed fingerprints in the input-output behavior of the classification model [184, 238], while our solution does not require such trigger input because GANs do not specify its input.

Conditioned parameters in networks. We achieve generator fingerprinting by modulating convolutional filters with fingerprint embeddings. There exist a variety of works about conditioning network parameters on the input, some using adaptive normalization [16, 22, 32, 34, 131], some using dynamic filters [17, 79, 86], and others using self/reference

attention [4, 107, 109, 110, 260]. However, none of them pays attention to mitigate deep fake misuse. We pioneer in this novel direction.

6.3 GAN fingerprinting networks

We present responsible disclosure of generative models as a novel solution to mitigate deep fake misuse, which is agnostic to GAN techniques and sustainable along with their evolution. In Section 6.3.1 we depict how our fingerprinting mechanism enables responsible disclosure. We introduce in details our loss design in Section 6.3.2 and modulation design in Section 6.3.2.

6.3.1 Problem statement

Generative models can be misused to spoof sensors, generate deep fakes, and enable misinformation at scale [252, 253, 254]. Responsible researchers and companies like OpenAI lean conservative to the side effects of their powerful generative models on society. As a result, they have to provide only the APIs rather than the source code of their models, and limit the access to them by approving users only after rigorous reviews. This notably slows down the industrialization process for the deployment of generative models.

One promising solution to mitigate responsible model owners' concerns and enhance their regulation is to enable responsible disclosure of their models and trace the users who misuse their models. We achieve this by equipping owners with a model fingerprinting auto-encoder, such that they can use the encoder to efficiently instantiate a large population

of models with the same generation performance but with distinct fingerprints. As a result, the model owner can release different model instances to different users, where each user's download corresponds to a unique fingerprint maintained by the owner's database. When misuse of a model happens, the model owner can use the decoder to detect the fingerprint from the generated images, verify it to the database, and then trace the responsibility of the user. Based on this form of responsible disclosure, responsible model owners, like OpenAI, should feel safer and be stressed less by society when releasing their powerful models, while at the same time having an automatic way to attribute misuses.

Throughout the paper, we stand for the model owner's perspective. The owner is regarded as the management hub of our experiments. The generic model training, model fingerprinting, model instance allocation to users, and deepfake detection/attribution are all conducted on the owner's side. None of the encoder, decoder, and training data are accessible to the public.

6.3.2 Loss design

We list symbol notations at the beginning. We use latent code $\sim \mathcal{N}(\mathbf{0}, \mathbf{I}^{d_z})$ to control generated contents. We set $d_z = 512$. We represent fingerprint $\sim \text{Ber}(0.5)^{d_c}$ as a sequence of bits. It follows Bernoulli distribution with probability 0.5. We non-trivially choose the fingerprint length d_c in Section 6.4.3. We denote encoder E mapping to its embedding, generator G mapping $(, E())$ to the image domain, discriminator D mapping an image $\sim p_{\text{data}}$ to the real/fake classification probability, and decoder F mapping an

image to the decoded latent code and fingerprint $\overset{\frown}{(,)}$. In the following formulations, we denote $G(, E())$ as $G(,)$ for brevity.

We consider three goals in our training. First, we preserve the original functionality of GANs to generate realistic images, as close to real distribution as possible. We use the unsaturated logistic loss as in [9, 16, 17] for real/fake binary classification:

$$_{adv} = \underset{\sim p_{\text{data}}}{\mathbb{E}} \log D() + \underset{\substack{\sim \mathcal{N}(\mathbf{0}, \mathbf{I}^{d_z}) \\ \sim \text{Ber}(0.5)^{d_c}}}{\mathbb{E}} \log(1 - D(G(,)))$$

(6.1)

In addition, similar to [63], we reconstruct the latent code through the decoder F to augment generation diversity and mitigate the mode collapse issue of GANs [28, 63, 64].

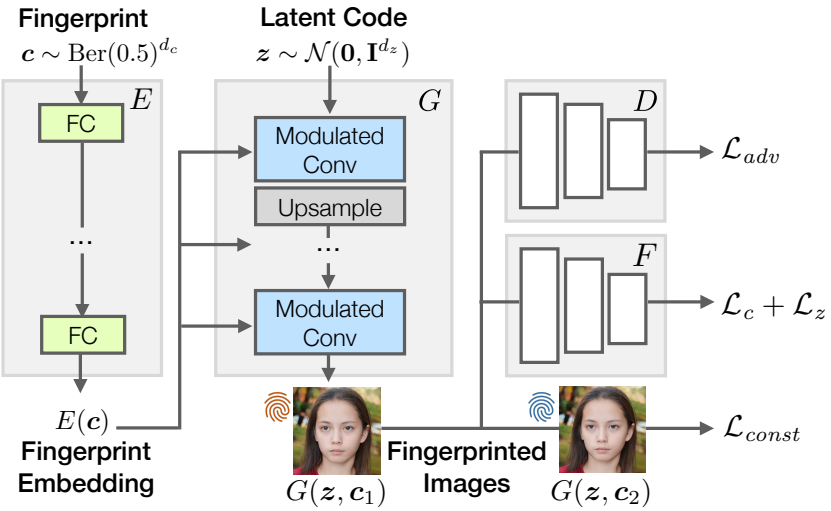
$$_{z} = \underset{\substack{\sim \mathcal{N}(\mathbf{0}, \mathbf{I}^{d_z}) \\ \sim \text{Ber}(0.5)^{d_c}}}{\mathbb{E}} \sum_{k=1}^{d_z} \left(z_k - F\left(G(,)\right)_k \right)^2 \tag{6.2}$$

where we use the first d_z output elements of F that correspond to the decoded latent code.

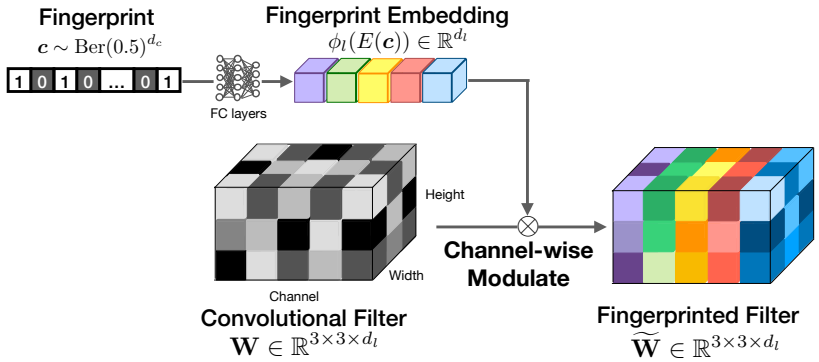
The second goal is to reconstruct the fingerprint so as to achieve our core functionality of fingerprint detection.

$$\begin{aligned} _{c} = \underset{\substack{\sim \mathcal{N}(\mathbf{0}, \mathbf{I}^{d_z}) \\ \sim \text{Ber}(0.5)^{d_c}}}{\mathbb{E}} & \sum_{k=1}^{d_c} c_k \log \sigma \left(F\left(G\left(,\right)\right)_{d_z+k} \right) \\ & + (1 - c_k) \log \left(1 - \sigma \left(F\left(G\left(,\right)\right)_{d_z+k} \right) \right) \end{aligned} \tag{6.3}$$

where we use the last d_c output elements of F as the decoded fingerprint. $\sigma(\cdot)$ denotes the sigmoid function that differentiably clips the output to the range of $[0, 1]$. The reconstruction



(a) Pipeline.



(b) Modulated convolutional layer.

Figure 6.2: The diagram of our fingerprinting pipeline and the zoom-in of the modulated convolutional layer.

is therefore a combination of binary classification for each bit.

It is worth noting that we use one decoder to decode both the latent code and fingerprint, which benefits for cooperating between them so as to explicitly disentangle their representations as discussed below.

The third goal is to disentangle the representation between latent code and fingerprint. Desirably, latent code should have exclusive control over the generated content. This sticks to the original generation functionality. Therefore, two images with different fingerprints but with identical latent code should have a consistent appearance. We formulate the consistency loss as:

$$\mathcal{L}_{const} = \mathop{\mathbb{E}}_{\substack{\sim\mathcal{N}(\mathbf{0},\mathbf{I}^{d_z}) \\ \mathbf{c}_1,\mathbf{c}_2\sim\text{Ber}(0.5)^{d_c}}} \|G(\mathbf{z},\mathbf{c}_1) - G(\mathbf{z},\mathbf{c}_2)\|_2^2 \tag{6.4}$$

The disentangled effect is demonstrated in Figure 6.3.

Our final training objective is as follows. We optimize it under the adversarial training framework w.r.t. E, G, F, and D.

$$\min_{E,F,G} \max_{D} \lambda_{1}\mathcal{L}_{adv} + \lambda_{2}\mathcal{L}_{z} + \lambda_{3}\mathcal{L}_{c} + \lambda_{4}\mathcal{L}_{const} \tag{6.5}$$

where $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 2.0$, and $\lambda_4 = 2.0$ are hyper-parameters to balance the magnitude of each loss term. See Figure 6.2(a) for the diagram.

6.3.3 Fingerprint modulation

At the architectural level, it is non-trivial how to embed $E()$ into G. The gist is to embed fingerprint into the generator parameters rather than generator input, so that (1) our mechanism is agnostic to the architecture of a GAN model, and (2) after training a generic model we can instantiate a large population of generators ad-hoc with different fingerprints. The second point is a critical advantage to make our fingerprinting efficient and scalable, as validated in Section 6.4.4. We then deploy only the fingerprinted generator instances, not including the encoder.

We achieve this by modulating convolutional filters in the generator backbone with our fingerprint embedding, similar in spirit of [17]. Given a convolutional kernel $\mathbf{W} \in \mathbb{R}^{3\times3\times d_l}$ at layer l, we first project the fingerprint embedding $E()$ through an affine transformation ϕ_l such that $\phi_l(E()) \in \mathbb{R}^{d_l}$. The transformation is implemented as a fully-connect neural layer with learnable parameters. We then scale each channel of $\mathbf{W}$ with the corresponding value in ϕ_l. In specific,

$$\widetilde{\mathbf{W}}_{i,j,k} = \phi_l(E())_k \cdot \mathbf{W}_{i,j,k}, \quad \forall i,j,k \tag{6.6}$$

See Figure 6.2(b) for a diagram illustration. We compare to the other fingerprint embedding architectures in Section 6.4.2 and validate the advantages of this one. We conduct modulation for all the convolutional filters at layer l with the same fingerprint embedding. And we investigate in Section 6.4.7 at which layer to modulate we can achieve the optimal performance. A desirable trade-off is to modulate all convolutional layers.

Note that, during training, latent code and fingerprint are jointly sampled. Yet for deployment, the model owner first samples a fingerprint $_0$, then modulates the generator G with $_0$, and then deploys only the modulated (fingerprinted) generator $G(\cdot,_0)$ to a user. For that user there allows only one input, i.e. the latent code, to the modulated generator. The encoder E, decoder F, and discriminator D are all unavailable to the user. Once a misuse happens, the model owner uses the decoder to decode the fingerprint and attribute it to the user, so as to achieve responsible disclosure.

6.4 Experiments

We describe the experiment settings in Section 7.4.1. We validate several key properties of a fingerprint mechanism from Section 6.4.2 to 6.4.5. We then apply our mechanism to deep fake detection and attribution in Section 6.4.6. We conduct an ablation study on filter modulation in Section 6.4.7.

6.4.1 Setup

Datasets. We conduct experiments on CelebA face dataset [206], LSUN Bedroom and Cat datasets [207]. They are common datasets for image generation, and LSUN Cat is the most challenging one reported in StyleGAN2 [17]. We train/evaluate on 30k/30k CelebA, 30k/30k LSUN Bedroom at the size of 128×128×3, and 50k/50k LSUN Cat at the size of 256×256×3.

GAN backbone. Our fingerprinting mechanism is agnostic to GAN configurations. Without losing representativeness, we follow the line of milestone research [15, 16, 17]

and build upon the most recent state-of-the-art StyleGAN2 [17] config E. This also aligns to the settings in [3] and facilitates our direct comparisons in Section 6.4.2 and 6.4.6. One modification happens to how we input the latent code. Instead of encoding it through filter modulation, we find that directly feeding it through the input of the generator achieves better results. See Section 6.4.2. Our model is trained from scratch with the original training protocol in [17].

6.4.2 Effectiveness and fidelity

Evaluation. The effectiveness indicates that the input fingerprints consistently appear in the generated images and can be accurately detected by the decoder. This is measured by fingerprint detection bitwise accuracy over 30k random samples (with random latent codes and random fingerprint codes). A larger value is more desirable. We use 128 bits to represent a fingerprint. This is a non-trivial setting as analyzed in Section 6.4.3.

The fidelity reflects how imperceptible the original generation performance is affected by fingerprinting. It also helps avoid one's suspect of the presence of fingerprints which may attract adversarial fingerprint removal. Fréchet Inception Distance (FID) [118] is the standard measure for generation quality. We report FID between 30k generated images and 30k real testing images. A smaller value indicates the generated images are more realistic in general.

Baselines. We compare to five baseline methods. The first baseline is the StyleGAN2 [17] backbone with the original architecture: the latent code is encoded through filter modulation.

Method	CelebA		LSUN Bedroom		LSUN Cat	
	Bit acc ⇑	FID ⇓	Bit acc ⇑	FID ⇓	Bit acc ⇑	FID ⇓
StyleGAN2	-	9.37	-	19.24	-	31.01
[3]	0.989	14.13	0.983	21.31	0.990	32.60
Ours	0.991	11.50	0.993	20.50	0.996	33.94
Ours Variant I	0.999	12.98	0.999	20.68	0.500	34.23
Ours Variant II	0.987	13.86	0.927	21.70	0.869	34.33
Ours Variant III	0.990	22.59	0.896	64.91	0.901	51.74

Table 6.1: Fingerprint detection in bitwise accuracy and generation fidelity in FID. ⇑/⇓ indicates a higher/lower value is more desirable.

It provides the upper bound of fidelity while has no fingerprinting functionality.

The second baseline is [3] which is the other proactive but *indirect* fingerprinting method for GANs. Regardless of its performance, our method is substantially more efficient and scalable than theirs in practice, because they have to restart training with a new data collection for each fingerprint. Preferably we can ad-hoc instantiate a large population of fingerprinted generators with little overhead.

We also compare our mechanism to three architectural variants. The motivation of these variants is to incorporate fingerprints in different manners. Variant I: modulating convolutional filters with only latent code embedding, while instead feeding the fingerprint code through the input of the generator. This is to test the necessity of fingerprint modulation. Variant II: modulating filters twice, with latent code embedding and fingerprint code embedding separately. Variant III: modulating filters with the embedding from the concatenation of latent code and fingerprint code.

Results. From Table 6.1, we find that:

(1) On CelebA, all the methods achieve almost perfect fingerprint detection accuracy. This is because CelebA is a landmark-aligned dataset with limited diversity, making

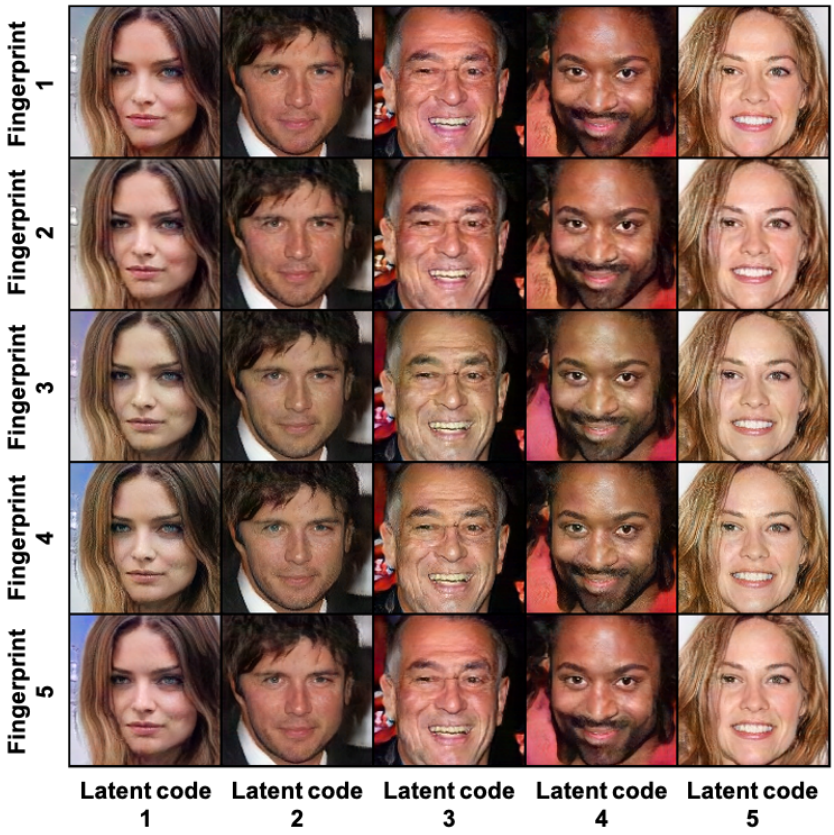


Figure 6.3: Generated samples from five of our generator instances. For each row, we use a unique fingerprint to instantiate a generator. For each column, we feed in the same latent code to the generator instances. We validate the disentangled effect between latent code and fingerprint, which equips each generator instance with identical functionality.

fingerprinting synergize well with aligned pixel generation, regardless of model configuration.

(2) On LSUN Bedroom and Cat, only [3] and our optimal model obtain saturated fingerprint detection accuracy. Ours Variant I, II, and III do not always achieve saturated performance. Especially Ours Variant I fails on LSUN Cat. We reason that filter modulation is a strong formulation for reconstruction. Modulating fingerprints is necessary for their detection while modulating latent code along with fingerprint code distracts fingerprint reconstruction and increases crosstalk among different fingerprints.

(3) Our method has comparable effectiveness and fidelity to [3], plus substantial advantages in practice: We can encode fingerprints with little overhead while they have to re-train for each fingerprint.

(4) Our method results in the optimal fidelity with slightly $\leq$2.93 FID degrading. It is the cost to multi-task with fingerprint detection, but is negligible and worthy. Therefore, our method is a desirable trade-off to achieves effectiveness and fidelity at the same time.

(5) We show in Figure 6.3 uncurated generated samples from a variety of our generator instances. Image qualities are high. Fingerprints are imperceptible. And thanks to the consistency loss $_{const}$ in Eq. 6.4, despite subtle color biases, different generator instances can generate identical images with realistic structures and patterns given the same latent code, while their fingerprints are still distinguishable by our decoder.

6.4.3 Capacity

The capacity indicates the number of unique fingerprints our mechanism can accommodate without crosstalk between two fingerprints. This is determined by d_c, the number of bits

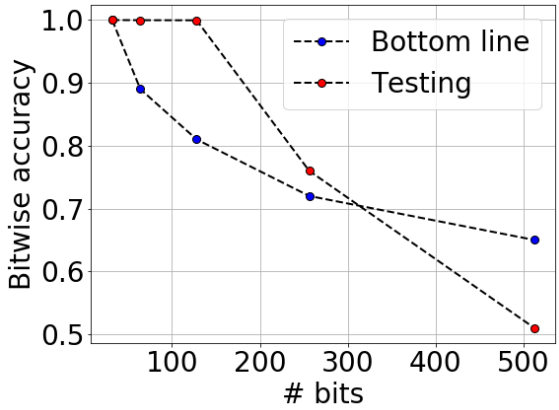


Figure 6.4: Fingerprint detection bitwise accuracy and its bottom line requirement w.r.t. fingerprint bit length on CelebA. The gap is maximized at bit length 128, which therefore becomes our choice.

for fingerprint representation, and by our detection accuracy (according to Section 6.4.2).

The choice of the number of bits is however non-trivial. A larger number can accommodate more unique fingerprints, but can also deteriorate fingerprint detection accuracy. This is because the limited number of fingerprint samples during training is not sufficient to cover the whole space, which may not generalize to testing. A testing fingerprint is very likely not seen or cannot be well represented by training samples.

To figure out the optimal fingerprint bit length, we conduct the following experiments. On one hand, given one length, we evaluate our detection accuracy. On the other hand, we estimate the bottom-line requirement for detection accuracy. This is calculated as the maximal bit overlap percentage among a large bag (1 million) of fingerprint samples. An ideal fingerprint detector should reach an accuracy that is not coarser than this overlap percentage.

In Figure 6.4, we vary the fingerprint bit length in the options of $\{32, 64, 128, 256, 512\}$, and plot the bitwise detection accuracy in red and the bottom line requirement in blue. We

find:

(1) The bottom line requirement to detection accuracy is monotonically decreasing w.r.t. the bit length of fingerprint because, given a finite bag of fingerprint samples, the larger the bit length, the less likely two fingerprints overlap heavily.

(2) The testing accuracy is also monotonically decreasing w.r.t. the bit length of fingerprints. This is due to the generalization issue of fingerprint sampling, as aforementioned.

(3) The testing accuracy is empirically decreasing more slowly at the beginning and then faster than its bottom line requirement w.r.t. bit length. We, therefore, pick the bit length 128 as the optimal choice for which the gap between the two plots is maximized. We stick to this for all our experiments.

(4) Considering together our detection bitwise accuracy $\geq$0.991 and our fingerprint bit length 128, we derive in principle our mechanism can hold $2^{128\times0.991} \approx 10^{36}$ distinct fingerprints and therefore can result in such a large capacity of identifiable generators.

6.4.4 Scalability

Scalability is one of the advantageous properties of our mechanism: During training, we can efficiently instantiate a large capacity of generators with arbitrary fingerprints on the fly, so that fingerprint detection generalizes well during testing. To validate this, we compare to baselines where we intentionally downgrade our method with access to only a fixed set of fingerprints. Without losing representativeness, these baselines stand for the category of non-scalable fingerprinting methods that have to re-train a generator instance for each fingerprint, e.g. [3]. We cannot directly compare to [3] because it is impractical

Fingerprint set size	Training acc ⇑	Testing acc ⇑
10	1.000	0.512
100	1.000	0.537
1k	1.000	0.752
10k	0.990	0.988
100k	0.983	0.981
Sampling on the fly	0.991	0.991

Table 6.2: Fingerprint detection in bitwise accuracy on CelebA during training and testing. ⇑ indicates a higher value is more desirable. Detection starts to generalize with 10k fingerprint training samples.

(time-consuming) to instantiate a large number of their generators for analysis.

From Table 6.2 we show that fingerprint detection fails to generalize unless we can instantiate generators with 10k or more fingerprint samples. This indicates the necessity to equip GANs with an efficient and scalable fingerprinting mechanism, preferably the one on the fly. The results show that we can stick to the principle to reach a capacity of 10^{36}.

6.4.5 Robustness and immunizability

Robustness against image perturbations is another advantageous property of our mechanism. When it does not hold for some perturbations, the immunizability property compensates for it.

The motivation to validate the robustness and immunizability of fingerprint detection lies in the fact that deep fakes in the open end may undergo post-processing environments and result in quality deterioration. Following the protocol in [2], we evaluate the robustness against five types of image perturbation: cropping and resizing, blurring with Gaussian kernel, JPEG compression, additive Gaussian noise, and random combination of them.

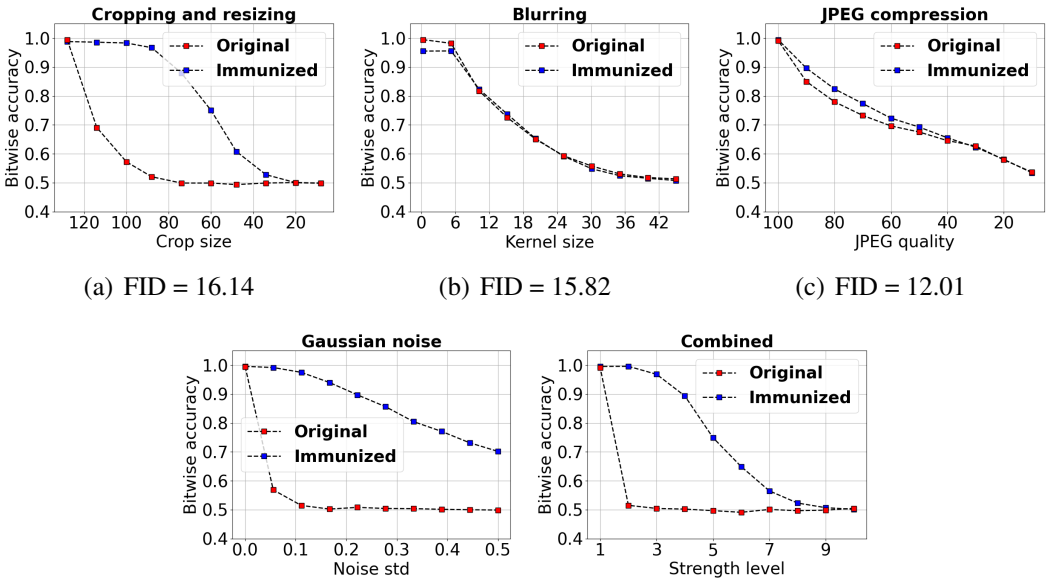
We consider two versions of our model: the original version and the immunized version. An immunized model indicates that during training we augment generated images with the corresponding perturbation in random strengths before feeding them to the fingerprint decoder.

It is worth noting that we, standing for the model owners, are regarded as the management hub of the experiments. Except for the fingerprinted generator instances, none of the encoder, decoder, and training data are accessible to the public. Therefore, the robustness against perturbation has to be experimented with the black-box assumption, as protocoled in [2]. In other words, white-box perturbations such as adversarial image modifications [261] and fingerprint overwriting, which requires access to the encoder, decoder, and/or training data, are not applicable in our scenario.

We plot in Figure 6.5 the fingerprint detection accuracy of our original and immunized models w.r.t. the strength of each perturbation. We find:

(1) For all the perturbations, fingerprint detection accuracy drops monotonically as we increase the strength of perturbation. For some perturbations in red plots, i.e., blurring and JPEG compression, accuracy drops slowly in a reasonably large range. We consider accepting accuracy $\geq$75%. As a result, the robust working range under blurring is: Gaussian blur kernel size $\sim [0, 7]$; under JPEG compression is: JPEG quality $\sim [80, 100]$. Usually, the images turn not functional with perturbations heavier than this range. We, therefore, validate the robustness of our original model against blurring and JPEG compression.

(2) For the other perturbations, although our original model is not robust enough, immunization (perturbed augmentation) compensates significantly in blue dots. We consider



(a) FID = 16.14 (b) FID = 15.82 (c) FID = 12.01

(d) FID = 34.78 (e) FID = 18.40

Figure 6.5: Red plots show, on CelebA, the fingerprint detection of our original model in bitwise accuracy w.r.t. the strength of perturbations. Blue plots show those of our immunized models. We consider accepting accuracy $\geq$75%. Therefore, our model is robust against blurring and JPEG compression, and is immunizable against cropping, Gaussian noise, and the combined perturbation. FID under each plot indicates the fidelity of each immunized model. FID of our original model is 11.50.

accepting accuracy $\geq$75%. As a result, the immunized working range under cropping is: cropping size $\sim [60, 128]$; under Gaussian noise is: noise standard deviation $\sim [0.0, 0.4]$; under combined perturbation is: the combination of the robust or immunized working ranges aforementioned. Usually the images turn not functional with perturbations heavier than this range. We, therefore, validate the immunizability of our model against cropping, Gaussian noise, and the combined perturbation.

(3) We also report the FID of each immunized model in the sub-captions. Augmentation with perturbations indeed has a side effect on the adversarial training and consequently on the fidelity. However, except for Gaussian noise immunization, the degrading from the others is to a reasonably small extent. We reason the challenge roots in the instability of

adversarial training: the minimax formulation and alternating gradient ascent-descent.

6.4.6 Deep fake detection and attribution

We have justified the effectiveness, robustness, and immunizability of our method for fingerprint detection. It in turn benefits our initial motivation: deep fake detection and attribution. The former task is a binary classification problem to distinguish between real and fake. The latter task is to further finely label the source of a generated image. We merge the two tasks into one with 1+N classes: 1 real-world source and N GAN sources, where N can be extremely large, as large as our capacity 10^{36} in Section 6.4.3.

Unlike previous methods that have to rely on inherent differences between real and fake [1, 2, 218, 220], our method *proactively* encodes identifiable fingerprints into generator instances and consequently into the generated images. Then the tasks are converted to verifying if one decoded fingerprint is in our database or not. This is achieved by comparing the decoded fingerprint to each fingerprint in the database given a threshold of bit overlap. According to our $\geq$0.991 fingerprint detection accuracy, it should be reliable to set the threshold at $128 \times 0.95 \approx 121$. If the bit overlap is larger than this threshold, the fingerprint is verified in the database. Then the attribution is trivial because we can directly look up the generator instance according to the fingerprint. If the fingerprint is not in the database, it should be a random fingerprint decoded from a real image. We use our immunized model against the combined perturbations in Section 6.4.5. We assume the testing images are within its wide working ranges.

Baselines. We compare to two state-of-the-art deep fake classifiers [1, 2] as learning-

Method	Closed world #GANs 1	10	100	Open world #GANs 1	10	100
[2]	0.997	0.998	0.955	0.893	0.102	N/A
[1]	0.890	N/A	N/A	0.883	N/A	N/A
[3]	1.000	1.000	N/A	1.000	1.000	N/A
Ours	1.000	1.000	1.000	1.000	1.000	1.000

Table 6.3: Deep fake detection and attribution accuracy on CelebA. A higher value is more desirable. It is impractical to train too many binary classifiers for [2] when the number of GANs is large (e.g. 100) in the open world. It is neither impractical to train too many fingerprinted generators (e.g. 100) for [3]. [1] is not applicable for deep fake attribution (i.e. $N > 1$).

based baselines *passively* relying on inherent visual clues. Because a learning-based method can only enumerate a small fixed set of training labels, we consider two scenarios for it: closed world and open world. The difference is whether the testing GAN sources are seen during training or not. This does not matter to our method because ours can work with any $N \leq 10^{36}$. For the closed world, we train/evaluate a baseline classifier on 10k/1k images from each of the $N+1$ sources. For the open world, we train $N+1$ 1-vs-the-other binary classifiers, and predict as "the other" label if and only if all the classifiers predict negative results. We test on 1k images from each of the real source or N unseen GAN sources.

We in addition refer to [3] in comparisons as the other *proactive* but *indirect* model fingerprinting baseline.

Results. From Table 6.3 we find:

(1) Deep fake detection and attribution based on our fingerprints perform equally perfectly ($\sim 100\%$ accuracy) to most of the baselines in the closed world when the number of GAN sources is not too large. However, when $N = 100$, [3] is not applicable due to

its limited efficiency and scalability. Neither is [1] due to its binary classification nature.

(2) Open world is also a trivial scenario to our method but challenges the baseline classifiers [1, 2]. When the number of unseen GAN sources increases to 10, [2] even degenerates close to random guess. This is a common generalization issue of the learning-based method. [3] is still impractical when N is large.

(3) Since deep fake detection and attribution is a trivial task to our method, it makes our advantages independent of the evolution of GAN techniques. It guarantees to verify decoded fingerprints to the model owner's fingerprint database and trace the responsibility of model misuse. This suggests a novel direction for model owner's responsible disclosure.

6.4.7 Ablation study on modulation

For completeness, as an ablation study, we investigate the effectiveness of fingerprint detection and fidelity of generation when modulating fingerprint embeddings to different generator layers (resolutions).

From Table 6.4 we find:

(1) For effectiveness, the optimal single layer to modulate fingerprints appears in one of the middle layers, specific to datasets: 16×16 for CelebA and 8×8 for LSUN Bedroom. But our all-layer modulation can achieve comparable or better performance. This should be consistent with different datasets because fingerprint detection turns more effective when we encode fingerprints to more parts of the generator.

(2) For fidelity, the side effect of fingerprinting is less significant if modulation happens in the shallower layer. This is because fingerprinting and generation are distinct

Layer	CelebA		LSUN Bedroom	
	Bit acc ⇑	FID ⇓	Bit acc ⇑	FID ⇓
4×4	0.953	12.06	0.693	21.44
8×8	0.981	12.06	0.950	21.15
16×16	0.993	11.90	0.935	20.98
32×32	0.991	11.07	0.894	20.24
64×64	0.972	10.77	0.816	19.85
128×128	0.946	10.67	0.805	19.67
Ours (all layers)	0.991	11.50	0.993	20.50

Table 6.4: Fingerprint detection in bitwise accuracy and generation fidelity in FID w.r.t. the layer to modulate fingerprints. ⇑/⇓ indicates a higher/lower value is more desirable.

tasks, and a shallower modulation leads to less crosstalk. However, considering the FID variance is not significant in general, we regard all-layer modulation as a desirable trade-off between effectiveness and fidelity.

6.5 Conclusion

One sustainable solution to mitigate the misuse of deep fakes is to enable a responsible disclosure of generative models. We achieve this by a novel fingerprinting mechanism. It allows an efficient and scalable ad-hoc generation of a large population of models with distinct fingerprints. Experiments show that our method fulfills several key properties: effectiveness, fidelity, large capacity, scalability, robustness, and immunizability. We validate its saturated performance and advantages over previous learning-based discriminative methods in the deep fake detection and attribution tasks, which makes it independent of the evolution of GAN techniques and agnostic to the other detection baselines.

6.6 Acknowledgement

Ning Yu is partially supported by Twitch Research Fellowship. This work is also supported, in part, by the US Defense Advanced Research Projects Agency (DARPA) Media Forensics (MediFor) Program under FA87501620191 and Semantic Forensics (SemaFor) Program under HR001120C0124. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA. We acknowledge the Maryland Advanced Research Computing Center for providing computing resources. We thank David Jacobs, Matthias Zwicker, Abhinav Shrivastava, and Yaser Yacoob for constructive advice in general.

Chapter 7: Inclusive GAN: Improving Data and Minority Coverage in Generative Models

7.1 Introduction

Photorealistic image generation has increasingly become reality, thanks to the emergence of large-scale datasets [18, 19, 20] and deep generative models [64, 145, 248, 249]. However, these advances have come at a cost: there could be potential biases in the learned model against underrepresented data subgroups [57, 58, 59, 60, 61]. The biases are rooted in the inevitable imbalance in the dataset [62], which are preserved or even exacerbated by the generative models [58]. In particular, reconstructive (non-adversarial) generative models like variational autoencoders (VAEs) [248, 262] can preserve data biases against minorities due to their objective of reproducing the frequencies images occur in the dataset, while adversarial generative models (GANs) [13, 132, 145, 158, 263, 264, 265, 266, 267] can implicitly disregard infrequent images due to the well-established problem of mode collapse [63, 64], thereby further introducing model biases on top of data biases. This issue is particularly acute from the perspective of minority inclusion, because training data associated with minority subgroups by definition do not form dominant modes. Consequently, data from minority groups are rare to begin with,

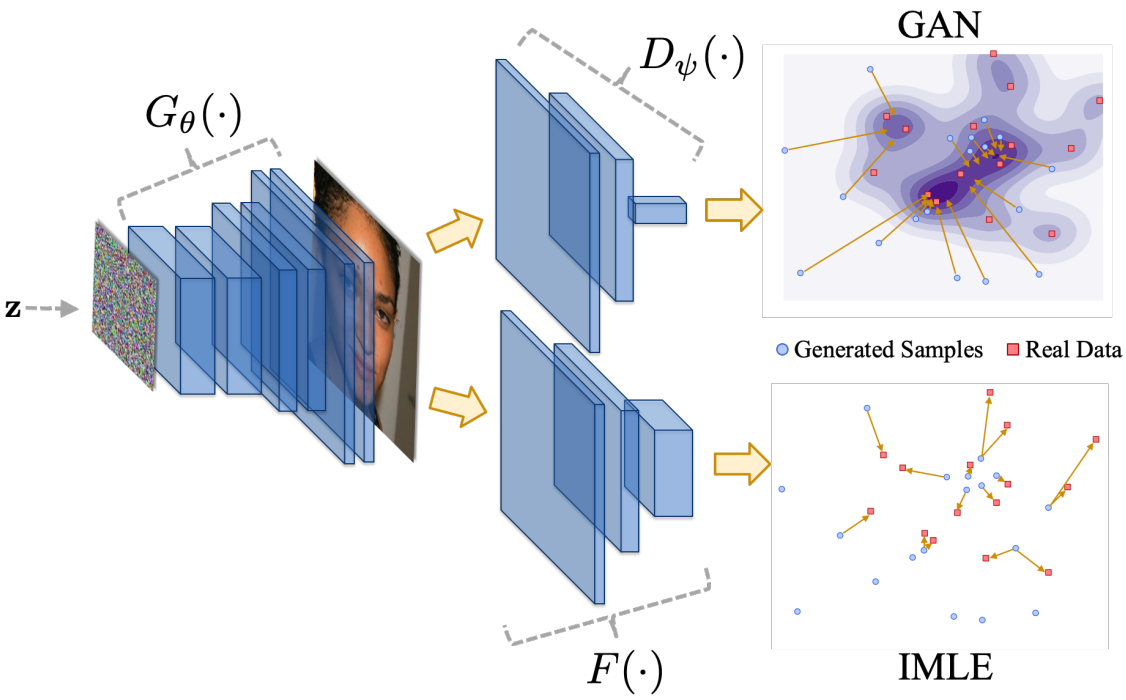


Figure 7.1: The diagram of our method. It harmonizes adversarial (GAN) and reconstructive (IMLE) training in one framework without introducing an auxiliary encoder. GAN guides arbitrary sampling towards generating realistic appearances approximate to some real data while IMLE ensures data coverage where there are always generated samples approximate to each real data. See Section 7.3.3 for more details where G_θ and D_ψ represent the trainable generator and discriminator in a GAN, and F represents a distant metric, in some cases, a pre-trained neural network.

and would not be capable of being produced by the generative model at all due to mode collapse.

In this work, we aim to improve the *comprehensive* performance of the state-of-the-art generative models, with a specific focus on their coverage of minority subgroups. We start with an empirical study on the correlation between data biases and model biases, and then formalize the objective of alleviating model bias in terms of improving data coverage, in particular over the minority subgroups. We propose a new method known as IMLE-GAN that achieves competitive image quality while ensuring improved coverage of minority groups.

Our method harmonizes adversarial and reconstructive generative models, in the process combining the benefits of both. Adversarial models have evolved to generate photorealistic results, whereas reconstructive models offer guarantees on data coverage. We build upon one of the state-of-the-art implementations of adversarial models, i.e., StyleGAN2 [267], and incorporate it with the Implicit Maximum Likelihood Estimation (IMLE) framework [64], which is at its core reconstructive. See Figure 7.1 for a diagram.

Different from the existing hybrid generative models [63, 249, 268, 269] that require training an auxiliary encoder network alongside a vanilla GAN, our method operates purely with the standard components of a GAN. This brings two main benefits: (1) it sidesteps the complication from combining the minimax objective used by adversarial models and the pure minimization objective used by reconstructive models, and (2) it avoids carrying over the practical issues of training auxiliary encoder, like posterior collapse [270, 271], which can cause the regression-to-the-mean problem, leading to blurry images.

We validate our method with thorough experiments and demonstrate more comprehensive data coverage that goes beyond that of existing state-of-the-art methods. In addition, our method can be flexibly adapted to ensure the inclusion of specified minority subgroups, which cannot be easily achieved in the context of existing methods.

Contributions. We summarize our main contributions as follows: (1) we study the problem of underrepresented minority inclusion and formalize it as a data coverage problem in generative modeling; (2) we present a novel paradigm of harmonizing adversarial and reconstructive modeling for improving data coverage; (3) our experiments set up a new suite of state-of-the-art performance in terms of covering both seen and unseen data; and (4) we develop an effective extension of our technique to ensure inclusion of the specified

minority subgroups.

7.2 Related Work

Bias mitigation efforts for machine learning. Bias in machine learning results from data imbalance, which can be detected and alleviated by three categories of approaches: The pre-process approaches that purify data from bias before training [272, 273, 274, 275], the in-process approaches that enforce fairness during training with constraints or regularization in the objectives [62, 276, 277, 278], and the post-process approaches that adjust the output from a learned model [279, 280]. A comprehensive survey [281] articulates this taxonomy. These approaches target biases in classification and cannot be adapted to generative modeling.

Bias mitigation efforts for generative models. There have been relatively few papers [57, 58, 59, 60, 61] that focus on biases in generative models. [57, 59, 60], motivated from benefiting a downstream classifier, mainly aim for fair generation conditioned on attribute inputs, in terms of yielding allocative decisions and/or removing the correlation between generation and attribute conditions. [58] focuses on understanding the inductive bias so as to investigate the generalization of generative models. [61] proposes an importance weighting strategy to compensate for the biases of learned generative models. Different from their goals and solutions that equalize performance across different data subgroups possibly at the cost of overall performance, we instead aim to improve the overall data coverage, with a specific purpose of ensuring more significant gains over the underrepresented minorities.

Data coverage in GANs. GANs are finicky to train because of the minimax formulation and the alternating gradient ascent-descent. In addition, GANs are known to exhibit mode collapse, where the generator only learns to generate a subset of the modes of the underlying data distribution. To alleviate mode collapse in GANs, some methods propose to improve the minimax loss function [65, 66, 282, 283], some methods apply constraints or regularization terms along with the minimax objectives [284, 285, 286, 287, 288], and some other methods aim to modify the discriminator designs [13, 289, 290, 291]. These directions are orthogonal to our research while, in principle, demonstrate less effective data coverage than the hybrid models below.

Data coverage in hybrid generative models. Reconstructive (non-adversarial) generative models like variational autoencoders (VAEs) [248, 262], on the other hand, are more successful at data coverage because they explicitly try to maximize a lower bound on the likelihood of the real data. This motivates a variety of designs for hybrid models that combine reconstruction and adversarial training. α-GAN [268] is trained to reconstruct pixels while VAEGAN [249] is trained to reconstruct discriminator features. ALI [264], BiGAN [265], and SVAE [292] propose to instead jointly match the bidirectional mappings between data and latent distributions. VEEGAN [63] is designed with reconstruction in the latent space, in the purpose of avoiding the metric dilemma in the data space. Hybrid models benefit for mode coverage, but deteriorate generation fidelity in practice, because of their dependency on auxiliary encoder networks. In contrast, our method follows the idea of hybrid models, but avoids an encoder network and instead apply all training back-propagation through the generator. A recent non-adversarial generative framework, Implicit Maximum Likelihood Estimation (IMLE) [64], satisfies our design. We discuss

more about the advantages of IMLE in Section 7.3.2.

7.3 Inclusive GAN for Data and Minority Coverage

Our method is a novel paradigm of harmonizing the strengths of adversarial (Section 7.3.1) and reconstructive generative models (Section 7.3.2) that avoids mode collapse. The harmonization efforts (Section 7.3.3) are necessary and non-trivial due to the incompatibility between the two. In Section 7.3.4 we show the straightforward adaptation of our method to improve minority inclusion.

7.3.1 Adversarial Generation: GANs

Photorealistic image generation can be viewed as the problem of sampling from the unknown probability distribution of real-world images. Generative Adversarial Networks (GANs) [145] introduce an elegant solution for distribution estimation, which is formulated as a discriminative classification problem, and enables supervised learning methods to be used for this task.

A GAN consists of two deep neural networks: a generator $G_\theta : \mathbb{R}^d \mapsto \mathbb{R}^D$ and a discriminator $D_\psi : \mathbb{R}^D \mapsto [0, 1]$. The generator maps a latent noise vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ to an image, and the discriminator predicts the probability that the image it sees is real. The real ground truth images are denoted as $\mathbf{x} \sim \hat{p}(\mathbf{x})$, sampled from an unknown distribution $\hat{p}(\mathbf{x})$. The discriminator is trained to maximize classification accuracy while the generator is trained to produce images that can fool the discriminator. More precisely, the objective

is shown in Eq. 7.1:

$$\min_{\theta} \max_{\psi} L^{adv}(\theta, \psi) = \mathbb{E}_{\mathbf{x}\sim\hat{p}(\mathbf{x})} [\log D_{\psi}(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)} [\log(1 - D_{\psi}(G_{\theta}(\mathbf{z})))] \quad (7.1)$$

Unfortunately, GANs are unstable to train and suffer from mode collapse: While each generated sample gets to pick a mode it is drawn to, each mode does not get to pick a generated sample. After training, the generator will not be able to generate samples around the "unpopular" modes.

Minority modes are precisely the "unpopular" modes that are more likely to be collapsed. As shown in Section 7.4.3 and Figure 7.2, minority subgroups with diverse appearances indeed bring more challenges to generative modeling and are allocated worse coverage compared to the others. Therefore, we propose to leverage reconstructive models to improve the coverage of minority subgroups.

7.3.2 Reconstructive Generation: IMLE

Our novel paradigm is based on a recent reconstructive framework, Implicit Maximum Likelihood Estimation (IMLE) [64], that favors complete mode coverage. IMLE avoids mode collapse by reversing the direction in which generated samples are matched to real modes. In GANs, each generated sample is effectively matched to a real mode. In IMLE, each real mode is matched to a generated sample. This ensures that all real modes, including each underrepresented minority mode, are matched, and no real mode is left out.

Mathematically, IMLE tackles the optimization problem in Eq. 7.2:

$$\min_{\theta} \mathbb{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_m \sim \mathcal{N}(\mathbf{0},\mathbf{I}_d)} \left[\mathbb{E}_{\mathbf{x}\sim\hat{p}(\mathbf{x})} \left[\min_{i\in\{1,\ldots,m\}} \|G_\theta(\mathbf{z}_i) - \mathbf{x}\|_2^2 \right] \right] \tag{7.2}$$

$$= \min_{\theta} \mathbb{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_m \sim \mathcal{N}(\mathbf{0},\mathbf{I}_d)} \left[\mathbb{E}_{\mathbf{x}\sim\hat{p}(\mathbf{x})} \left[\|G_\theta(\mathbf{z}^*(\mathbf{x})) - \mathbf{x}\|_2^2 \right] \right], \tag{7.3}$$

$$\text{where } \mathbf{z}^* = {}_{i\in\{1,\ldots,m\}} \|G_\theta(\mathbf{z}_i) - \mathbf{x}\|_2^2 \tag{7.4}$$

The joint optimization is achieved by alternating between the two decoupled phases until convergence. The first phase corresponds to the inner optimization, where we search for each $\mathbf{x}$ the optimal $\mathbf{z}^*(\mathbf{x})$ from the latent vector candidates, given a fixed G_θ. This is implemented by the Prioritized DCI [293], a fast nearest neighbor search algorithm. The second phase corresponds to the outer optimization, where we train the generator in the regular back-propagation manner, given pairs of $(\mathbf{x}, \mathbf{z}^*(\mathbf{x}))$.

One significant advantage of IMLE over the other reconstructive models is the elimination of the need for an auxiliary encoder. The encoder encourages mode coverage but at the cost of either deviating the latent sampling distribution from the original prior (in VAEGAN [249]) or absorbing the training gradients before substantially back-propagating to the generator (in VEEGAN [63]). Unlike them, IMLE directly samples latent vector from a natural prior during training and encourages explicit reconstruction fully upon the generator.

7.3.3 Harmonizing Adversarial and Reconstructive Generation: IMLE-GAN

Below we propose a way to harmonize adversarial training with the IMLE framework, so as to ensure both generation quality (precision) and coverage (recall) simultaneously.

The vanilla hybrid model between IMLE and GAN is to directly add the adversarial loss in Eq. 7.1 to the non-adversarial loss in Eq. 7.2. This has two problems because of (1) differences in the domains over which latent vectors are sampled and (2) differences in the metric spaces on which GAN and IMLE operate. For (1), in the case of GAN, a different latent vector is randomly sampled every iteration, whereas in the case of IMLE, many latent vectors are sampled at once (over which matching is performed) and are kept fixed for many iterations. The former gives up control over which data point each latent vector is asked to generate by the discriminator, but can avoid overfitting to any one latent vector. The latter explicitly controls which latent vectors are matched to data points, but can overfit to the set of matched latent vectors until they are resampled. For (2), in the case of GAN, the discriminator takes the inner product between the features and the weight vector of the last layer to produce a realism score, and so it effectively operates on features of images; on the other hand, in the case of IMLE, matching is performed on raw pixels.

To bridge the gap in losses, we propose two adaptations that better harmonize the GAN and IMLE objectives. First, to make the domain over which latent vectors are sampled denser, we augment the matched latent vectors with random linear interpolations. Second, to make the spaces on which the two losses are computed more comparable, we

measure the reconstruction loss in a deep feature space instead of pixel space, such that it contains a comparable amount and level of semantic information to that used by the discriminator. Mathematically, our goal is to optimize Eq. 7.5:

$$\min_{\theta} \max_{\psi} L^{adv}(\theta, \psi) + \mathbb{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_m \sim \mathcal{N}(\mathbf{0},\mathbf{I}_d)} \left[\lambda L^{rec}(\theta) + \beta L^{itp}(\theta)\right] \tag{7.5}$$

Here $L^{adv}(\theta, \psi)$ is as defined in Eq. 7.1,

$$L^{rec}(\theta) = \mathbb{E}_{\mathbf{x}\sim\hat{p}(\mathbf{x})} \left[\|F(G_\theta(\mathbf{z}^*(\mathbf{x}))) - F(\mathbf{x})\|_2^2\right] \tag{7.6}$$

$$\text{where } \mathbf{z}^*(\mathbf{x}) = _{i\in\{1,\ldots,m\}} \|F(G_\theta(\mathbf{z}_i)) - F(\mathbf{x})\|_2^2, \tag{7.7}$$

$$\text{and } L^{itp}(\theta) = \mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}\sim\hat{p}(\mathbf{x}),\alpha\sim U[0,1]} \left[\alpha\|F(G_\theta(\mathbf{z}^*(\alpha, \mathbf{x}, \tilde{\mathbf{x}}))) - F(\mathbf{x})\|_2^2 + \right. \tag{7.8}$$

$$\left. (1-\alpha)\|F(G_\theta(\mathbf{z}^*(\alpha, \mathbf{x}, \tilde{\mathbf{x}}))) - F(\tilde{\mathbf{x}})\|_2^2\right] \tag{7.9}$$

$$\text{where } \mathbf{z}^*(\alpha, \mathbf{x}, \tilde{\mathbf{x}}) = \alpha\mathbf{z}^*(\mathbf{x}) + (1-\alpha)\mathbf{z}^*(\tilde{\mathbf{x}}) \tag{7.10}$$

Here Eq. 7.6 generalizes Eq. 7.3 by computing distance in feature space, where $F(\cdot)$ is a fixed function to compute features of images. Eq. 7.8 and 7.9 defines the interpolation loss, which linearly interpolates between two matched latent vectors $\mathbf{z}^*(\mathbf{x}), \mathbf{z}^*(\tilde{\mathbf{x}})$ (as shown in Eq. 7.10) and tries to make the image generated from the interpolated latent vector $\mathbf{z}^*(\alpha, \mathbf{x}, \tilde{\mathbf{x}})$ similar to the two ground truth images $\mathbf{x}, \tilde{\mathbf{x}}$ that correspond to the latent vectors at the endpoints. The weight on the distance to each ground truth image depends on how close the interpolated latent vector is to the endpoint, which is denoted by α. λ and β are used to balance each loss term. We experiment with four possible feature spaces: raw pixels, discriminator features [249], Inception features [7], and LPIPS features (i.e.:

features such that the ℓ_2 distance between them is equivalent to the LPIPS perceptual metric [294]), and find LPIPS features perform the best.

Algorithm 1: IMLE-GAN with Minority Inclusion

Data: Real training data $\hat{p}(\mathbf{x})$ and a specified minority subgroup $\hat{q}(\mathbf{y})$
Result: A generator G_θ with specified minority inclusion performance
for epoch = $\{1, \ldots, E\}$ **do**
  if epoch % $S == 0$ **then**
    Sample $\mathbf{z}_1, \ldots, \mathbf{z}_m \sim \mathcal{N}(0, \mathbf{I}_d)$ i.i.d.;
    for $\mathbf{y}_j \sim \hat{q}(\mathbf{y})$ **do**
      $\mathbf{z}^*(\mathbf{y}_j) \leftarrow \arg\min_{i \in \{1,\ldots,m\}} ||F(G_\theta(\mathbf{z}_i)) - F(\mathbf{y}_j)||_2^2$;
  for $\mathbf{x}_k \sim \hat{p}(\mathbf{x})$ and $\mathbf{y}_i, \mathbf{y}_j \sim \hat{q}(\mathbf{y})$ **do**
    Sample $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_d)$;
    $L^{adv} \leftarrow \log D_\psi(\mathbf{x}_k) + \log(1 - D_\psi(G_\theta(\mathbf{z})))$;
    Sample $\delta_i, \delta_j \sim \mathcal{N}(0, \sigma \mathbf{I}_d)$ i.i.d.;
    $\mathbf{z}_i^* \leftarrow \mathbf{z}^*(\mathbf{y}_i) + \delta_i$;
    $\mathbf{z}_j^* \leftarrow \mathbf{z}^*(\mathbf{y}_j) + \delta_j$;
    $L^{rec} \leftarrow \frac{1}{2}(||F(G_\theta(\mathbf{z}_i^*)) - F(\mathbf{y}_i)||_2^2 + ||F(G_\theta(\mathbf{z}_j^*)) - F(\mathbf{y}_j)||_2^2)$;
    Sample $\alpha \sim U[0, 1]$;
    $\mathbf{z}_{ij}^* = \alpha \mathbf{z}_i^* + (1 - \alpha)\mathbf{z}_j^*$;
    $L^{itp} \leftarrow \alpha ||F(G_\theta(\mathbf{z}_{ij}^*)) - F(\mathbf{y}_i)||_2^2 + (1 - \alpha)||F((G_\theta(\mathbf{z}_{ij}^*)) - F(\mathbf{y}_j)||_2^2$;
    $L \leftarrow L^{adv} + \lambda L^{rec} + \beta L^{itp}$;
    $\psi = \psi + \eta \nabla_\psi L$;
    $\theta = \theta - \eta \nabla_\theta L$;

7.3.4 Minority Coverage in IMLE-GAN

IMLE-GAN framework is designed to improve the overall mode coverage. One benefit compared to other hybrid models is that it is straightforward to adapt it for minority inclusion. We simply need to replace the empirical distribution over the entire dataset $\hat{p}(\mathbf{x})$ with a distribution $\hat{q}(\mathbf{x})$ whose support only covers a specified minority subgroup (i.e.: $\text{supp}(\hat{q}) \subset \text{supp}(\hat{p})$) in Eq. 7.6 and 7.8 (for reconstructive training) and leave Eq. 7.1 unchanged (for adversarial training). This ensures an explicit coverage over the minority

while still carrying out the approximation to the entire real data. This comes with another advantage: because $\hat{q}(\mathbf{x})$ in practice has support over a much smaller set than $\hat{p}(\mathbf{x})$, there is less data imbalance and variance within the support of $\hat{q}(\mathbf{x})$ than in $\hat{p}(\mathbf{x})$, thereby requiring less model capacity to model. As a result, covering $\hat{q}(\mathbf{x})$ should be easier than covering $\hat{p}(\mathbf{x})$, and so the perceptual quality of samples tend to improve.

We summarize our IMLE-GAN algorithm with minority inclusion in Algorithm 1, where E is the number of training epochs, S indicates how often (in epochs) to update latent matching, m is the pool size of the latent vector candidates, δ_i, δ_j are the additive Gaussian perturbations, and η is the learning rate.

7.4 Experiments

We articulate the experimental setup in Section 7.4.1. In Section 7.4.2 we start with preliminary validation on Stacked MNIST dataset [282], an easy and interpretable task. In Section 7.4.3 we conduct empirical study to analyze the correlation between data bias and model bias. In Section 7.4.4 we perform comprehensive evaluation and comparisons on CelebA dataset [19], and finally specify minority inclusion applications in Section 7.4.5.

7.4.1 Setup

Datasets. For preliminary study, we employ Stacked MNIST dataset [282] for explicit data coverage evaluation. 240,000 RGB images in the size of 32×32 are synthesized by stacking three random digit images from MNIST [295] along the color channel, resulting in 1,000 explicit modes in a uniform distribution.

We conduct our main experiments on CelebA human face dataset [19], where the 40 binary facial attributes are used to specify minority subgroups. We sample the first 30,000 images in the size of 128×128 for GAN training, and sample the last 3,000 or 30,000 images for validation.

GAN backbone. We build our IMLE-GAN framework on the state-of-the-art StyleGAN2 [267] architecture for unconditional image generation. We reuse all their default settings.

Baseline methods. Besides the backbone StyleGAN2 [267], we also compare our method to eight techniques that show improvement in data coverage and/or generation diversity: SNGAN [13], Dist-GAN [286], DSGAN [287], PacGAN [288], ALI [264], VAEGAN [249], α-GAN [268], and VEEGAN [63]. For VAEGAN which originally involves image reconstruction in the discriminator feature space, we also experiment with three other distance metrics as discussed in Section 7.3.3. For fair comparisons, we replace the original architectures used in all methods with StyleGAN2.

Evaluation. For Stacked MNIST, following [63, 282], we report the number of generated modes that is detected by a pre-trained mode classifier, as well as the KL divergence between the generated mode distribution and the uniform distribution. The statistics are calculated from 240,000 randomly generated samples.

For CelebA, Fréchet Inception Distance (FID) [118] is used to reflect both data quality (precision) and coverage (recall) in an entangled manner. We also explicitly measure the Precision and Recall [296] of a generative model w.r.t. the real dataset in the Inception space. Moreover, to emphasize on instance-level data coverage, we further include Inference via Optimization Measure (IvOM) [282] into our metric suite, which measures the mean reconstruction error from a generative model given each query image.

	# modes (max 1000) (⇑)	KL to uniform (⇓)
StyleGAN2 [267]	940	0.424
SNGAN [13]	571	1.382
DSGAN [287]	955	0.343
PacGAN [288]	908	0.638
ALI [264]	956	0.680
VAEGAN [249]	929	0.534
VEEGAN [63]	987	0.310
Ours LPIPS interp	**997**	**0.200**

Table 7.1: Comparisons on Stacked MNIST dataset. The statistics are calculated from 240,000 randomly generated samples. We indicate for each metric whether a higher (⇑) or lower (⇓) value is more desirable. We highlight the best performance in **bold**.

We also report the standard deviation of IvOM across 40 CelebA attributes, in order to evaluate the balance of generative coverage. For the generalization purpose, we evaluate over both the training set and a validation set (unseen during training).

7.4.2 Preliminary Study on Stacked MNIST

In a real-world data distribution, the notion of modes is difficult to quantize. We instead start with Stacked MNIST [282] where 1,000 discrete modes are unambiguously synthesized. This allows us to zoom in the challenge of mode collapse and facilitate a precise pre-validation.

We report the evaluation in Table 7.1. Our method narrows down the gap between experimental performance and the theoretical limit: It covers the most number of modes and achieves the closest mode distribution to the uniform distribution ground truth. This study validates the improved effectiveness of harmonizing IMLE with GAN, compared to the other GAN models or hybrid models, in terms of explicit mode/data coverage. This sheds the light and pre-qualifies to apply our method on more complicated real-world

datasets.

7.4.3 Empirical Study on Data and Model Biases

As discussed in Section 7.2, data biases lead to biases in generative models. Even worse, a model without attention to minorities can exacerbate such biases against allocating adequate representation capacities to them. In this empirical study, we first show the existence of biases across CelebA attributes in terms of sample counts and sample variance, and then correlate them to the biased performance of the backbone StyleGAN2 [267].

As shown in the left barplot of Figure 7.2, given the attribute histogram over 30,000 samples, 29 out of 40 binary attributes are more than 50% biased from the balance point (15,000 out of 30,000 samples with a positive attribute annotation, shown as the red dashed line). On the other hand, in the right barplot of Figure 7.2, we calculate the standard deviation of Inception features [7] of samples within each attribute, and notice a wide range spanning from 0.038 to 0.062.

Too few samples or too large appearance variance in one attribute discourages generative coverage for that attribute, and thus results in biases. To quantify the per-attribute coverage, we measure the mean IvOM [282] over positive training samples. A larger value indicates a worse coverage. In the middle barplot of Figure 7.2, we visualize the correlation between IvOM and the joint distribution of sample counts and sample variance. There is a clear gradient trend of IvOM when the samples of an attribute turn rarer and/or more diverse. To validate such a strong correlation, we first normalize the sample counts and sample variance across attributes by their means and standard

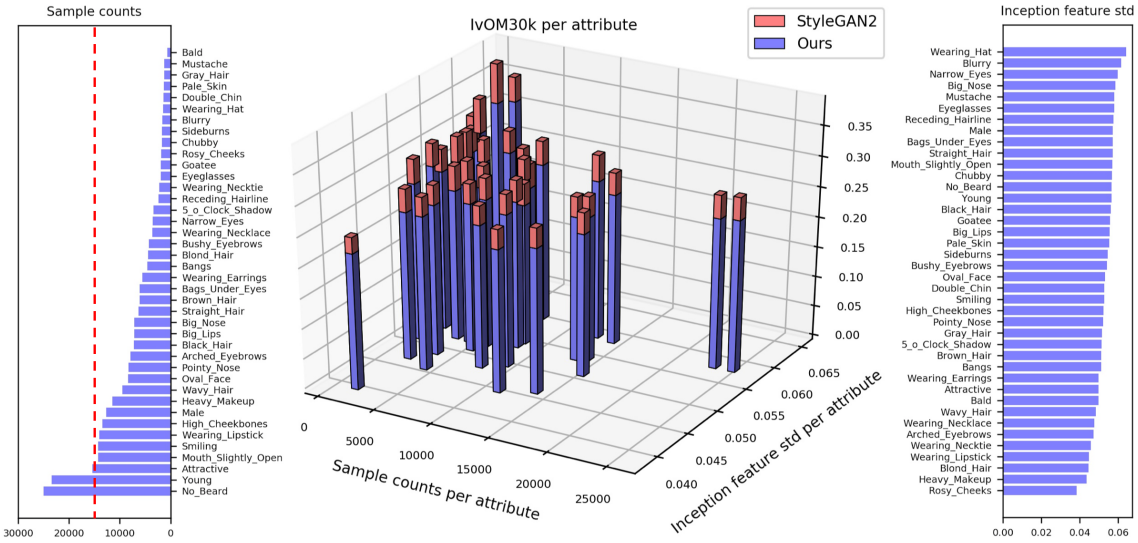


Figure 7.2: Visualizations for data and model biases. Left: Sorted CelebA attribute histogram with a balance point marked by the red dashed line. Right: Sorted Inception feature variance per attribute. Middle: Per-attribute mean IvOM over 30,000 CelebA training samples for StyleGAN2 (red) and for our method (blue), where each bar corresponds to one attribute.

deviations. Then we simply add them up as a joint variable vector, and calculate its Spearman's ranking correlation to the per-attribute IvOM. For StyleGAN2 (the red bar), the correlation coefficient of 0.75 indicates a strong correlation between data biases and model biases. This evidences the urgency to mitigate biases against the rare and diverse samples, in another word, to enhance the coverage over minority subgroups.

7.4.4 Comparisons on CelebA

In Section 7.3.3 we propose two strategies to harmonize adversarial and reconstructive training: the deep distance metric and the interpolation-based augmentation. We obtain: (1) LPIPS similarity shows near-top performance all around measures; and (2) interpolation-based augmentation consistently benefits all the measures in general for all the distance metrics. We therefore employ both into our full method.

Method	FID30k ⇓ Train	FID30k ⇓ Val	Precision30k ⇑ Train	Precision30k ⇑ Val	Recall30k ⇑ Train	Recall30k ⇑ Val	IvOM3k ⇓ Train	IvOM3k ⇓ Val	IvOM3k std ⇓ Train	IvOM3k std ⇓ Val
StyleGAN2 [267]	**9.37**	**9.49**	0.855	0.844	0.730	0.741	0.303	0.302	0.0268	0.0264
SNGAN [13]	13.32	13.24	0.792	0.787	0.631	0.616	0.325	0.322	0.0274	0.0261
Dist-GAN [286]	30.97	30.44	0.511	0.595	0.360	0.385	0.282	0.280	0.0220	0.0209
DSGAN [287]	14.29	14.00	0.868	0.862	0.679	0.724	0.301	0.300	0.0227	0.0220
PacGAN [288]	15.05	15.12	0.870	0.869	0.726	0.758	0.311	0.308	0.0256	0.0238
ALI [264]	10.09	10.06	0.842	0.867	0.688	0.710	0.298	0.297	0.0240	0.0245
VAEGAN [249] LPIPS	24.10	23.47	0.878	0.851	0.572	0.560	0.318	0.315	0.0284	0.0272
α-GAN [268]	12.65	12.53	0.803	0.810	0.757	0.763	0.267	0.267	0.0208	0.0192
VEEGAN [63]	16.34	16.13	0.752	0.768	0.660	0.695	0.260	0.269	**0.0190**	**0.0181**
Ours LPIPS interp	11.56	11.28	**0.927**	**0.941**	**0.849**	**0.848**	**0.255**	**0.262**	0.0193	0.0195
Ours *Eyeglasses*	13.54	14.43	0.914	0.910	0.890	0.895	0.255	0.265	0.0249	0.0193
Ours *Bald*	13.34	13.46	0.903	0.895	0.886	0.892	0.268	0.272	0.0381	0.0227
Ours *EN&HM*	15.18	15.00	0.885	0.891	0.830	0.842	0.268	0.270	0.0318	0.0277
Ours *BUE&HC&A*	14.27	13.85	0.878	0.874	0.871	0.884	0.262	0.266	0.0300	0.0254

Table 7.2: Comparisons on CelebA dataset. We indicate for each metric whether a higher (⇑) or lower (⇓) value is more desirable. The first part corresponds to the comparisons among different methods. For VAEGAN we report the results based on LPIPS distance metric. We highlight the best performance in **bold** and the second best performance with underline. We visualize the radar plots in Figure 7.3 for the comprehensive evaluation of each method over the validation set. The second part corresponds to our minority inclusion model variants in Section 7.4.5.

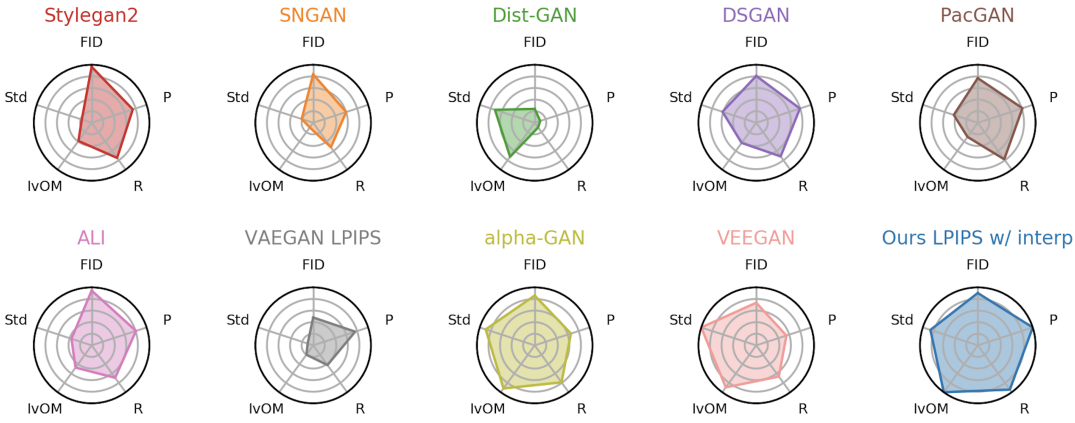


Figure 7.3: Radar plots for the first part of Table 7.2. "P" represents Precision, "R" represents Recall, and "Std" represents IvOM standard deviation. Values have been normalized to the unit range, and axes are inverted so that the higher value is always better.

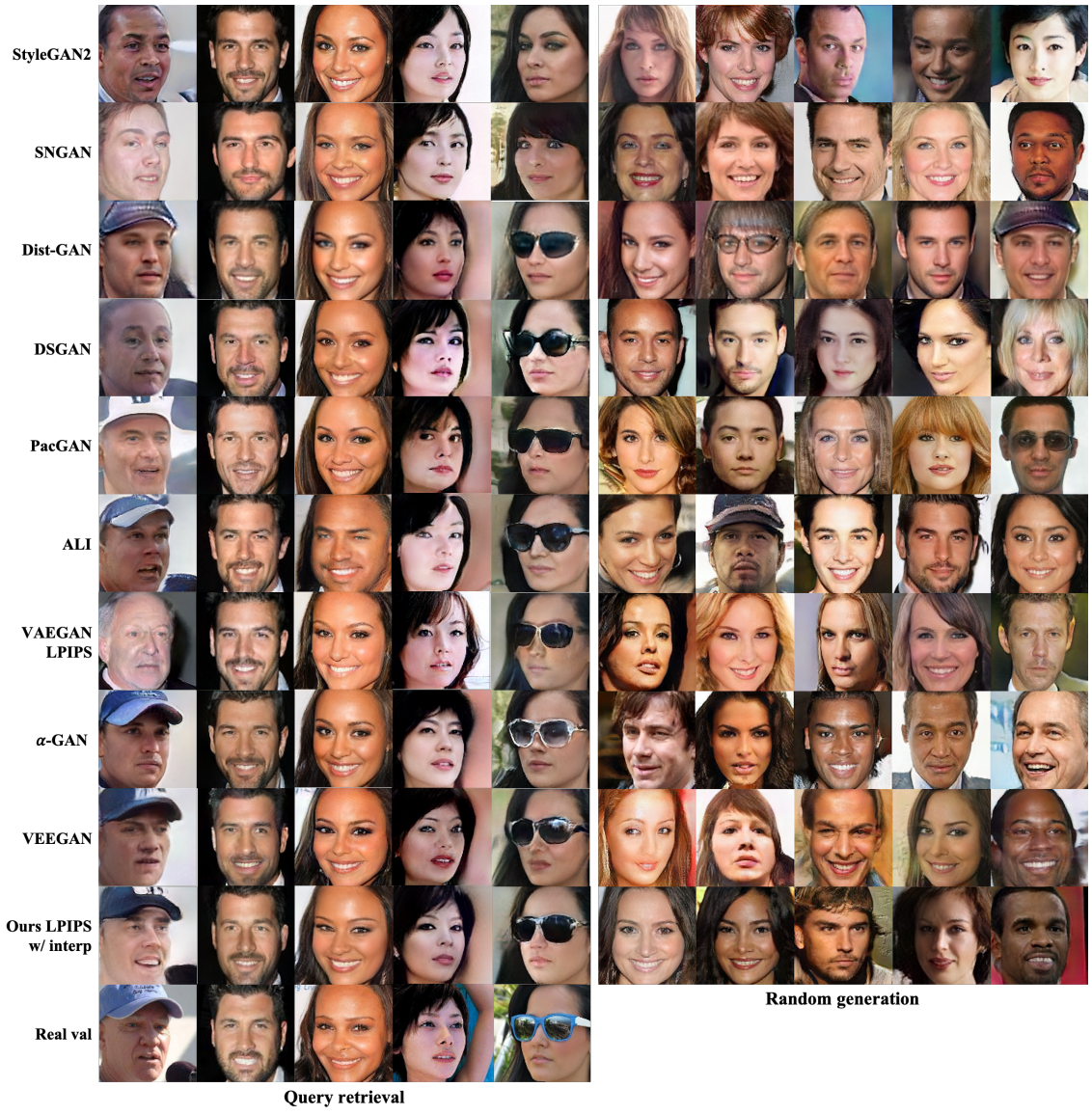


Figure 7.4: Reconstructed samples on the left (used for IvOM evaluation) and random generation samples on the right (used for FID, precision, and recall evaluation). The query images for reconstruction in the bottom left row are real and unseen during training.

To evaluate our data coverage performance in practice, we conduct comprehensive comparisons on CelebA [19] against baseline methods. The first part of Table 7.2 shows our comparisons. Figure 7.3 assists interpret the table. We find:

(1) FID is not a gold standard to reflect the entire capability of a generative model, as it ranks differently from the other metrics.

(2) Compared to the original backbone StyleGAN2 which achieves the second-best FID, our full method ("Ours LPIPS interp") trades slight FID deterioration for significant boosts in all the other metrics. This is meaningful because precision (FID) can be traded off at the expense of recall (Recall, IvOM) via the truncation trick used in [158, 267], while the opposite direction is infeasible.

(3) Our full method outperforms all the existing state-of-the-art techniques in terms of Precision, Recall, and IvOM, where the latter two are the key evidence for effective data coverage. The last radar plot in Figure 7.3 shows our method achieves near-top measures all around with the most balanced performance.

(4) Our method also achieves the top-3 performance in the standard deviation of per-attribute IvOM, indicating an equalized capacity across the attribute spectrum. The blue bars in the middle barplot of Figure 7.2 also visualize our method consistently outperforms StyleGAN2 (red bars) for all the attributes, in particular with more significant improvement for the minority subgroups.

(5) Figure 7.4 shows qualitative comparisons in terms of query reconstruction and uncurated random generation. StyleGAN2 suffers from mode collapse. For the collapsed modes, our method significantly improves the generation from non-existence of rare attributes to good quality (hat, sunglasses, etc.). Our method also demonstrates desirable

Arbitrary minority subgroup	Method	Precision1k minority only ⇑ Train	Precision1k minority only ⇑ Val	Recall1k minority only ⇑ Train	Recall1k minority only ⇑ Val	IvOM1k minority only ⇓ Train	IvOM1k minority only ⇓ Val
Eyeglasses (6%)	StyleGAN2 [267]	0.719	0.704	0.582	0.589	0.355	0.352
	Ours LPIPS interp	0.843	0.845	0.740	0.708	0.309	0.308
	Ours *Eyeglasses*	**0.904**	**0.919**	**0.897**	**0.892**	**0.261**	**0.288**
Bald (2%)	StyleGAN2 [267]	0.707	0.750	0.461	0.424	0.301	0.305
	Ours LPIPS interp	0.763	**0.783**	0.666	0.670	0.269	**0.273**
	Ours *Bald*	**0.779**	0.718	**0.842**	**0.810**	**0.189**	**0.273**
Narrow_Eyes &Heavy_Makeup (4%)	StyleGAN2 [267]	0.719	0.701	0.543	0.577	0.272	0.274
	Ours LPIPS interp	0.794	0.760	0.632	0.621	0.246	0.248
	Ours *EN&HM*	**0.799**	**0.766**	**0.698**	**0.696**	**0.194**	**0.244**
Bags_Under_Eyes &High_Cheekbones &Attractive (4%)	StyleGAN2 [267]	0.838	0.804	0.736	0.725	0.263	0.268
	Ours LPIPS interp	0.816	0.831	0.700	0.742	0.237	0.241
	Ours *BUE&HC&A*	**0.889**	**0.883**	**0.813**	**0.809**	**0.191**	**0.237**

Table 7.3: Comparisons on CelebA minority subgroups, where the percentages show their portion w.r.t. the entire population. The metrics are measured on the corresponding subgroups only. We indicate for each metric whether a higher (⇑) or lower (⇓) value is more desirable. We highlight the best performance in **bold**.

generation fidelity and diversity.

(6) All the conclusions above generalize well to unseen data, as evidenced by the "Val" columns in Table 7.2.

7.4.5 Extension to Minority Inclusion

We adapt our method for ensuring specific coverage over minority subgroups (Algorithm 1). Without introducing unconscious bias on the CelebA attributes, we arbitrarily specify four sets of attributes, the samples of which count for no more than 6% of the population, and therefore, constitute four minority subgroups respectively. The attribute sets and their portions are listed in the first column of Table 7.3.

To validate minority inclusion, we first compare our minority model variants over

Figure 7.5: Reconstructed samples according to different minority subgroups. The query images for reconstruction in the bottom row of each sub-figure are real from the training set.

the corresponding minority subsets against the backbone StyleGAN2 and against our general full model. See Table 7.3 for the results. Our minority variants consistently outperform the two baselines over all the minority subgroups. In Figure 7.5, our method reconstructs the minority attributes the most accurately, even for the subtle attributes like eye bags where StyleGAN2 fails. It validates better training data utilization of our minority models.

To validate the overall performance beyond minority subgroups, we show at the bottom of Table 7.2 the performance on the entire attribute spectrum. We conclude that the improvement of all our minority models comes at little or no compromise from their performance on the overall dataset.

7.5 Conclusion

In this paper, we formalized the problem of minority inclusion as one of data coverage and improved data coverage using a novel paradigm that harmonizes adversarial training (GAN) with reconstructive generation (IMLE). Our method outperforms state-of-the-art methods in terms of Precision, Recall, and IvOM on CelebA, and the improvement generalizes well on unseen data. We further extended our method to ensure explicit inclusion for minority subgroups at little or no compromise on overall full-dataset performance. We believe this is an important step towards fairness in generative models, with the aim to reduce and ultimately prevent discrimination due to model and data biases.

7.6 Acknowledgement

This project was partially funded by DARPA MediFor program under cooperative agreement FA87501620191 and by ONR MURI (N00014-14-1-0671). We thank Tero Karras and Michal Lukáč for sharing code. We also thank Richard Zhang and Dingfan Chen for constructive advice in general.

Chapter 8: Conclusion

Deep generative modeling is never a niche topic on its own. In fact, there are many aspects that need our attention. We need to care about the generation quality, i.e., **performance**. We need to care about the control from input to output, i.e., **steerability**. We need to care about the margins between real and fake, i.e., **security**. We also need to care about the minority representation, i.e., **inclusion**. Each of them has some tradeoffs. We not only ask what can the **blessing** of deep generative models do for us. We also ask what can we do for its **curse**. Both questions lead to my Ph.D. research for human-centric deep generative models. It lies on interpreting the behaviors and mitigating the misbehaviors of generative models.

My research works are instantiated but not limited to the following topics: *Contrastive and Attentive GANs* for improved generation performance, *Texture Mixer* for improved texture steerability, a series of *GAN Fingerprinting* solutions for improved deepfake security, as well as *Inclusive GAN* for improved minority inclusion. I propose to examine and improve the human-centric properties of generative models, then project actionable insights to their applications, and finally contribute to human-generator interaction.

When we look back and forward the roadmap of human-generator interaction, we see three clear stages. (1) The first stage is **human-driven generation**. Artists use

brushes or software to describe our world and imagination with pixels. They provide realistic or artistic representation ground truth for generators to mimic. (2) The second stage is **super-human generation**. With unlimited computation power, generators can automatically recreate visual world with their own imagination. Sometimes they can assist artists for more efficient production. In many other cases they can even outperform what humans can make or go beyond what humans can think of. (3) Guess what will the third stage be? Generators are going to conquer and dominate human beings? I hope not. I hope the third stage would be **human-centric generation**. I would have a long-term passion for calibrating deep generative models to be human-centric. I hope to turn generators from enemies to partners and peers, so as to augment humans' life through auto-generation. I also hope to synergize machine intelligence with human intelligence. It is acknowledged that human is good at high-level reasoning while machine is good at computation. I am looking for the best of the two worlds and build more beneficial applications with human-generator interaction.

Bibliography

[1] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.

[2] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *ICCV*, 2019.

[3] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial gan fingerprints: Rooting deepfake attribution in training data. *arXiv*, 2020.

[4] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

[5] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *CVPR*, 2018.

[6] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[7] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

[8] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[11] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, 2016.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017.

[13] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[14] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2018.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.

[16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[17] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[20] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv*, 2015.

[21] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[22] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017.

[23] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019.

[24] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *CVPR*, 2020.

[25] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[26] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *ICLR*, 2020.

[27] Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved consistency regularization for gans. *arXiv*, 2020.

[28] Ning Yu, Ke Li, Peng Zhou, Jitendra Malik, Larry Davis, and Mario Fritz. Inclusive gan: Improving data and minority coverage in generative models. In *ECCV*, 2020.

[29] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[31] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017.

[32] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[33] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[34] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[35] Aysegul Dundar, Karan Sapra, Guilin Liu, Andrew Tao, and Bryan Catanzaro. Panoptic-based image synthesis. In *CVPR*, 2020.

[36] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*, 2020.

[37] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[38] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. In *ICLR*, 2017.

[39] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *CVPR*, 2018.

[40] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *ICCV*, 2019.

[41] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018.

[42] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. In *ECCV*, 2020.

[43] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[44] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[45] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020.

[46] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. In *CVPR*, 2020.

[47] Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *arXiv*, 2020.

[48] Guilin Liu, Rohan Taori, Ting-Chun Wang, Zhiding Yu, Shiqiu Liu, Fitsum A Reda, Karan Sapra, Andrew Tao, and Bryan Catanzaro. Transposer: Universal texture synthesis using feature maps as transposed convolution filter. *arXiv*, 2020.

[49] Morteza Mardani, Guilin Liu, Aysegul Dundar, Shiqiu Liu, Andrew Tao, and Bryan Catanzaro. Neural ffts for universal texture image synthesis. In *NeurIPS*, 2020.

[50] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ToG*, 2017.

[51] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018.

[52] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019.

[53] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[54] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[55] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *PAMI*, 2018.

[56] Fuwen Tan, Song Feng, and Vicente Ordonez. Text2scene: Generating compositional scenes from textual descriptions. In *CVPR*, 2019.

[57] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. Fairgan: Fairness-aware generative adversarial networks. In *Big Data*, 2018.

[58] Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. In *NeurIPS*, 2018.

[59] Prasanna Sattigeri, Samuel C Hoffman, Vijil Chenthamarakshan, and Kush R Varshney. Fairness gan: Generating datasets with fairness properties using a generative adversarial network. 2019.

[60] Aditya Grover, Kristy Choi, Rui Shu, and Stefano Ermon. Fair generative modeling via weak supervision. *arXiv*, 2019.

[61] Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In *NeurIPS*, 2019.

[62] Hee Jung Ryu, Hartwig Adam, and Margaret Mitchell. Inclusivefacenet: Improving face attribute detection with race and gender diversity. 2018.

[63] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NeurIPS*, 2017.

[64] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv*, 2018.

[65] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*.

[66] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017.

[67] Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. In *ICLR*, 2019.

[68] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *ICLR*, 2019.

[69] Minguk Kang and Jaesik Park. Contragan: Contrastive learning for conditional image generation. In *NeurIPS*, 2020.

[70] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for gan training. In *arXiv*, 2020.

[71] Jongheon Jeong and Jinwoo Shin. Training GANs with stronger augmentations via contrastive discriminator. In *ICLR*, 2021.

[72] Ning Yu, Connelly Barnes, Eli Shechtman, Sohrab Amirghodsi, and Michal Lukác. Texture mixer: A network for controllable synthesis and interpolation of texture. In *CVPR*, 2019.

[73] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *ICCV*, 2019.

[74] Ricard Durall, Margret Keuper, and Janis Keuper. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *CVPR*, 2020.

[75] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 2019.

[76] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Coco-gan: generation by parts via conditional coordinating. In *ICCV*, 2019.

[77] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv*, 2018.

[78] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[79] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, 2016.

[80] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. In *arXiv*, 2020.

[81] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[82] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 1993.

[83] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[84] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[85] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015.

[86] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *NeurIPS*, 2019.

[87] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[88] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv*, 2019.

[89] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020.

[90] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[91] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[92] Yixiao Ge, Dapeng Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. In *NeurIPS*, 2020.

[93] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[94] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *ICLR*, 2019.

[95] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019.

[96] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[97] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

[98] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *CVPR*, 2017.

[99] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *CVPR*, 2017.

[100] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *NeurIPS*, 2018.

[101] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[102] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018.

[103] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.

[104] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[105] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[106] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *CVPR*, 2016.

[107] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *CVPR*, 2017.

[108] Lu Chi, Zehuan Yuan, Yadong Mu, and Changhu Wang. Non-local neural networks with grouped bilinear attentional transforms. In *CVPR*, 2020.

[109] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018.

[110] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[111] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

[112] Hao Tang, Dan Xu, Yan Yan, Philip HS Torr, and Nicu Sebe. Local class-specific and global image-level generative adversarial networks for semantic-guided scene generation. In *CVPR*, 2020.

[113] Bowen Li, Xiaojuan Qi, Philip Torr, and Thomas Lukasiewicz. Lightweight generative adversarial networks for text-guided image manipulation. In *NeurIPS*, 2020.

[114] Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. Convolutional generation of textured 3d meshes. In *NeurIPS*, 2020.

[115] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

[116] Jae Hyun Lim and Jong Chul Ye. Geometric gan. In *arXiv*, 2017.

[117] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *ICCV*, 2019.

[118] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[119] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[120] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001.

[121] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *TOG*, 2009.

[122] David Heeger and James Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, 1995.

[123] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 2000.

[124] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015.

[125] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[126] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[127] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, 2016.

[128] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017.

[129] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *TOG*, 2012.

[130] Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. *TOG*, 2015.

[131] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[132] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2018.

[133] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NeurIPS*, 2017.

[134] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017.

[135] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[136] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv*, 2017.

[137] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *TOG*, 2017.

[138] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, 2000.

[139] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *TOG*, 2003.

[140] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *TOG*, 2005.

[141] Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Texture design using a simplicial complex of morphable textures. In *TOG*, 2005.

[142] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. In *TOG*, 2006.

[143] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *TPAMI*, 2007.

[144] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, 2015.

[145] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[146] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[147] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv*, 2016.

[148] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *TOG*, 2018.

[149] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. In *ICML*, 2017.

[150] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *CVPR*, 2017.

[151] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. 2019.

[152] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *TOG*, 2003.

[153] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[154] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *ICLR*, 2018.

[155] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[156] Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. State of the art on monocular 3d face reconstruction, tracking, and applications. In *Computer Graphics Forum*, 2018.

[157] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

[158] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[159] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *TOG*, 2015.

[160] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[161] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *TOG*, 2017.

[162] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Visualizing and understanding generative adversarial networks. In *ICLR*, 2019.

[163] You thought fake news was bad? deep fakes are where truth goes to die. https://www.theguardian.com/technology/2018/nov/12/deep-fakes-fake-news-truth.

[164] Deep fakes: How they are made and how they can be detected. https://www.nbcnews.com/mach/video/deep-fakes-how-they-are-made-and-how-they-can-be-detected-1354417219989.

[165] In the age of a.i., is seeing still believing? https://www.newyorker.com/magazine/2018/11/12/in-the-age-of-ai-is-seeing-still-believing.

[166] Paolo Bestagini, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Local tampering detection in video sequences. In *MMSP*, 2013.

[167] Husrev Taha Sencar and Nasir Memon. Digital image forensics. *Counter-Forensics: Attacking Image Forensics*, 2013.

[168] Hany Farid. *Photo forensics*. MIT Press, 2016.

[169] Belhassen Bayar and Matthew C Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, 2016.

[170] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, 2017.

[171] Jawadul H Bappy, Amit K Roy-Chowdhury, Jason Bunk, Lakshmanan Nataraj, and BS Manjunath. Exploiting spatial structure for localizing manipulated image regions. In *ICCV*, 2017.

[172] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 2018.

[173] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Two-stream neural networks for tampered face detection. In *CVPR Workshop*, 2017.

[174] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018.

[175] Luca D'Amiano, Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. A patchmatch-based dense-field algorithm for video copy–move detection and localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[176] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of gan-generated fake images over social networks. In *MIPR*, 2018.

[177] Huaxiao Mo, Bolin Chen, and Weiqi Luo. Fake faces identification via convolutional neural network. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, 2018.

[178] Shahroz Tariq, Sangyup Lee, Hoyoung Kim, Youjin Shin, and Simon S Woo. Detecting both machine and human created fake face images in the wild. In *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, 2018.

[179] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[180] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 2016.

[181] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Multimedia*, 2014.

[182] Model gallery. https://www.microsoft.com/en-us/cognitive-toolkit/features/model-gallery.

[183] The value of stolen data on the dark web. https://darkwebnews.com/dark-web/value-of-stolen-data-dark-web.

[184] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Asia CCS*, 2018.

[185] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ICMR*, 2017.

[186] Jan Lukas, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *TIFS*, 2006.

[187] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukás. Determining image origin and integrity using sensor noise. *TIFS*, 2008.

[188] Davide Cozzolino and Luisa Verdoliva. Noiseprint: a cnn-based camera model fingerprint. *TIFS*, 2020.

[189] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? In *MIPR*, 2019.

[190] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *ICML*, 2017.

[191] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. In *ICIP*, 2017.

[192] Marc G Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv*, 2017.

[193] Jessica Fridrich. *Digital image forensics: there is more to a picture than meets the eye*. Springer New York, 2012.

[194] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *TIFS*, 2012.

[195] Luca Bondi, Silvia Lameri, David Guera, Paolo Bestagini, Edward J Delp, Stefano Tubaro, et al. Tampering detection and localization through clustering of camera-based cnn features. In *CVPR Workshop*, 2017.

[196] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *WIFS*, 2018.

[197] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv*, 2018.

[198] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv*, 2018.

[199] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. *arXiv preprint arXiv:1901.08971*, 2019.

[200] Mitchell D Swanson, Mei Kobayashi, and Ahmed H Tewfik. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE*, 1998.

[201] Gerhard C Langelaar, Iwan Setyawan, and Reginald L Lagendijk. Watermarking digital image and video data. a state-of-the-art overview. *IEEE Signal Processing Magazine*, 2000.

[202] Lalit Kumar Saini and Vishal Shrivastava. A survey of digital watermarking techniques and its applications. *IJEIT*, 2014.

[203] Efstathios Stamatatos. A survey of modern authorship attribution methods. *AIST*, 2009.

[204] Sadia Afroz, Aylin Caliskan Islam, Ariel Stolerman, Rachel Greenstadt, and Damon McCoy. Doppelgänger finder: Taking stylometry to the underground. In *S&P*, 2014.

[205] Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *Transactions on Communications*, 1983.

[206] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[207] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv*, 2015.

[208] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Optical Society of America*, 1987.

[209] DC Dowson and BV Landau. The fréchet distance between multivariate normal distributions. *JMA*, 1982.

[210] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[211] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D Castillo, and David W Jacobs. Sfsnet: Learning shape, reflectance and illuminance of facesin the wild'. In *CVPR*, 2018.

[212] How to recognize fake ai-generated images. https://medium.com/@kcimc/how-to-recognize-fake-ai-generated-images-4d1f6f9a2842.

[213] Charlotte Jee. An indian politician is using deepfake technology to win new voters. 2020.

[214] James Vincent. An online propaganda campaign used ai-generated headshots to create fake journalists. 2020.

[215] Dan Robitzski. Someone used deepfake tech to invent a fake journalist. 2020.

[216] Baiwu Zhang, Jin Peng Zhou, Ilia Shumailov, and Nicolas Papernot. Not my deepfake: Towards plausible deniability for machine-generated media. *arXiv*, 2020.

[217] Ricard Durall, Margret Keuper, Franz-Josef Pfreundt, and Janis Keuper. Unmasking deepfakes with simple features. *arXiv*, 2019.

[218] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *WIFS*, 2019.

[219] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *ICML*, 2020.

[220] Ricard Durall, Margret Keuper, and Janis Keuper. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *CVPR*, 2020.

[221] Nicholas Carlini and Hany Farid. Evading deepfake-image detectors with white- and black-box attacks. In *CVPR Workshops*, 2020.

[222] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *NeurIPS*, 2017.

[223] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *CVPR*, 2020.

[224] outguess, http://www.outguess.org/.

[225] steghide, http://steghide.sourceforge.net.

[226] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*, 2020.

[227] Jessica Fridrich. *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press, 2009.

[228] Ingemar Cox, Matthew Miller, Jeffrey Bloom, and Chris Honsinger. *Digital watermarking*. Springer, 2002.

[229] Francois Cayre, Caroline Fontaine, and Teddy Furon. Watermarking security: theory and practice. In *TSP*, 2005.

[230] Tomáš Pevnỳ, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In *IWIH*, 2010.

[231] Vojtěch Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *WIFS*, 2012.

[232] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. In *EURASIP JIS*, 2014.

[233] Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. In *NeurIPS*, 2017.

[234] Vedran Vukotić, Vivien Chappelier, and Teddy Furon. Are deep neural networks good for blind image watermarking? In *WIFS*, 2018.

[235] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, 2018.

[236] Ru Zhang, Shiqi Dong, and Jianyi Liu. Invisible steganography via generative adversarial networks. In *Multimedia Tools and Applications*, 2019.

[237] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. Distortion agnostic deep watermarking. In *CVPR*, 2020.

[238] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX*, 2018.

[239] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *ICMR*, 2019.

[240] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: an end-to-end watermarking framework for protecting the ownership of deep neural networks. In *ASPLOS*, 2019.

[241] Zhengzhe Liu, Xiaojuan Qi, Jiaya Jia, and Philip Torr. Global texture enhancement for fake face detection in the wild. In *CoRR*, 2020.

[242] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. *arXiv*, 2020.

[243] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In *arXiv*, 2019.

[244] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *arXiv*, 2020.

[245] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[246] Daniel Lerch-Hostalot and David Megías. Unsupervised steganalysis based on artificial training sets. In *EAAI*, 2016.

[247] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. In *TIST*, 2011.

[248] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[249] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. 2016.

[250] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[251] Xiaojuan Qi, Qifeng Chen, Jiaya Jia, and Vladlen Koltun. Semi-parametric image synthesis. In *CVPR*, 2018.

[252] Douglas Harris. Deepfakes: False pornography is here and the law cannot protect you. *Duke L. & Tech. Rev.*, 2018.

[253] Robert Chesney and Danielle Citron. Deepfakes and the new disinformation war: The coming age of post-truth geopolitics. *Foreign Aff.*, 2019.

[254] Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, et al. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv*, 2018.

[255] Chih-Chung Hsu, Chia-Yen Lee, and Yi-Xiu Zhuang. Learning to detect fake face images in the wild. In *IS3C*, 2018.

[256] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *AVSS*, 2018.

[257] Zihan Wang, Neng Gao, Xin Wang, Xuexin Qu, and Linghui Li. Sstegan: self-learning steganography based on generative adversarial networks. In *ICONIP*, 2018.

[258] Zhuo Zhang, Jia Liu, Yan Ke, Yu Lei, Jun Li, Minqing Zhang, and Xiaoyuan Yang. Generative steganography by sampling. *IEEE Access*, 2019.

[259] Denis Volkhonskiy, Ivan Nazarov, and Evgeny Burnaev. Steganographic generative adversarial networks. In *ICMV*, 2019.

[260] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *CVPR*, 2020.

[261] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[262] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *ICML*, 2014.

[263] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[264] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2016.

[265] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2016.

[266] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[267] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

[268] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv*, 2017.

[269] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. "best-of-many-samples" distribution matching. *arXiv*, 2019.

[270] Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *SIGNLL*, 2016.

[271] Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. In *ICML*, 2018.

[272] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints. In *ICDM Workshops*, 2009.

[273] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012.

[274] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *KDD*, 2015.

[275] Lu Zhang, Yongkai Wu, and Xintao Wu. A causal framework for discovering and removing direct and indirect discrimination. In *IJCAI*, 2017.

[276] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *ICDM Workshops*, 2011.

[277] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *AISTATS*, 2017.

[278] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *EMNLP*, 2017.

[279] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. Discrimination aware decision tree learning. In *ICDM*, 2010.

[280] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *NeurIPS*, 2016.

[281] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. 2019.

[282] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.

[283] Jonas Adler and Sebastian Lunz. Banach wasserstein gan. In *NeurIPS*, 2018.

[284] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016.

[285] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. 2017.

[286] Ngoc-Trung Tran, Tuan-Anh Bui, and Ngai-Man Cheung. Dist-gan: An improved gan using distance constraints. In *ECCV*, 2018.

[287] Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. In *ICLR*, 2019.

[288] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *NeurIPS*, 2018.

[289] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *ICLR*, 2017.

[290] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *ICLR*, 2017.

[291] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. 2019.

[292] Liqun Chen, Shuyang Dai, Yunchen Pu, Erjin Zhou, Chunyuan Li, Qinliang Su, Changyou Chen, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. In *AISTATS*, 2018.

[293] Ke Li and Jitendra Malik. Fast k-nearest neighbour search via prioritized dci. In *ICML*, 2017.

[294] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[295] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[296] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *NeurIPS*, 2018.