UNIVERSITY OF CALIFORNIA SAN DIEGO

Learning Generative Models with Energy-Based Models and Transformer GANs

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Kwonjoon Lee

Committee in charge:

Professor Zhuowen Tu, Chair
Professor Lawrence Saul, Co-Chair
Professor Virginia De Sa
Professor Ravi Ramamoorthi
Professor Hao Su

2022

University of California San Diego

2022

DEDICATION

To my family.

EPIGRAPH

What I cannot create, I do not understand.

—Richard Feynman

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Zhuowen Tu, for his support and guidance throughout my graduate studies. He taught me not only necessary research skills but also mindsets required for being a good researcher and a person. He always encouraged me to do impactful research. As a student who had no research experience before meeting him, I couldn't imagine I could achieve it.

I would like also thank my thesis committee members, Professors Lawrence Saul, Ravi Ramamoorthi, Hao Su, and Virginia De Sa, for their constructive and insightful comments on the thesis. In addition, I have benefited significantly from their courses and research papers. I appreciate Lawrence Saul for serving as the co-chair of the committee and teaching me the first course on artificial intelligence; Ravi Ramamoorthi for his research on neural radiance field which had a substantial impact on my ViTGAN work; Hao Su for his course on 3D computer vision; Virginia De Sa for her research on self-supervised learning which had a significant influence on my research.

I appreciate friends at Machine Learning, Perception, and Cognition Lab (mlPC), Weijian Xu, Yifan Xu, Justin Lazarow, Kunyu Shi, Gaurav Parmar, Dacheng Li, Fan Fan, Jeng-Hau Lin, and Siyang Wang for their help in research and everyday life.

I am grateful for three summers I spent at industrial research labs. I thank Subhransu Maji, Avinash Ravichandran, and Stefano Soatto at Amazon Web Services for encouraging me to start research in meta-learning which was entirely new to me. I appreciate Shubham Tulsiani and Abhinav Gupta at Facebook AI Research for giving me an opportunity to do research in meta reinforcement learning. I thank Huiwen Chang, Lu Jiang, Han Zhang, Deqing Sun, Charles Herrmann and Ce Liu at Google Research for a year-long collaboration and mentorship that led to the ViTGAN work. These individuals are one of the greatest minds in the field and helped me develop all aspects of research process.

Finally, I thank my parents for always helping me out. Without their support, I couldn't come to the US for graduate studies. I also thank Lin for her love and being supportive of my dream.

This work is supported by NSF IIS-1717431, NSF IIS-1618477, and Google Cloud Platform (GCP) Credit Award.

Chapter 2, in full, is a reprint of the material as it appears in “Wasserstein Introspective Neural Networks,” Kwonjoon Lee, Weijian Xu, Fan Fan, Zhuowen Tu, Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2018. The dissertation author is the primary investigator and author of this material.

Chapter 3, in part, is a reprint of the material as it appears in “ViTGAN: Training GANs with Vision Transformers,” Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, Ce Liu, International Conference on Learning Representations (ICLR), 2022. The dissertation author is the primary investigator and author of this material.

Chapter 4, in full, is a reprint of the material as it appears in “Meta-Learning with Differentiable Convex Optimization,” Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, Stefano Soatto, Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2019. The dissertation author is the primary investigator and author of this material.

VITA

2016	B.S. in Electrical and Computer Engineering *summa cum laude*, Seoul National University, Seoul, Korea
2018	M.S. in Computer Science, University of California San Diego
2022	Ph.D. in Computer Science, University of California San Diego

PUBLICATIONS

Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu, “ViTGAN: Training GANs with Vision Transformers”, To appear in *International Conference on Learning Representations (ICLR)*, 2022.

Gaurav Parmar*, Dacheng Li*, **Kwonjoon Lee***, and Zhuowen Tu, “Dual Contradistinctive Generative Autoencoder”, In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Justin Lazarow*, **Kwonjoon Lee***, Kunyu Shi*, and Zhuowen Tu, “Learning Instance Occlusion for Panoptic Segmentation”, In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto, “Meta-Learning with Differentiable Convex Optimization”, In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Kwonjoon Lee, Weijian Xu, Fan Fan, and Zhuowen Tu, “Wasserstein Introspective Neural Networks”, In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

ABSTRACT OF THE DISSERTATION

Learning Generative Models with Energy-Based Models and Transformer GANs

by

Kwonjoon Lee

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Zhuowen Tu, Chair
Professor Lawrence Saul, Co-Chair

In this thesis, we study approaches to learn priors on data (i.e. generative modeling) and learners (i.e. meta-learning) for computer vision tasks. We present our approaches to improve the stability and performance of generative modeling and meta-learning methods.

First, we study the use of natural image prior on computer vision tasks. To this end, we introduce a suite of regularization techniques that enhances the performance of energy-based models on realistic image datasets. On generative modeling, we achieve competitive results while using much smaller models. On supervised classification, we observe a significant error reduction against adversarial examples. Our model is the first computer vision model to achieve state-of-the-art image generation and classification within a single model.

Next, we investigate if natural image prior can be learned with less vision-specific inductive biases. To this end, we integrate the Vision Transformer architecture into generative adversarial networks (GAN). We propose novel regularization methods and architectural choices to achieve this goal. The resulting approach, named ViTGAN, achieves comparable performance to the leading CNN-based GAN models on popular image generation benchmarks for the first time.

Lastly, we study a meta-learning approach, which automatically extracts prior knowledge from the set of observed tasks. We present our work on improving the computational complexity of meta-learning. Our approach, named MetaOptNet, offers better few-shot generalization at a modest increase in computational overhead.

Chapter 1

Introduction

In the first part the dissertation, I present approaches and architectures for learning natural image priors. I also show how they can benefit synthesis and classification of images. Then, in the second part, I present a meta-learning approach — which which automatically extracts priors on learners from the set of observed tasks — for few-shot visual learning.

First, we study learning image priors through a lens of energy-based models. Energy-based model (EBM) [HSA84, HS86] is a principled approach for modeling data densities. It has been a long-standing research problem, but its applicability was limited to simple datasets due to difficulties in training and sampling. Introspective Neural Networks (INN) [JLT17] greatly enhanced EBM's modeling capability by adopting modern recipes such as convolutional neural networks and Langevin dynamics. Its application was limited to simple datasets such as MNIST. In Chapter 2, we present Wasserstein introspective neural networks (WINN) that improve training stability of INN [JLT17]. This was achieved by the adoption of gradient penalty, along with several modifications to architecture and sampling strategy. On image generative modeling benchmarks, we show that improved stability leads to greatly enhanced modeling capability of EBM — rivaling state-of-the-art generative adversarial networks [RMC16a]. When applied to supervised classification, WINN also gives rise to improved robustness against adversarial examples in terms of the error reduction. To the best of our knowledge, our approach is the first to achieve competitive results on both generative and discriminative learning benchmarks including texture, face, and object datasets.

State-of-the-art generative modeling methods for various modalities such as video are equipped with architectures [DF18, TLYK18, CDS19] that are vastly different from the ones for image generative modeling [ZGMO19, KLA$^+$20, BDS19]. Still, generative models learned by such methods are not very faithful to training data. Furthermore, it is unsatisfactory if we need to design a different architecture for every single generative modeling problem. Hence, it would be a scientific dream if we could develop generative modeling architectures that do not rely on modality-specific architectural priors. Recent work on Vision Transformers (ViT) [DBK$^+$21] shows a promising approach towards this goal. Notably, [DBK$^+$21, ADH$^+$21] showed that vanilla Transformer [VSP$^+$17] can be used for image or video by turning it into a sequence of non-overlapping patches. Hence, in Chapter 3, we investigate if such observations can be extended to generative modeling of images. To this end, we integrate the ViT architecture into generative adversarial networks (GANs) [GPAM$^+$14a]. For ViT discriminators, we observe that existing regularization methods for GANs interact poorly with self-attention, causing serious instability during training. To resolve this issue, we introduce several novel regularization techniques for training GANs with ViTs. For ViT generators, we examine architectural choices for latent and pixel mapping layers to facilitate convergence. Empirically, our approach, named *ViTGAN*, achieves comparable performance to the leading CNN-based GAN models on three datasets: CIFAR-10, CelebA, and LSUN bedroom.

Many meta-learning approaches for few-shot learning rely on simple base learners such as nearest-neighbor classifiers. However, even in the few-shot regime, discriminatively trained linear predictors can offer better generalization. Hence, in Chapter 4, we propose to use these predictors as base learners to learn representations for few-shot learning and show they offer better tradeoffs between feature size and performance across a range of few-shot recognition benchmarks. Our

objective is to learn feature embeddings that generalize well under a linear classification rule for novel categories. To efficiently solve the objective, we exploit two properties of linear classifiers: implicit differentiation of the optimality conditions of the convex problem and the dual formulation of the optimization problem. This allows us to use high-dimensional embeddings with improved generalization at a modest increase in computational overhead. Our approach, named MetaOptNet, achieves state-of-the-art performance on miniImageNet, tieredImageNet, CIFAR-FS, and FC100 few-shot learning benchmarks.

Chapter 2

Wasserstein Introspective Neural Networks

2.1 Introduction

Performance within the task of supervised image classification has been vastly improved in the era of deep learning using modern convolutional neural network (CNN) [LBD$^+$89b] based discriminative classifiers [KSH12, SLJ$^+$15, LXG$^+$15, SZ15, HZRS16, XGD$^+$17, HLvdMW17]. On the other hand, unsupervised generative models in deep learning were previously attained using methods under the umbrella of graphical models — e.g., the Boltzmann machine [HOT06] or autoencoder [Bal12, KW14] architectures. However, the rich representational power seen within convolution-based (discriminative) models is not being directly enjoyed in these generative models. Later, inverting convolutional neural networks in order to convert internal representations into a real image was investigated in [DSB15, KWKT15]. Recently, generative adversarial networks (GAN) [GPAM$^+$14b] and followup works [RMC16b, ACB17, GAA$^+$17] have attracted a tremendous amount of attention in machine learning and computer vision by producing high quality synthesized images by training a pair of competing models against one another in an adversarial manner. While a generator tries to create "fake" images to fool the discriminator, the discriminator attempts to discern between these "real" (given training) and "fake" images. After convergence, the generator is able to produce images faithful to the underlying data distribution.

In terms of building generative models from discriminative classifiers, there have been early attempts in [WZH02, Tu07]. In [WZH02], a generative model was obtained from a repeatedly trained boosting algorithm [FS97] using a weak classifier whereas [Tu07] used a strong classifier in order to self-generate negative examples or “pseudo-negatives”.

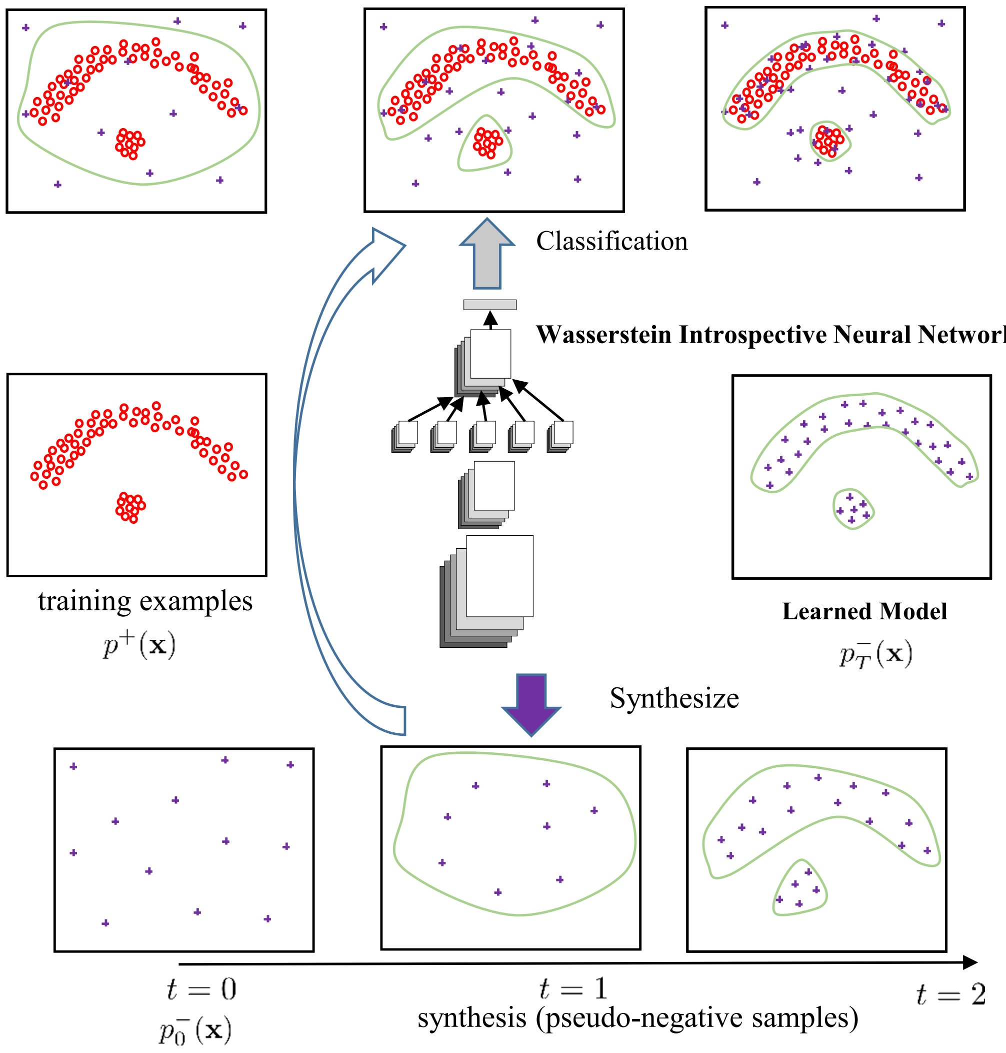


Figure 2.1: **Schematic illustration of Wasserstein introspective neural networks for unsupervised learning.** The left figure shows the input examples; the bottom figures show the pseudo-negatives (purple crosses) being progressively synthesized; the top figures show the classification between the given examples (positives) and synthesized pseudo-negatives (negatives). The right figure shows the model learned to approach the target distribution based on the given data.

To address the lack of richness in representation and efficiency in synthesis, convolutional neural networks were adopted in introspective neural networks (INN) [LJT17, JLT17] to build a single model that is simultaneously generative and discriminative. The generative modeling aspect was studied in [LJT17] where a sequence of CNN classifiers ($10-60$) were trained, while the power within the classification setting was revealed in [JLT17] in the form of introspective convolutional networks (ICN) that used only a single CNN classifier. Although INN models [LJT17, JLT17] point to a promising direction to obtain a single model being both a good generator and a strong discriminative classifier, a sequence of CNNs were needed to generate realistic synthesis. As a result, this requirement may serve as a possible bottleneck with respect to training complexity and model size.

Recently, a generic formulation [ACB17] was developed within the GAN model family to incorporate a Wasserstein objective to alleviate the well-known difficulty in GAN training. Motivated by introspective neural networks (INN) [LJT17] and this Wasserstein objective [ACB17], we propose to adopt the Wasserstein term into the INN formulation to enhance the modeling capability. The resulting model, Wasserstein introspective neural networks (WINN) shows greatly enhanced modeling capability over INN by having $20\times$ reduction in the number of CNN classifiers.

2.2 Significance and Related Work

We make several interesting observations for WINN:

- A mathematical connection between the WGAN formulation [ACB17] and the INN algorithm [LJT17] is made to better understand the overall objective function within INN.
- By adopting the Wasserstein distance into INN, we are able to generate images using a single CNN in WINN with even higher quality than those by INN that uses 20 CNNs (as seen in Figure 2.2, 2.4, 2.5, 2.6, and 2.7; the similar underlying CNN architectures are used in WINN and INN). WINN

achieves a significant reduction in model complexity over INN, making the generator more practical.

- Within texture modeling, INN and WINN are able to inherently model the input image space, making the synthesis of large texture images realistic, whereas GAN projects a noise vector onto the image space making the image patch stitching more difficult (although extensions exist), as demonstrated in Figure 2.2.
- To compare with the family of GAN models, we compute Inception scores using the standard procedure on the CIFAR-10 datasets and observed modest results. Here, we typically train 4-5 cascades to boost the numbers but WINN with one CNN is already promising. Overall, modern GAN variants (e.g., [GAA+17]) still outperform our WINN with better quality images. Some results are shown in Figure 2.7.
- To test the robustness of the discriminative abilities of WINN, we directly make WINN into a discriminative classifier by training it on the standard MNIST and SVHN datasets. Not only are we able to improve over the previous ICN [JLT17] classifier for supervised classification, we also observe a large improvement in robustness against adversarial examples compared with the baseline CNN, ResNet, and the competing ICN.

In terms of other related work, we briefly discuss some existing methods below.

Wasserstein GAN. A closely related work to our WINN algorithm is the Wasserstein generative adversarial networks (WGAN) method [ACB17, GAA+17]. While WINN adopts the Wasserstein distance as motivated by WGAN, our overall algorithm is still within the family of introspective neural networks (INN) [LJT17, JLT17]. WGAN on the other hand is a variant of GAN with an improvement over GAN by having an objective that is easier to train. The level of difference between WINN and WGAN is similar to that between INN [LJT17, JLT17] and GAN [GPAM+14b]. The overall comparisons between INN and GAN have been described in [LJT17, JLT17].

Generative ConvNets. Recently, there has also been a cluster of algorithms developed in [XLG+18, XLZW16, HLZW17] where Langevin dynamics are adopted in generator CNNs.

However, the models proposed in [XLG$^+$18, XLZW16, HLZW17] do not perform introspection (Figure 2.1) and their generator and discriminator components are still somewhat separated; thus, their generators are not used as effective discriminative classifiers to perform state-of-the-art classification on standard supervised machine learning tasks. Their training processes are also more complex than those of INN and WINN.

Deep energy models (DEMs) [NCKN11]. DEM [NCKN11] extends the standard density estimation by using multi-layer neural networks (MLNN) with a rather complex training procedure. The probability model in DEM includes both the raw input and the features computed by MLNN. WINN instead takes a more general and simplistic form and is easier to train (see Eq. (2.1)). In general, DEM belongs to the minimum description length (MDL) family models in which the maximum likelihood is achieved. WINN, instead, has a formulation being simultaneously discriminative and generative.

2.3 Introspective Neural Networks

2.3.1 Brief introduction of INN

We first briefly introduce the introspective neural network method (INNg) [LJT17] for generative modeling and its companion model [JLT17] which focuses on the classification aspect. The main motivation behind the INN work [LJT17, JLT17] is to make a convolutional neural network classifier simultaneously discriminative and generative. A single CNN classifier is trained in an introspective manner to improve the standard supervised classification result [JLT17], however, a sequence of CNNs (typically $10-60$) is needed to be able to synthesize images of good quality [LJT17].

Figure 2.1 shows a brief illustration of a single introspective CNN classifier [JLT17]. We discuss our basic unsupervised formulation next. Suppose we are given a set of training examples: $S = \{\mathbf{x}_i \mid i = 1, \ldots, n\}$ where we assume each $x_i \in \mathbb{R}^m$ — e.g., $m = 4096$ for images of size 64×64.

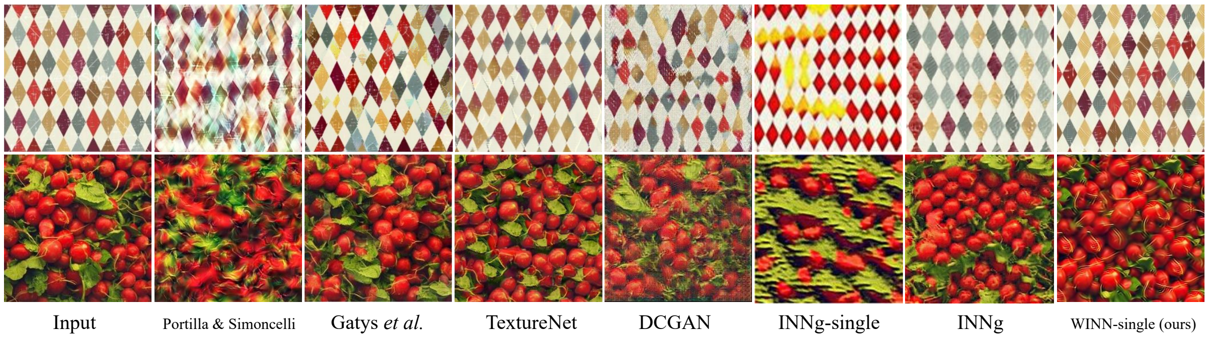


Figure 2.2: **Comparison of texture synthesis algorithms.** Gatys *et al.* [GEB15a], and TextureNet [ULVL16] results are from [ULVL16].

These will constitute positive examples of the patterns/targets we wish to model. The main idea of INN is to define pseudo-negative examples that are to be self-generated by the discriminative classifier itself. We therefore define label y for each example $\mathbf{x}$, $y = +1$ if $\mathbf{x}$ is from the given training set and $y = -1$ if $\mathbf{x}$ is self-generated. Motivated by the generative via discriminative learning (GDL) framework [Tu07], one could try to learn the generative model for the given examples, $p(\mathbf{x}|y = +1)$, by a sequentially learned distribution of the **pseudo-negative** samples, $p_t(\mathbf{x}|y = -1; \mathsf{W}_t)$ which is abbreviated as $p^-_{\mathsf{W}_t}(\mathbf{x})$ where W_t includes all the model parameters learned at step t.

$$p^-_{\mathsf{W}_t}(\mathbf{x}) = \frac{1}{Z_t}\exp\{\mathbf{w}_t^{(1)} \cdot \phi(\mathbf{x}; \mathbf{w}_t^{(0)})\} \cdot p^-_0(\mathbf{x}),\ t = 1, \ldots, T \tag{2.1}$$

where $Z_t = \int \exp\{\mathbf{w}_t^{(1)} \cdot \phi(\mathbf{x}; \mathbf{w}_t^{(0)})\} \cdot p^-_0(\mathbf{x}) d\mathbf{x}$ and the initial distribution $p^-_0(\mathbf{x})$ such as a Gaussian distribution over the entire space of $\mathbf{x} \in \mathbb{R}^m$. The discriminative classifier is a convolutional neural network (CNN) parameterized by $\mathsf{W}_t = (\mathbf{w}_t^{(0)}, \mathbf{w}_t^{(1)})$ where $\mathbf{w}_t^{(1)}$ denotes the weights of the top layer combining the features through $\phi(\mathbf{x}; \mathbf{w}_t^{(0)})$ (e.g., softmax layer) and $\mathbf{w}_t^{(0)}$ parameterizing the internal representations. The synthesis process through which pseudo-negative samples are generated is carried out by stochastic gradient Langevin dynamics [WT11] as

$$\Delta\mathbf{x} = \frac{\varepsilon}{2}\nabla(\mathbf{w}_t^{(1)} \cdot \phi(\mathbf{x}; \mathbf{w}_t^{(0)})) + \eta$$

where $\eta \sim \mathcal{N}(0, \varepsilon)$ is a Gaussian distribution and ε is the step size that is annealed in the sampling process. Overall, we desire

$$p^-_{\mathsf{W}_t}(\mathbf{x}) \overset{t=\infty}{\rightarrow} p(\mathbf{x}|y=+1), \tag{2.2}$$

using the iterative reclassification-by-synthesis process [JLT17, LJT17] guided by Eq. (2.1).

2.3.2 Connection to the Wasserstein distance

The overall training process, reclassification-by-synthesis, is carried out iteratively without an explicit objective function. The generative adversarial network (GAN) model [GPAM$^+$14b] instead has an objective function formulated in a minimax fashion with the generator and discriminator competing against each other. The Wasserstein generative adversarial network (WGAN) work [ACB17] improves GAN [GPAM$^+$14b] by replacing the Jensen-Shannon distance with an efficient approximation of the Earth-Mover distance [ACB17]. Also, there has been further generalization of the GAN family models in [LBC17].

Let $p^+(\mathbf{x}) \equiv p(\mathbf{x}|y=+1)$ be the target distribution and $p^-_{\mathsf{W}}(\mathbf{x}) \equiv p(\mathbf{x}|y=-1;\mathsf{W})$ be the pseudo-negative distribution parameterized by W. Next, we show a connection between the INN framework and the WGAN formulation [ACB17], whose objective (rewritten with our notations) can be defined as

$$\min_{\mathsf{W}} \max_{||f||_L \leq 1} E_{\mathbf{x}\sim p^+}[f(\mathbf{x})] - E_{\mathbf{x}\sim p^-_{\mathsf{W}}}[f(\mathbf{x})], \tag{2.3}$$

where $||f||_L \leq 1$ denotes the space of 1-Lipschitz functions. To build the connection between Eq. (2.3) of WGAN and Eq. (2.1) of INN, we first present the following lemma.

Lemma 1 *Considering $f(\mathbf{x}) = \ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})}$ and assuming its 1-Lipschitz property, we have a lower bound on the Wasserstein distance by*

$$\begin{aligned} & \max_{||f||_L \leq 1} E_{\mathbf{x}\sim p^+}[f(\mathbf{x})] - E_{\mathbf{x}\sim p^-_{\mathsf{W}}}[f(\mathbf{x})] \\ \geq \quad & KL(p^+||p^-_{\mathsf{W}}) + KL(p^-_{\mathsf{W}}||p^+) \end{aligned} \tag{2.4}$$

where $KL(p||q)$ denotes the Kullback-Leibler divergence between the two distributions p and q, and $KL(p^+||p^-_{\mathsf{W}}) + KL(p^-_{\mathsf{W}}||p^+)$ is the Jeffreys divergence.

Proof. Plugging $f(\mathbf{x}) = \ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})}$ into Eq. (2.3), we have

$$\begin{aligned} & E_{\mathbf{x}\sim p^+}[f(\mathbf{x})] - E_{\mathbf{x}\sim p^-_{\mathsf{W}}}[f(\mathbf{x})] \\ = \;& E_{\mathbf{x}\sim p^+}[\ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})}] - E_{\mathbf{x}\sim p^-_{\mathsf{W}}}[\ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})}] \\ = \;& \int p^+(\mathbf{x}) \ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})} d\mathbf{x} - \int p^-_{\mathsf{W}}(\mathbf{x}) \ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})} d\mathbf{x} \\ = \;& KL(p^+||p^-_{\mathsf{W}}) + KL(p^-_{\mathsf{W}}||p^+) \qquad \square \end{aligned}$$

Note that using the Bayes' rule, the ratio of the generative probabilities $\frac{p(\mathbf{x}|y=+1)}{p(\mathbf{x}|y=-1)}$ in Lemma 1 can be turned into the ratio of the discriminative probabilities $\frac{p(y=+1|\mathbf{x})}{p(y=-1|\mathbf{x})}$ assuming equal priors $p(y=+1) = p(y=-1)$.

Corollary 1 *The Jeffreys divergence in Eq. (2.4) of lemma 1 is lower and upper bounded by $KL(p^+||p^-_{\mathsf{W}})$ up to some multiplicative constant*

$$(1 + \frac{p^+_{min}}{2}) KL(p^+||p^-_{\mathsf{W}}) \leq JD(p^+; p^-_{\mathsf{W}}) \leq (1 + \frac{2}{p^+_{min}}) KL(p^+||p^-_{\mathsf{W}}) \tag{2.5}$$

where $JD(p^+; p^-_{\mathsf{W}}) = KL(p^+||p^-_{\mathsf{W}}) + KL(p^-_{\mathsf{W}}||p^+)$ is the Jeffreys divergence.

Proof. Based on the Pinsker's inequality [Pin60, SV16], it is observed that

$$KL(p^-_{\mathsf{W}}||p^+) \geq \frac{1}{2} |p^-_{\mathsf{W}} - p^+|^2 \log e,$$

where $|p^-_{\mathsf{W}} - p^+|$ is total variation (TV) distance. From [SV16] we also have

$$KL(p^-_{\mathsf{W}}||p^+) \leq \frac{\log e}{p^+_{min}} |p^-_{\mathsf{W}} - p^+|^2,$$

where $p^+_{min} = \min_{\mathbf{x}} p^+(\mathbf{x})$. Applying the above bounds to $KL(p^+||p^-_{\mathsf{W}})$ and using the symmetry of the TV distance $|p^-_{\mathsf{W}} - p^+| \equiv |p^+ - p^-_{\mathsf{W}}|$,

$$\frac{p^+_{min}}{2} KL(p^+||p^-_{\mathsf{W}}) \leq KL(p^-_{\mathsf{W}}||p^+) \leq \frac{2}{p^+_{min}} KL(p^+||p^-_{\mathsf{W}}).$$

Plugging the equation above into the Jeffreys divergence, we observe that $KL(p^+||p^-_{\mathsf{W}}) + KL(p^-_{\mathsf{W}}||p^+)$ is upper and lower bounded by by $KL(p^+||p^-_{\mathsf{W}})$. □

Theorem 1 *The introspective neural network formulation (Eq. (2.1)) implicitly minimizes a lower bound of the WGAN objective [ACB17] (Eq. (2.3)).*

Proof. It was shown in [JLT17] that Eq. (2.1) reduces $KL(p^+||p^-_{\mathsf{W}})$, which bounds the Jeffreys divergence $KL(p^+||p^-_{\mathsf{W}}) + KL(p^-_{\mathsf{W}}||p^+)$ as shown in corollary 1. Lemma 1 shows the connection between Jeffreys divergence and the WGAN objective (Eq. (2.3)) when $f(\mathbf{x}) = \ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})}$. We therefore can see that the formulation of introspective neural networks (Eq. (2.1)) connects to a lower bound of the WGAN [ACB17] objective (Eq. (2.3)). □

2.4 Wasserstein Introspective Networks

Here we present the formulation for WINN building upon the formulation of the prior introspective learning works presented in Section 2.3.

2.4.1 WINN algorithm

We denote our unlabeled input training data as $S_+ = \{\mathbf{x}_i | y_i = +1, i = 1, \ldots, n\}$. Also, we denote the set of all the self-generated pseudo-negative samples up to step t as $S^t_- = \{\mathbf{x}_i | y_i = -1, i = 1, \ldots, l\}$. In other words, S^t_- consists of pseudo-negatives $\mathbf{x}$ sampled from our model $p^-_{\mathsf{W}_t}(\mathbf{x})$ for $t \geq 1$ where W_t is the model parameter vector at step t.

Algorithm 1 Outline of WINN-single training algorithm. We use $k = 3$ and $\lambda = 10$.

```
while $\mathsf{W}_t$ has not converged do
    // Classification-step:
    for $k$ steps do
        Sample $m$ positive samples $\{\mathbf{x}_1^+, \cdots, \mathbf{x}_m^+\}$ from $S_+$.
        Sample $m$ pseudo-negative samples $\{\mathbf{x}_1^-, \cdots, \mathbf{x}_m^-\}$ from $S_-^t$.
        Sample $m$ random numbers $\{\alpha_1, \cdots, \alpha_m\}$ from $U[0,1]$.
        $\hat{\mathbf{x}}_i \leftarrow \alpha_i \mathbf{x}_i^+ + (1-\alpha_i)\mathbf{x}_i^-$, $i = 1, \dots, m$.
        Perform stochastic gradient descent:
        $\nabla_{\mathsf{W}_t} \frac{1}{m} \sum_{i=1}^m \{[f_{\mathsf{W}_t}(\mathbf{x}_i^-) - f_{\mathsf{W}_t}(\mathbf{x}_i^+)] + \lambda(\|\nabla_{\hat{\mathbf{x}}_i} f_{\mathsf{W}_t}(\hat{\mathbf{x}}_i)\|_2 - 1)^2\}$.
    end for
    // Synthesis-step:
    Sample $r$ noise samples $\{\mathbf{x}_1, \cdots, \mathbf{x}_r\}$ from $p_0^-(\mathbf{x})$.
    Perform stochastic gradient ascent with early-stopping.
    $S_-^{t+1} \leftarrow S_-^t \cup \{\mathbf{x}_1, \cdots, \mathbf{x}_r\}$.
    $t \leftarrow t + 1$.
end while
```

Classification-step. The classification-step can be viewed as training a classifier to approximate the Wasserstein distance between S_+ and S_-^t for $t \geq 1$. Note that we also keep pseudo-negatives from earlier stages – which are essentially the mistakes of the earlier stages – to prevent the classifier forgetting what it has learned in previous stages. We use CNNs parametrized by W_t as base classifiers. Let $f_{\mathsf{W}_t}(\cdot)$ denote the output of final fully connected layer (without passing through sigmoid nonlinearity) of the CNN. In the previous introspective learning frameworks [LJT17, JLT17], the classifier learning objective was to minimize the following standard cross-entropy loss function on $S_+ \cup S_-^t$:

$$\mathcal{L}(\mathsf{W}_t) = -\{\mathbb{E}_{\mathbf{x}^+ \sim p^+} \ln \sigma[+f_{\mathsf{W}_t}(\mathbf{x}^+)] + \mathbb{E}_{\mathbf{x}^- \sim p^-_{\mathsf{W}_t}} \ln \sigma[-f_{\mathsf{W}_t}(\mathbf{x}^-)]\}$$

where $\sigma(\cdot)$ denotes the sigmoid nonlinearity. Motivated by Section 2.3.2, in WINN training we wish to minimize the following Wasserstein loss function by the stochastic gradient descent algorithm via backpropagation:

$$\mathcal{L}(\mathsf{W}_t) = -\left[\mathbb{E}_{\mathbf{x}^+ \sim p^+} f_{\mathsf{W}_t}(\mathbf{x}^+) - \mathbb{E}_{\mathbf{x}^- \sim p^-_{\mathsf{W}_t}} f_{\mathsf{W}_t}(\mathbf{x}^-)\right] \tag{2.6}$$

To enforce the function f_{W_t} to be 1-Lipschitz, we add the following gradient penalty term [GAA$^+$17] to $\mathcal{L}(\mathsf{W}_t)$:

$$\lambda \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} f_{\mathsf{W}_t}(\hat{\mathbf{x}})\|_2 - 1)^2]$$

where $\hat{\mathbf{x}} = \alpha \mathbf{x}^+ + (1-\alpha)\mathbf{x}^-$, $\mathbf{x}^+ \sim p^+$, $\mathbf{x}^- \sim p^-_{\mathsf{W}_t}$, and $\alpha \sim U[0,1]$.

Synthesis-step. Obtaining increasingly difficult pseudo-negative samples is an integral part of the introspective learning framework, as it is crucial for tightening the decision boundary. To this end, we develop an efficient sampling procedure under the Wasserstein formulation. After the classification-step, we obtain the following distribution of pseudo-negatives:

$$p^-_{\mathsf{W}_t}(\mathbf{x}) = \frac{1}{Z_t} \exp\{f_{\mathsf{W}_t}(\mathbf{x})\} \cdot p^-_0(\mathbf{x}),\ t = 1, \ldots, T \tag{2.7}$$

where $Z_t = \int \exp\{f_{\mathsf{W}_t}(\mathbf{x})\} \cdot p^-_0(\mathbf{x}) d\mathbf{x}$; the initial distribution $p^-_0(\mathbf{x})$ is a Gaussian distribution $G(\mathbf{x}; 0, \sigma^2)$ or the distribution defined in Appendix **D**. We find that the distribution of Appendix **D** encourages the diversity of sampled images.

The following equivalence is shown in [LJT17, JLT17]:

$$\frac{p(y = +1|\mathbf{x}; \mathsf{W}_t)}{p(y = -1|\mathbf{x}; \mathsf{W}_t)} = \exp\{f_{\mathsf{W}_t}(\mathbf{x})\}. \tag{2.8}$$

The sampling strategy of [LJT17] was to carry out gradient ascent on the term $\ln \frac{p(y=+1|\mathbf{x};\mathsf{W}_t)}{p(y=-1|\mathbf{x};\mathsf{W}_t)}$. In Lemma 1 we chose $f(\mathbf{x})$ to be $\ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}}(\mathbf{x})}$. Using Bayes' rule, it is easy to see that $\nabla \ln \frac{p(y=+1|\mathbf{x};\mathsf{W}_t)}{p(y=-1|\mathbf{x};\mathsf{W}_t)}$ is loosely connected to $\nabla \ln \frac{p^+(\mathbf{x})}{p^-_{\mathsf{W}_t}(\mathbf{x})}$. Also, [ACB17, GAA$^+$17] argue that $f_{\mathsf{W}_t}(\mathbf{x})$ correlates with the quality of the sample $\mathbf{x}$. This motivates us to use the following sampling strategy. After initializing $\mathbf{x}$ by drawing a fair sample from $p^-_0(\mathbf{x})$, we increase $f_{\mathsf{W}_t}(\mathbf{x})$ using gradient ascent on the image $\mathbf{x}$ via backpropagation. Specifically, as shown in [WT11], we can obtain fair samples from the distribution $p^-_{\mathsf{W}_t}$ using the following update rule:

$$\Delta\mathbf{x} = \frac{\varepsilon}{2}\,\nabla f_{\mathsf{W}_t}(\mathbf{x}) + \eta$$

where ε is a time-varying step size and η is a random variable following the Gaussian distribution $N(0, \varepsilon)$. Gaussian noise term is added to make samples cover the full distribution. Inspired by [IZZE17], we found that injecting noise in the image space could be substituted by applying Dropout to the higher layers of CNN. In practice, we were able obtain the samples of enough diversity without step size annealing and noise injection.

As an early stopping criterion, we empirically find that the following is effective: (1) we measure the minimum and maximum $f_{\mathsf{W}_t}(\cdot)$ of positive examples; (2) we set the early stopping threshold to a random number from the uniform distribution between these two numbers. Intuitively, by matching the value of $f_{\mathsf{W}_t}(\cdot)$ positives and pseudo-negatives, we expect to obtain pseudo-negative samples that match the quality of positive samples.

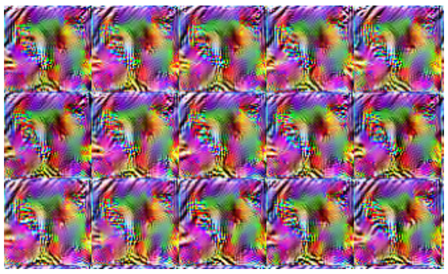
2.4.2 Expanding model capacity

In practice, we find that the version with the single classifier – which we call WINN-single – is expressive enough to capture the generative distribution under variety of applications. The introspective learning formulation [Tu07, LJT17, JLT17] allows us to model more complex distributions by adding a sequence of cascaded classifiers parameterized by $(\mathsf{W}^1, \ldots, \mathsf{W}^K)$. Then, we can model the distribution as:

$$p^-_{\mathsf{W}^k}(\mathbf{x}) = \frac{1}{Z_t}\exp\{f_{\mathsf{W}^k}(\mathbf{x})\} \cdot p^-_{\mathsf{W}^{k-1}}(\mathbf{x}),\ k = 2, \ldots, K \tag{2.9}$$

In the next sections, we demonstrate the modeling capability of WINN under cascaded classifiers, as well as its agnosticy to the type of base classifier.

2.4.3 GAN's discriminator vs. WINN's classifier


(a) WGAN-GP discriminator

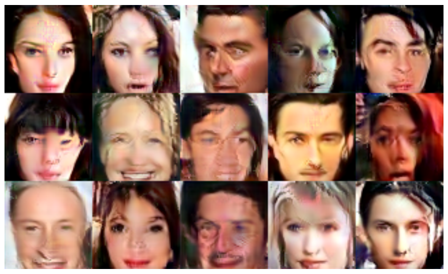

(b) WINN-single

Figure 2.3: **Synthesized images from the discriminators of WGAN and WINN trained on the CelebA dataset.** Both use the same ResNet architecture for the discriminator as the one adopted in [GAA+17].

GAN uses the competing discriminator and the generator whereas WINN maintains a single model being both generative and discriminative. Some general comparisons between GAN and INN have been provided in [LJT17, JLT17]. Below we make a few additional observations that are worth future exploration and discussions.

- First, the generator of GAN is a cost-effective option for image patch synthesis, as it works in a feed-forward fashion. However the generator of GAN is not meant to be trained as a classifier to perform the standard classification task, while the generator in the introspective framework is also a strong classifier. Section 2.5.6 shows WINN to have significant robustness to external adversarial examples.
- Second, the discriminator in GAN is meant to be a critic but not a generator. To show whether or not the discriminator in GAN can also be used as a generator, we train WGAN-GP [GAA+17] on the CelebA face dataset. Using the same CNN architecture (ResNet from [GAA+17]) that was used as GAN's discriminator, we also train a WINN-single model, making GAN's discriminator and WINN-single to have the identical CNN architecture. Applying the sampling strategy to WGAN-GP's discriminator allows us to synthesize image form WGAN-GP's discriminator as well and we show some samples in Figure 2.3 (a). These synthesized images are not like faces, yet they have been classified by the discriminator of WGAN-GP as "real" faces; this demonstrates the separation between the generator and the discriminator in GAN. In contrast, images synthesized by

WINN-single's CNN classifier are faces like, as shown in Figure 2.3 (b).

- Third, the discriminator of GAN may not be used as a direct discriminative classifier for the standard supervised learning task. As shown and discussed in ICN [JLT17], the introspective framework has the ability of classification for discriminator.

2.5 Experiments

2.5.1 Implementation

Classification-step. For training the discriminator network, we use Adam [KB15] with a mini-batch size of 100. The learning rate was set to 0.0001. We set $\beta_1 = 0.0$, and $\beta_2 = 0.9$, inspired by [GAA$^+$17]. Each batch consists of 50 positive images sampled from the set of positives S_+ and 50 pseudo-negative images sampled from the set of pseudo-negatives S_-. In each iteration, we limit total number of training images to $10,000$.

Synthesis-step. For synthesizing pseudo-negative images via back-propagation, we perform gradient ascent on the image space. In the first cascade, each image is initialized with a noise sampled from the distribution described in Appendix **D**. In the later cascades, images are initialized with the images sampled from the last cascade. We use Adam with a mini-batch size of 100. The learning rate was set to 0.01. We set $\beta_1 = 0.9$, and $\beta_2 = 0.99$.

2.5.2 Texture modeling

We evaluate the texture modeling capability of WINN. For a fair comparison, we use the same 7 texture images presented in [GEB15a] where each texture image has a size of 256×256. We follow the training method of Section 2.5.1 except that positive images are constructed by cropping 64×64 patches from the source texture image at random positions. We use network architecture of Appendix **C**. After training is done on the 64×64 patch-based model, we try to synthesize texture images of arbitrary size using the anysize-image-generation method following

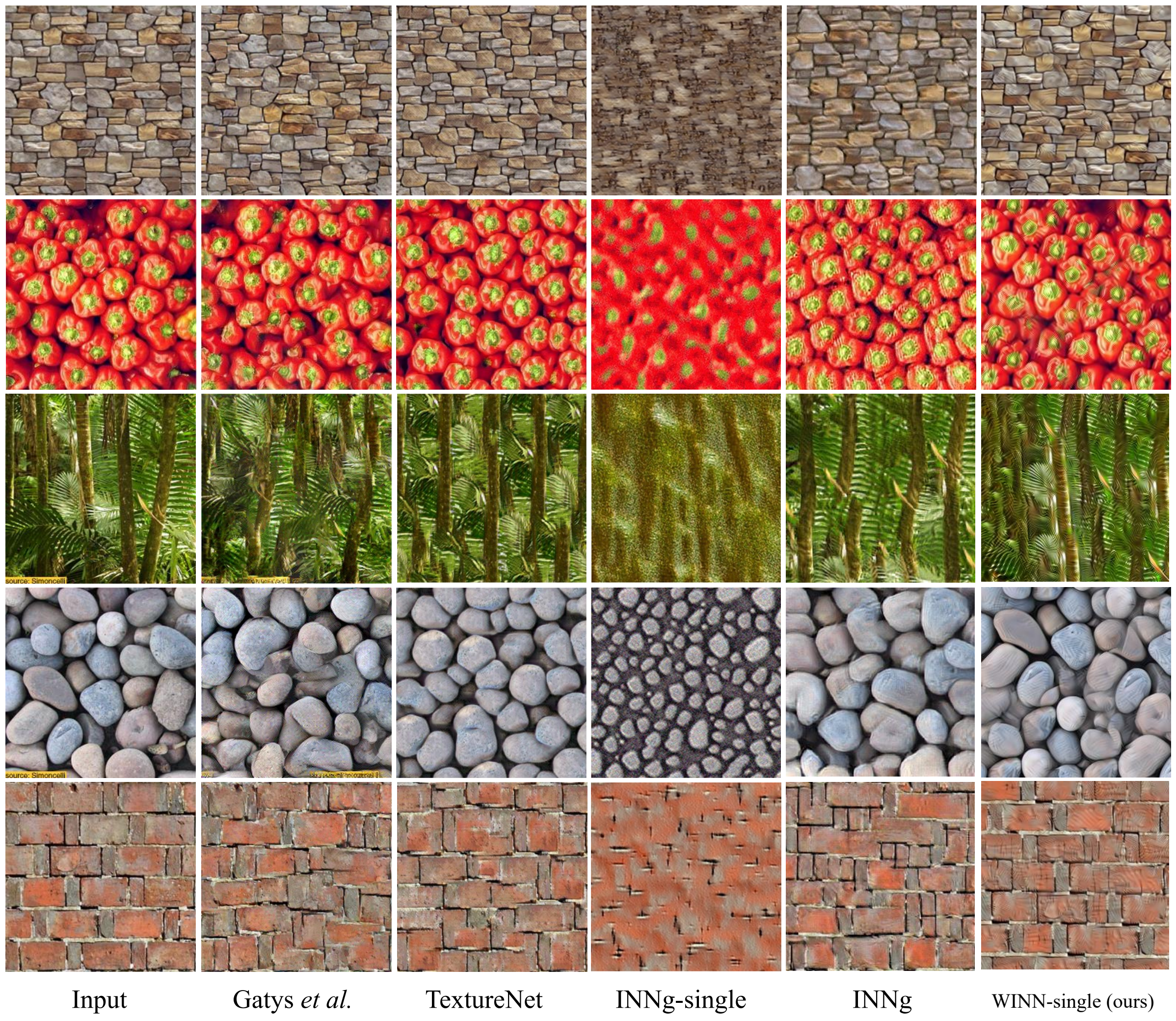


Figure 2.4: **More texture synthesis results.** Gatys *et al*. [GEB15a] and TextureNets [ULVL16] results are from [ULVL16].

[LJT17]. During the synthesis process, we keep a single working image of size 320×320. Note that we expand the image so that center 256×256 pixels are covered with equal probability. In each iteration, we sample 200 patches from the working image, and perform gradient ascent on the chosen patches. For the overlapping pixels between patches, we take the average of the gradients assigned to such pixels. We show synthesized texture images in Figure 2.2 and 2.4. WINN-single shows a significant improvement over INNg-single and comparable results to INNg (using 20 CNNs). It is worth noting that [GEB15b, ULVL16] leverage rich features of VGG-19 network pretrained on ImageNet. WINN and INNg instead train networks from scratch.

2.5.3 CelebA face modeling

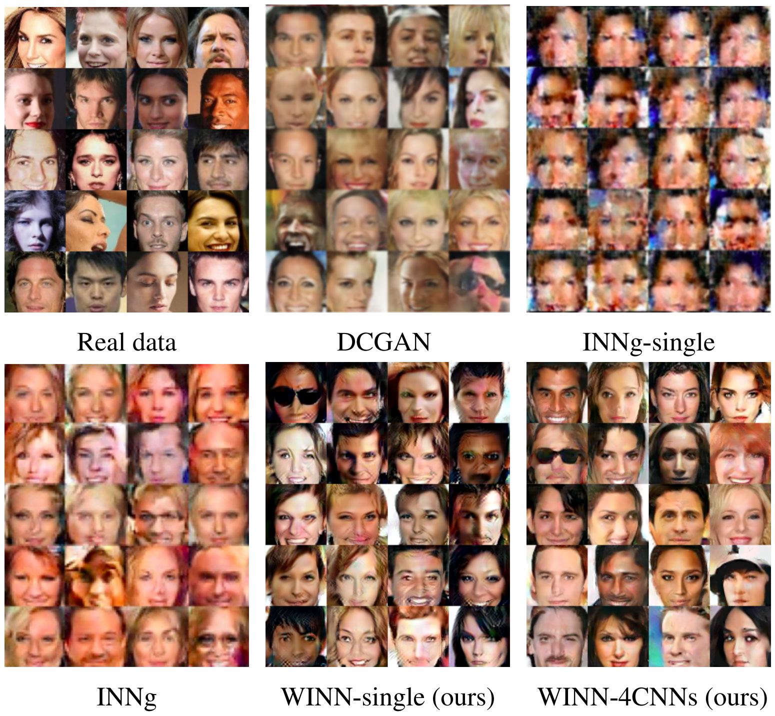


Figure 2.5: **Images generated by various models trained on CelebA.**

The CelebA dataset [LLWT15] consists of 202,599 face images of celebrities. This dataset has been widely used in the previous generative modeling works since it contains large pose variations and background clutters. The network architecture adopted here is described in Appendix **C**. In Figure 2.5, we show some synthesized face images using WINN-single and

WINN, as well as those by DCGAN [RMC16b], INNg-single, and INNg [LJT17]. WINN-single attains image quality even higher than that of INNg (12 CNNs).

2.5.4 SVHN modeling

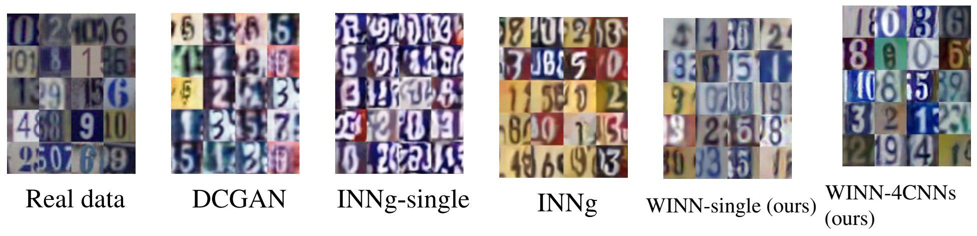


Figure 2.6: **Images generated by various models trained on SVHN.** DCGAN [RMC16b] result is from [LJT17].

SVHN [NWC+11] consists of 32×32 images from Google Street View. It contains $73,257$ training images, $26,032$ test images, and $531,131$ extra images. We use only the training images for the unsupervised SVHN modeling. We use the ResNet architecture described in [GAA+17]. Generated images by WINN-single and WINN (4 CNN classifiers) as well as DCGAN and INN are shown in Figure 2.6. The improvement of WINN over INNg is evident.

2.5.5 CIFAR-10 modeling

Table 2.1: **Inception score on CIFAR-10.** "-L" in Improved GANs means without labels.

Method	Score
Real data	$11.95 \pm .20$
WGAN-GP [GAA+17]	$\mathbf{7.86} \pm .07$
WGAN [ACB17]	$5.88 \pm .07$
DCGAN [RMC16b] (in [HLP+17])	$6.16 \pm .07$
ALI [DBP+17] (in [WFB17])	$5.34 \pm .05$
Improved GANs (-L) [SGZ+16a]	$4.36 \pm .04$
INNg-single [LJT17]	$1.95 \pm .01$
INNg [LJT17]	$3.04 \pm .02$
WINN-single (ours)	$4.62 \pm .05$
WINN-5CNNs (ours)	$5.58 \pm .05$

CIFAR-10 [KNHa] consists of $50,000$ training images and $10,000$ test images of size 32×32 in 10 classes. We use training images augmented by horizontal flips [KSH12] for unsupervised CIFAR-10 modeling. We use the ResNet given in [GAA^{+}17]. Figure 2.7 shows generated images by various models.

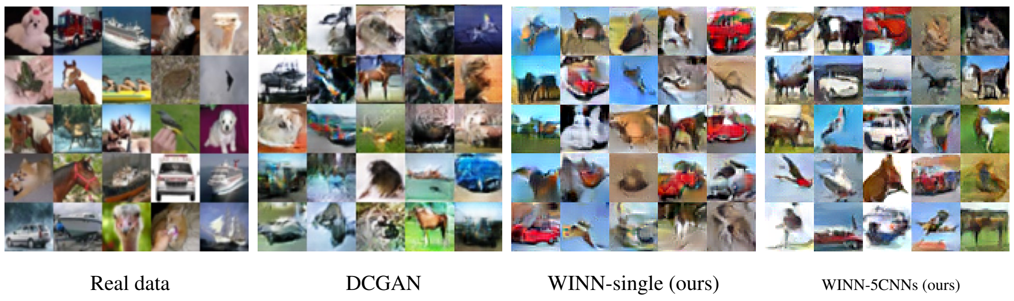


Figure 2.7: **Images generated by models trained on CIFAR-10.**

To measure the semantic discriminability, we compute the Inception scores [SGZ^{+}16a] on $50,000$ generated images. WINN shows its clear advantage over INN. WINN-5CNNs produces a result close to WGAN but there is still a gap to the state-of-the-art results by WGAN-GP.

2.5.6 Image classification and adversarial examples

To demonstrate the robustness of WINN as a discriminative classifier, we present experiments on the supervised classification tasks.

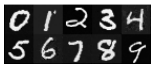
Training Methods. We add the Wasserstein loss term to the ICN [JLT17] loss function, obtaining the following:

$$\begin{aligned}\mathcal{L}(\mathsf{W}_t) = \quad & -\sum_{\mathbf{x}_i \in S_+} \ln \frac{\exp\{\mathbf{w}_t^{(1)y_i} \cdot \phi(\mathbf{x}_i;\mathbf{w}_t^{(0)})\}}{\sum_{k=1}^{K} \exp\{\mathbf{w}_t^{(1)k} \cdot \phi(\mathbf{x}_i;\mathbf{w}_t^{(0)})\}} \\ & +\alpha\Big(\sum_{\mathbf{x}_i \in S_-^t} f_{\mathsf{W}_t}(\mathbf{x}_i) - \sum_{\mathbf{x}_i \in S_+} f_{\mathsf{W}_t}(\mathbf{x}_i) \\ & +\lambda \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} f_{\mathsf{W}_t}(\hat{\mathbf{x}})\|_2 - 1)^2] \Big)\end{aligned}$$

where $\mathsf{W}_t =< \mathbf{w}_t^{(0)}, \mathbf{w}_t^{(1)_1}, \ldots, \mathbf{w}_t^{(1)_K} >$, $\mathbf{w}_t^{(0)}$ denotes the internal parameters for the CNN, and $\mathbf{w}_t^{(1)_k}$ denotes the top-layer weights for the k-th class. In the experiments, we set the weight of the WINN loss, α, to 0.01. We use the vanilla network architecture resembling [JLT17] as the baseline

Table 2.2: **Test errors on MNIST and SVHN.** When training on SVHN, we only use training set. All results except WINN-single and baseline ResNet-32 are from [JLT17]. [JLT17] adopted DCGAN [RMC16b] discriminator as their CNN architecture. The advantage of WINN over a vanilla CNN is evident. When applied to a stronger baseline such as ResNet-32, WINN is not losing ResNet's superior classification capability in standard supervised classification, while attaining special generative capability and robustness to adversarial attacks (see Table 2.3 and 2.4) that do not exist in ResNet. The images below on the right are the generated samples by the corresponding WINN (ResNet-32) classifiers reported in the table.

Method	Error
MNIST	
Baseline vanilla CNN (4 layers)	0.89%
ICN [JLT17]	0.81%
WINN-single vanilla (ours)	0.67%
Baseline ResNet-32	**0.45**%
WINN-single ResNet-32 (ours)	0.48%
SVHN	
Baseline ResNet-32	4.64%
WINN-single ResNet-32 (ours)	**4.50**%



CNN, which has less filters and parameters than the one in [JLT17]. We also use a ResNet-32 architecture with Layer Normalization [BKH16] on MNIST and SVHN. In the classification-step, we use Adam with a fixed learning rate of 0.001, β_1 of 0.0. In the synthesis-step, we use the Adam optimizer with a learning rate of 0.02 and β_1 of 0.9. Table 2.2 shows the errors on MNIST and SVHN.

Table 2.3: **Adversarial examples comparison between the baseline CNN and WINN on MNIST.** We first generate N adversarial examples using method A and count the number of adversarial examples misclassified by A ($= N_A$). **Adversarial error** of A is defined as test error rate against adversarial examples ($= N_A/N$). Then among A's mistakes, we count the number of adversarial examples misclassified by B ($= N_{A\cap B}$). Then **error correction rate** by B is $1 - N_{A\cap B}/N_A$. $\uparrow$ means higher the better; $\downarrow$ means lower the better. The comparisons are made in two groups: the first group builds on top a vanilla 4 layer CNN and the second group adopts ResNet-32. Note that the corresponding errors for these classifiers on the standard MNIST supervised classification task can be seen in Table 2.2.

Method	Adversarial error of **Method** $\downarrow$	Correction rate by **Method** $\uparrow$	Correction rate by **Baseline** $\downarrow$
Baseline vanilla CNN	32.41%	-	-
ICN [JLT17]	19.02%	52.58%	58.68%
WINN-single vanilla (ours)	**7.99**%	**90.00**%	**46.93**%
Baseline ResNet-32	11.28%	-	-
WINN-single ResNet-32 (ours)	**2.05**%	**89.68**%	**43.69**%

Robustness to adversarial examples. It is argued in [GPAM$^+$14b] that discriminative CNN's vulnerability to adversarial examples primarily arises due to its linear nature. Since the reclassification-by-synthesis process helps tighten the decision boundary (Figure 2.1), one might expect that CNNs trained with the WINN algorithm are more robust to adversarial examples. Note that unlike existing methods for adversarial defenses [GSS15, KGB17], our method does not train networks with specific types of adversarial examples. With test images of MNIST and SVHN, we adopt "fast gradient sign method" [GPAM$^+$14b] ($\varepsilon = 0.125$ for MNIST and $\varepsilon = 0.005$ for SVHN) to generate adversarial examples clipped to range $[-1, 1]$, which differs from [JLT17]. We experiment with two networks having the same architecture and only differing in training method (the standard cross-entropy loss vs. the WINN procedure). We call the former as the baseline CNN. We summarize the results in Table 2.3. Compared to ICN [JLT17], WINN significantly reduces the adversarial error to 7.99% and improves the correction rate to 90.00%. In addition, we have adopted the ResNet-32 architecture into WINN. See Table 2.3 and 2.4. We still obtain the adversarial error reduction and correction rate improvement on MNIST and SVHN ($\varepsilon = 0.005$) with ResNet-32. Our observation is that WINN is not necessarily improving over a strong baseline for the supervised classification task but its advantage on adversarial attacks is evident.

Table 2.4: **Adversarial examples comparison between the baseline ResNet-32 [HZRS16] and WINN on SVHN.** Note that the corresponding errors for these classifiers on the standard supervised SVHN classification task can be seen in Table 2.2.

Method	Adversarial error of **Method**	Correction rate by **Method** $\uparrow$	Correction rate by **Baseline** $\downarrow$
Baseline ResNet-32	29.29%	-	-
WINN-single ResNet-32 (ours)	**19.53**%	74.39%	51.37%

2.5.7 Agnostic to different architectures

In Figure 2.8, we demonstrate our algorithm being agnostic to the type of classifier, by varying network architectures to ResNet [HZRS16] and DenseNet [HLvdMW17]. Little modification was required to adapt two architectures for WINN.

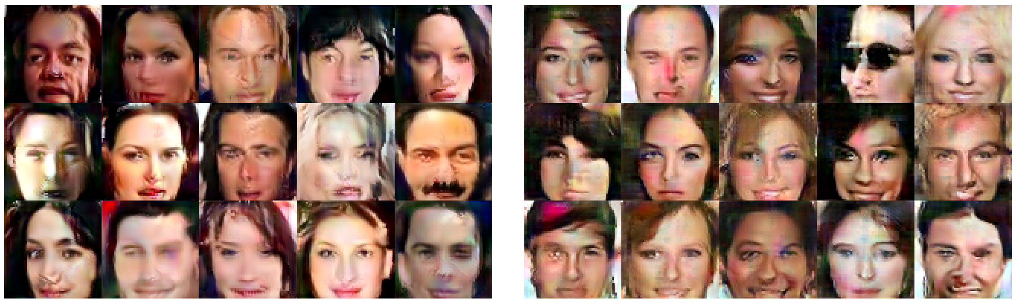


Figure 2.8: **Synthesized CelebA images with varying network architectures.** We use a single CNN for each experiment.

2.6 Conclusion

In this chapter, we have introduced Wasserstein introspective neural networks (WINN) that produce encouraging results as a generator and a discriminative classifier at the same time. WINN is able to achieve model size reduction over the previous introspective neural networks (INN) by a factor of 20. In most of the images shown in the chapter, we find a single CNN classifier in WINN being sufficient to produce visually appealing images as well as significant error reduction against adversarial examples. WINN is agnostic to the architecture design of CNN and we demonstrate results on three networks including a vanilla CNN, ResNet [HZRS16], and DenseNet [HLvdMW17] networks where popular CNN discriminative classifiers are turned into generative models under the WINN procedure. WINN can be adopted in a wide range of applications in computer vision such as image classification, recognition, and generation.

2.7 Acknowledgments

This work was supported by NSF IIS-1717431 and NSF IIS-1618477.

This chapter, in full, is a reprint of the material as it appears in “Wasserstein Introspective Neural Networks,” Kwonjoon Lee, Weijian Xu, Fan Fan, Zhuowen Tu, Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2018. The dissertation author is the primary investigator and author of this material.

Chapter 3

ViTGAN: Training GANs with Vision Transformers

3.1 Introduction

Convolutional neural networks (CNNs) [LBD+89a] are dominating computer vision today, thanks to their powerful capability of convolution (weight-sharing and local-connectivity) and pooling (translation equivariance). Recently, however, Transformer architectures [VSP+17] have started to rival CNNs in many vision tasks.

In particular, Vision Transformers (ViTs) [DBK+21], which interpret an image as a sequence of *tokens* (analogous to *words* in natural language), have been shown to achieve comparable classification accuracy with smaller computational budgets (*i.e*, fewer FLOPs) on the ImageNet benchmark. Unlike CNNs, ViTs capture a different inductive bias through self-attention where each patch is attended to *all* patches of the same image. ViTs, along with their variants [TCD+20, THK+21], though still in their infancy, have demonstrated advantages in modeling non-local contextual dependencies [RBK21, SGLS21] as well as promising efficiency and scalability. Since their recent inception, ViTs have been used in various tasks such as object detection [BKT+20], video recognition [BWT21, ADH+21], multitask pre-training [CWG+20], *etc*.

In this paper, we examine whether Vision Transformers can perform the task of image generation *without using convolution or pooling*, and more specifically, whether ViTs can be used to train generative adversarial networks (GANs) with comparable quality to CNN-based GANs. While we can naively train GANs following the design of the standard ViT [DBK+21], we find that GAN training becomes highly unstable when coupled with ViTs, and that adversarial training is frequently hindered by high-variance gradients in the later stage of discriminator training. Furthermore, conventional regularization methods such as gradient penalty [GAA+17, MNG18], spectral normalization [MKKY18] cannot resolve the instability issue, even though they are proved to be effective for CNN-based GAN models (shown in Fig. 3.4). As unstable training is uncommon in the CNN-based GANs training with appropriate regularization, this presents a unique challenge to the design of ViT-based GANs.

We propose several necessary modifications to stabilize the training dynamics and facilitate the convergence of ViT-based GANs. In the *discriminator*, we design an improved spectral normalization that enforces Lipschitz continuity for stabilizing the training dynamics. In the *generator*, we propose two key modifications to the layer normalization and output mapping layers after studying several architecture designs. Our ablation experiments validate the necessity of the proposed techniques and their central role in achieving stable and superior image generation.

The experiments are conducted on three public image synthesis benchmarks: CIFAR-10, CelebA, and LSUN bedroom. The results show that our model, named *ViTGAN*, yields comparable performance to the leading CNN-based StyleGAN2 [KLA+20, ZLL+20] when trained under the same setting. Moreover, we are able to outperform StyleGAN2 by combining the StyleGAN2 discriminator with our ViTGAN generator.

Note that it is not our intention to claim ViTGAN is superior to the best-performing GAN models such as StyleGAN2 + ADA [KAH+20] which are equipped with highly-optimized hyperparameters, architecture configurations, and sophisticated data augmentation methods. Instead, our work aims to close the performance gap between the conventional CNN-based GAN architectures and the novel GAN architecture composed of vanilla ViT layers. Furthermore, the ablation in Table 3.4 shows the advantages of ViT's intrinsic capability (*i.e*, adaptive connection weight and global context) for image generation.

3.2 Related Work

Generative Adversarial Networks. Generative adversarial nets (GANs) [GPAM$^+$14b] model the target distribution using adversarial learning. It is typically formulated as a min-max optimization problem minimizing some distance between the real and generated data distributions, *e.g*, through various f-divergences [NCT16] or integral probability metrics (IPMs) [Mül97, SE20] such as the Wasserstein distance [ACB17].

GAN models are notorious for unstable training dynamics. As a result, numerous efforts have been proposed to stabilize training, thereby ensuring convergence. Common approaches include spectral normalization [MKKY18], gradient penalty [GAA$^+$17, MNG18, KAHK17], consistency regularization [ZZOL20, ZSL$^+$21], and data augmentation [ZLL$^+$20, KAH$^+$20, ZZC$^+$20, TTN$^+$21]. These techniques are all designed inside convolutional neural networks (CNN) and have been only verified in convolutional GAN models. However, we find that these methods are insufficient for stabilizing the training of Transformer-based GANs. A similar finding was reported in [CXH21] on a different task of pretraining. This paper introduces several novel techniques to overcome the unstable adversarial training of Vision Transformers.

Vision Transformers. Vision Transformer (ViT) [DBK$^+$21] is a convolution-free Transformer that performs image classification over a sequence of image patches. ViT demonstrates the superiority of the Transformer architecture over the classical CNNs by taking advantage of pretraining on large-scale datasets. Afterward, DeiT [TCD$^+$20] improves ViTs' sample efficiency using knowledge distillation as well as regularization tricks. MLP-Mixer [THK+21] further drops self-attention and replaces it with an MLP to mix the per-location feature. In parallel, ViT has been extended to various computer vision tasks such as object detection [BKT+20], action recognition in video [BWT21, ADH+21], and multitask pretraining [CWG+20]. Our work is among the first to exploit Vision Transformers in the GAN model for image generation.

Generative Transformers in Vision. Motivated by the success of GPT-3 [BMR+20], a few pilot works study image generation using Transformer by autoregressive learning [CRC+20, ERO21] or cross-modal learning between image and text [RPG+21]. These methods are different from ours as they model image generation as a autoregressive sequence learning problem. On the contrary, our work trains Vision Transformers in the generative adversarial training paradigm. Recent work of [HZ21], embeds (cross-)attention module within the CNN backbone [KLA+20] in a similar spirit to [ZGMO19]. The closest work to ours is TransGAN [JCW21], presenting a GAN model based on Swin Transformer backbone [LLC+21]. Our approach is complementary to theirs as we propose key techniques for training stability within the original ViT backbone [DBK+21].

3.3 Preliminaries: Vision Transformers (ViTs)

Vision Transformer [DBK+21] is a pure transformer architecture for image classification that operates upon a sequence of image patches. The 2D image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is flattened into a sequence of image patches, following the raster scan, denoted by $\mathbf{x}_p \in \mathbb{R}^{L \times (P^2 \cdot C)}$, where $L = \frac{H \times W}{P^2}$ is the effective sequence length and $P \times P \times C$ is the dimension of each image patch.

Following BERT [DCLT19], a learnable classification embedding $\mathbf{x}_{\text{class}}$ is prepended to the image sequence along with the added 1D positional embeddings $\mathbf{E}_{pos}$ to formulate the patch embedding $\mathbf{h}_0$. The architecture of ViT follows the Transformer architecture [VSP+17].

$$\mathbf{h}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^L\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(L+1)\times D} \tag{3.1}$$

$$\mathbf{h}'_\ell = \text{MSA}(\text{LN}(\mathbf{h}_{\ell-1})) + \mathbf{h}_{\ell-1}, \qquad \ell = 1, \ldots, L \tag{3.2}$$

$$\mathbf{h}_\ell = \text{MLP}(\text{LN}(\mathbf{h}'_\ell)) + \mathbf{h}'_\ell, \qquad \ell = 1, \ldots, L \tag{3.3}$$

$$\mathbf{y} = \text{LN}(\mathbf{h}_L^0) \tag{3.4}$$

Equation 3.2 applies multi-headed self-attention (MSA). Given learnable matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ corresponding to query, key, and value representations, a single self-attention head is computed by:

$$\text{Attention}_h(\mathbf{X}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\Big)\mathbf{V}, \tag{3.5}$$

where $\mathbf{Q} = \mathbf{X}\mathbf{W}_q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_k$, and $\mathbf{V} = \mathbf{X}\mathbf{W}_v$. Multi-headed self-attention aggregates information from H self-attention heads by means of concatenation and linear projection: $\text{MSA}(\mathbf{X}) = \text{concat}_{h=1}^H[\text{Attention}_h(\mathbf{X})]\mathbf{W} + \mathbf{b}$.

3.4 Method

Fig. 3.1 illustrates the architecture of the proposed ViTGAN with a ViT discriminator and a ViT-based generator. We find that directly using ViT as the discriminator makes the training volatile. We introduce techniques to both generator and discriminator to stabilize the training dynamics and facilitate the convergence: *(1)* regularization on ViT discriminator and *(2)* new architecture for generator.

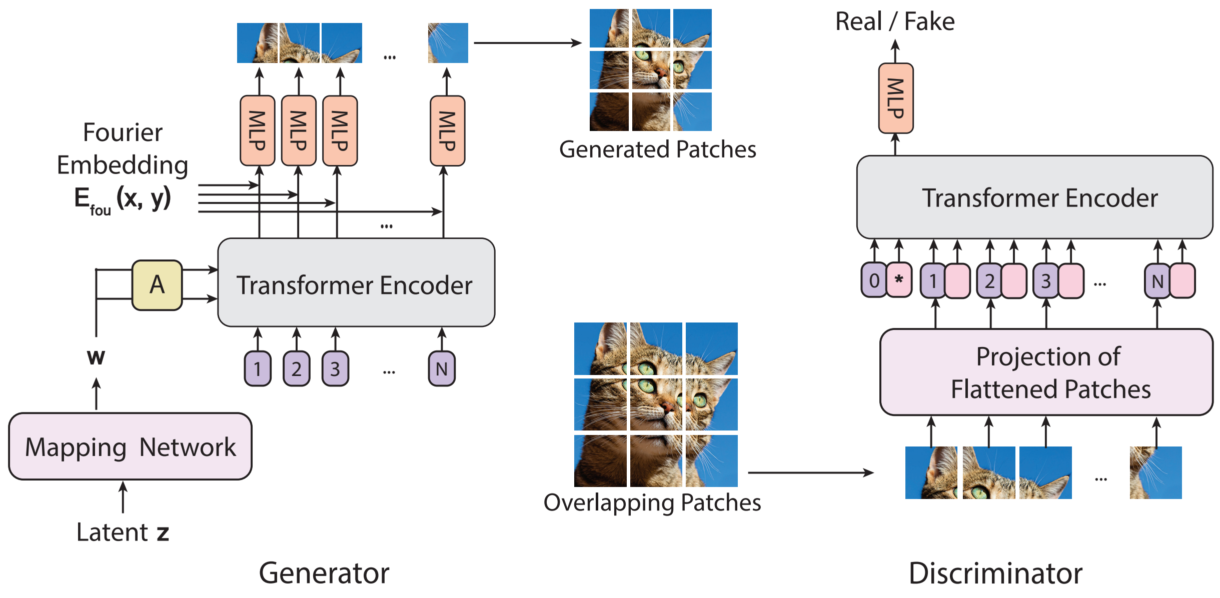


Figure 3.1: **Overview of the proposed ViTGAN framework.** Both the generator and the discriminator are designed based on the Vision Transformer (ViT). Discriminator score is derived from the classification embedding (denoted as [*] in the Figure). The generator generates pixels patch-by-patch based on patch embeddings.

3.4.1 Regularizing ViT-based discriminator

Enforcing Lipschitzness of Transformer Discriminator. Lipschitz continuity plays a critical role in GAN discriminators. It was first brought to attention as a condition to approximate the Wasserstein distance in WGAN [ACB17], and later was confirmed in other GAN settings [FRL$^+$18, MKKY18, ZGMO19] beyond the Wasserstein loss. In particular, [ZLS$^+$19] proves that Lipschitz discriminator guarantees the existence of the optimal discriminative function as well as the existence of a unique Nash equilibrium. A very recent work [KPM21], however, shows that Lipschitz constant of standard dot product self-attention (*i.e*, Equation 3.5) layer can be unbounded, rendering Lipschitz continuity violated in ViTs. To enforce Lipschitzness of our ViT discriminator, we adopt *L2 attention* proposed in [KPM21]. As shown in Equation 3.6, we replace the dot product similarity with Euclidean distance and also tie the weights for the projection matrices for query and key in self-attention:

$$\text{Attention}_h(\mathbf{X}) = \text{softmax}\Big(-\frac{d(\mathbf{X}\mathbf{W}_q, \mathbf{X}\mathbf{W}_k)}{\sqrt{d_h}}\Big)\mathbf{X}\mathbf{W}_v, \quad \text{where} \quad \mathbf{W}_q = \mathbf{W}_k, \tag{3.6}$$

$\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are the projection matrices for query, key, and value, respectively. $d(\cdot,\cdot)$ computes *vectorized L2 distances* between two sets of points. $\sqrt{d_h}$ is the feature dimension for each head. This modification improves the stability of Transformers when used for GAN discriminators.

Improved Spectral Normalization. To further strengthen the Lipschitz continuity, we also apply spectral normalization (SN) [MKKY18] in the discriminator training. The standard SN uses power iterations to estimate spectral norm of the projection matrix for each layer in the neural network. Then it divides the weight matrix with the estimated spectral norm, so Lipschitz constant of the resulting projection matrix equals 1. We find Transformer blocks are sensitive to the scale of Lipschitz constant, and the training exhibits very slow progress when the SN is used (*c.f*Table 3.3b). Similarly, we find R1 gradient penalty cripples GAN training when ViT-based discriminators are used (*c.f*Figure 3.4). [DCL21] suggests that the small Lipschitz constant of MLP block may cause the output of Transformer collapse to a rank-1 matrix. To resolve this, we propose to increase the spectral norm of the projection matrix.

We find that multiplying the normalized weight matrix of each layer with the **spectral norm at initialization** is sufficient to solve this problem. Concretely, we use the following update rule for our spectral normalization, where σ computes the standard spectral norm of weight matrices:

$$\bar{W}_{\text{ISN}}(\mathbf{W}) := \sigma(\mathbf{W}_{init}) \cdot \mathbf{W}/\sigma(\mathbf{W}). \tag{3.7}$$

Overlapping Image Patches. ViT discriminators are prone to overfitting due to their exceeding learning capacity. Our discriminator and generator use the same image representation that partitions an image as a sequence of non-overlapping patches according to a predefined grid $P \times P$. These arbitrary grid partitions, if not carefully tuned, may encourage the discriminator to memorize local cues and stop providing meaningful loss for the generator. We use a simple trick to mitigate this issue by allowing some overlap between image patches. For each border edge of the patch, we extend it by o pixels, making the effective patch size $(P+2o) \times (P+2o)$, where $o = \frac{P}{2}$. This operation has a connection to a convolution operation with kernel $(P+2o) \times (P+2o)$ and stride $P \times P$. Note that the extraction of (non-overlapping) patches in the Vanilla ViT [DBK$^+$21] also has a connection to a convolution operation with kernel $P \times P$ and stride $P \times P$.

This results in a sequence with the same length but less sensitivity to the predefined grids. It may also give the Transformer a better sense of which ones are neighboring patches to the current patch, hence giving a better sense of locality.

Convolutional Projection. To allow Transformers to leverage local context as well as global context, we apply convolutions when computing $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ in Equation 3.5. While variants of this idea were proposed in [WXC$^+$21, GHW$^+$21], we find the following simple option works well: we apply 3×3 convolution after reshaping image token embeddings into a feature map of size $\frac{H}{P} \times \frac{W}{P} \times D$. We note that this formulation does not harm the expressiveness of the original Transformer, as the original $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ projection can be recovered by using the identity convolution kernel.

3.4.2 Generator Design

Designing a generator based on the ViT architecture is a nontrivial task. A challenge is converting ViT from predicting a set of class labels to generating pixels over a spatial region. Before introducing our model, we discuss two plausible baseline models, as shown in Fig. 3.2 (A) and 3.2 (B). Both models swap ViT's input and output to generate pixels from embeddings, specifically from the latent vector $\mathbf{w}$ derived from a Gaussian noise vector $\mathbf{z}$ by an MLP, *i.e*, $\mathbf{w} = \text{MLP}(\mathbf{z})$ (called mapping network [KLA19] in Fig. 3.2). The two baseline generators differ

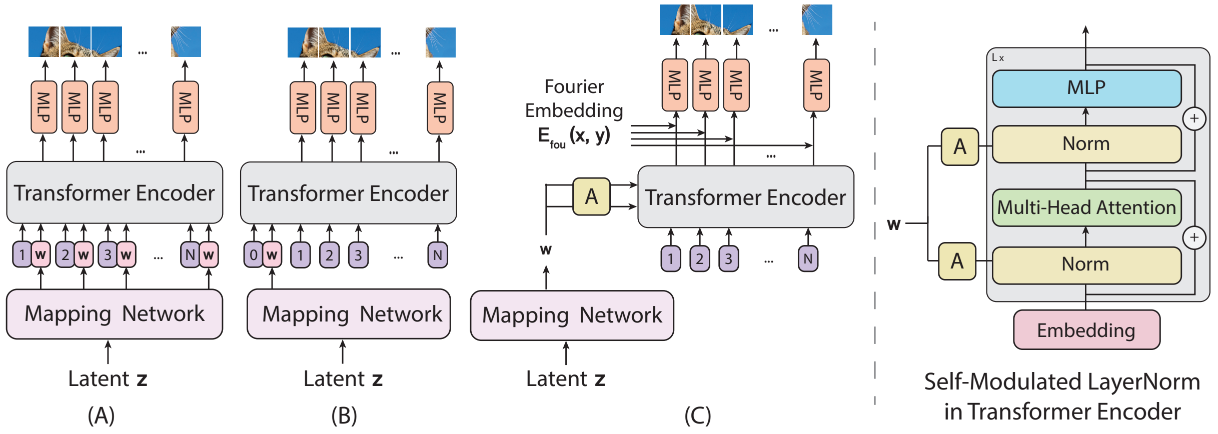


Figure 3.2: **Generator Architecture Variants.** The diagram on the left shows three generator architectures we consider: **(A)** adding intermediate latent embedding $\mathbf{w}$ to every positional embedding, **(B)** prepending $\mathbf{w}$ to the sequence, and **(C)** replacing normalization with self-modulated layernorm (SLN) computed by learned affine transform (denoted as **A** in the figure) from $\mathbf{w}$. On the right, we show the details of the self-modulation operation applied in the Transformer block.

in their input sequences. Fig. 3.2 (A) takes as input a sequence of positional embeddings and adds the intermediate latent vector $\mathbf{w}$ to every positional embedding. Alternatively, Fig. 3.2 (B) prepends the sequence with the latent vector. This design is inspired by inverting ViT where $\mathbf{w}$ is used to replace the classification embedding $\mathbf{h}_L^0$ in Equation 3.4.

To generate pixel values, a linear projection $\mathbf{E} \in \mathbb{R}^{D\times(P^2\cdot C)}$ is learned in both models to map a D-dimensional output embedding to an image patch of shape $P\times P\times C$. The sequence of $L = \frac{H\times W}{P^2}$ image patches $[\mathbf{x}_p^i]_{i=1}^L$ are finally reshaped to form an whole image $\mathbf{x}$.

These baseline transformers perform poorly compared to the CNN-based generator. We propose a novel generator following the design principle of ViT. Our ViTGAN Generator, shown in Fig. 3.2 (c), consists of two components (1) a transformer block and (2) an output mapping layer.

$$
\begin{aligned}
\mathbf{h}_0 &= \mathbf{E}_{pos}, && \mathbf{E}_{pos} \in \mathbb{R}^{L\times D}, && (3.8)\\
\mathbf{h}'_\ell &= \mathrm{MSA}(\mathrm{SLN}(\mathbf{h}_{\ell-1}, \mathbf{w})) + \mathbf{h}_{\ell-1}, && \ell = 1, \dots, L, \mathbf{w} \in \mathbb{R}^D && (3.9)\\
\mathbf{h}_\ell &= \mathrm{MLP}(\mathrm{SLN}(\mathbf{h}'_\ell, \mathbf{w})) + \mathbf{h}'_\ell, && \ell = 1, \dots, L && (3.10)\\
\mathbf{y} &= \mathrm{SLN}(\mathbf{h}_L, \mathbf{w}) = [\mathbf{y}^1, \cdots, \mathbf{y}^L] && \mathbf{y}^1, \dots, \mathbf{y}^L \in \mathbb{R}^D && (3.11)\\
\mathbf{x} &= [\mathbf{x}_p^1, \cdots, \mathbf{x}_p^L] = [f_\theta(\mathbf{E}_{fou}, \mathbf{y}^1), \dots, f_\theta(\mathbf{E}_{fou}, \mathbf{y}^L)] && \mathbf{x}_p^i \in \mathbb{R}^{P^2\times C}, \mathbf{x} \in \mathbb{R}^{H\times W\times C} && (3.12)
\end{aligned}
$$

The proposed generator incorporates two modifications to facilitate the training.

Self-Modulated LayerNorm. Instead of sending the noise vector $\mathbf{z}$ as the input to ViT, we use $\mathbf{z}$ to modulate the layernorm operation in Equation 3.9. This is known as self-modulation [CLHG19] since the modulation depends on no external information. The self-modulated layernorm (SLN) in Equation 3.9 is computed by:

$$\mathrm{SLN}(\mathbf{h}_\ell, \mathbf{w}) = \mathrm{SLN}(\mathbf{h}_\ell, \mathrm{MLP}(\mathbf{z})) = \gamma_\ell(\mathbf{w}) \odot \frac{\mathbf{h}_\ell - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \beta_\ell(\mathbf{w}), \tag{3.13}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ track the mean and the variance of the summed inputs within the layer, and γ_l and β_l compute adaptive normalization parameters controlled by the latent vector derived from $\mathbf{z}$. $\odot$ is the element-wise dot product.

Implicit Neural Representation for Patch Generation. We use an implicit neural representation [PFS$^+$19, MON$^+$19, TSM$^+$20, SMB$^+$20] to learn a continuous mapping from a patch embedding $\mathbf{y}^i \in \mathbb{R}^D$ to patch pixel values $\mathbf{x}_p^i \in \mathbb{R}^{P^2\times C}$. When coupled with Fourier features [TSM$^+$20] or sinusoidal activation functions [SMB$^+$20], implicit representations can constrain the space of generated samples to the space of smooth-varying natural signals. Concretely, similarly to [ADK$^+$21], $\mathbf{x}_p^i = f_\theta(\mathbf{E}_{fou}, \mathbf{y}^i)$ where $\mathbf{E}_{fou} \in \mathbb{R}^{P^2\cdot D}$ is a Fourier encoding of $P \times P$ spatial locations and $f_\theta(\cdot,\cdot)$ is a 2-layer MLP. Each positional embedding is a linear projection of pixel coordinate followed by a sine activation function (hence the name Fourier encoding). The pixel coordinates for P^2 pixels are normalized to lie between -1.0 and 1.0. The 2-layer MLP takes positional embedding $\mathbf{E}_{fou}$ as its input, and it is conditioned on patch embedding $\mathbf{y}^i$ via weight modulation as in [KLA+20, ADK+21]. We find implicit representation to be particularly helpful for training GANs with ViT-based generators, *c.f* Table 3.3a.

The generator and discriminator can have different image grids and thus different sequence lengths. We find that it is often sufficient to increase the sequence length or feature dimension of only the discriminator when scaling our model to higher resolution images.

3.5 Experiments

We train and evaluate our model on various standard benchmarks for image generation, including *CIFAR-10* [KNHa], *LSUN bedroom* [YZS+15] and *CelebA* [LLWT15]. We consider three resolution settings: 32×32, 64×64, and 128×128. We strive to use a consistent setup across different resolutions and datasets, and as such, keep all key hyper-parameters, except for the number of Transformer blocks and patch sizes, the same.

3.5.1 Experiment details

Datasets. The *CIFAR-10* dataset [KNHa] is a standard benchmark for image generation, containing 50K training images and 10K test images. Inception score (IS) [SGZ+16b] and Fréchet Inception Distance (FID) [HRU+17] are computed over the 50K images. The *LSUN bedroom* dataset [YZS+15] is a large-scale image generation benchmark, consisting of ~3 million training images and 300 images for validation. On this dataset, FID is computed against the training set due to the small validation set. The *CelebA* dataset [LLWT15] comprises 162,770 unlabeled face images and 19,962 test images. We use the *aligned* version of CelebA, which is different from *cropped* version used in prior literature [RMC16a, JCW21].

Implementation Details. We consider three resolution settings: 32$\times$32 on the CIFAR dataset, 64$\times$64 on CelebA and LSUN bedroom, and 128$\times$128 on LSUN bedroom. For 32$\times$32 resolution, we use a 4-block ViT-based discriminator and a 4-block ViT-based generator. For 64$\times$64 resolution, we increase the number of blocks to 6. Following ViT-Small [DBK$^+$21], the input/output feature dimension is 384 for all Transformer blocks, and the MLP hidden dimension is 1,536. Unlike [DBK$^+$21], we choose the number of attention heads to be 6. We find increasing the number of heads does not improve GAN training. For 32$\times$32 resolution, we use patch size 4$\times$4, yielding a sequence length of 64 patches. For 64$\times$64 resolution, we simply increase the patch size to 8$\times$8, keeping the same sequence length as in 32$\times$32 resolution. For 128$\times$128 resolution generator, we use patch size 8$\times$8 and 8 Transformer blocks. For 128$\times$128 resolution discriminator, we maintain the sequence length of 64 and 4 Transformer blocks. Similarly to [XSM$^+$21], 3×3 convolutions with stride 2 were used until desired sequence length is reached.

Translation, Color, Cutout, and Scaling data augmentations [ZLL$^+$20, KAH$^+$20] are applied with probability 0.8. All baseline transformer-based GAN models, including ours, use balanced consistency regularization (bCR) with $\lambda_{real} = \lambda_{fake} = 10.0$. Other than bCR, we do not employ regularization methods typically used for training ViTs [TCD$^+$20] such as Dropout, weight decay, or Stochastic Depth. We found that LeCam regularization [TJL$^+$21], similar to bCR, improves the performance. But for clearer ablation, we do not include the LeCam regularization. We train our models with Adam with $\beta_1 = 0.0$, $\beta_2 = 0.99$, and a learning rate of 0.002 following the practice of [KLA$^+$20]. In addition, we employ non-saturating logistic loss [GPAM$^+$14b], exponential moving average of generator weights [KALL18], and equalized learning rate [KALL18]. We use a mini-batch size of 128.

Both ViTGAN and StyleGAN2 are based on Tensorflow 2 implementation[1] and trained on Google Cloud TPU v2-32 and v3-8.

[1]`https://github.com/moono/stylegan2-tf-2.x`

Table 3.1: **Comparison to representative GAN architectures on unconditional image generation benchmarks.** *Results from the original papers. All other results are our replications and trained with DiffAug [ZLL+20] + bCR for fair comparison. $\downarrow$ means lower is better.

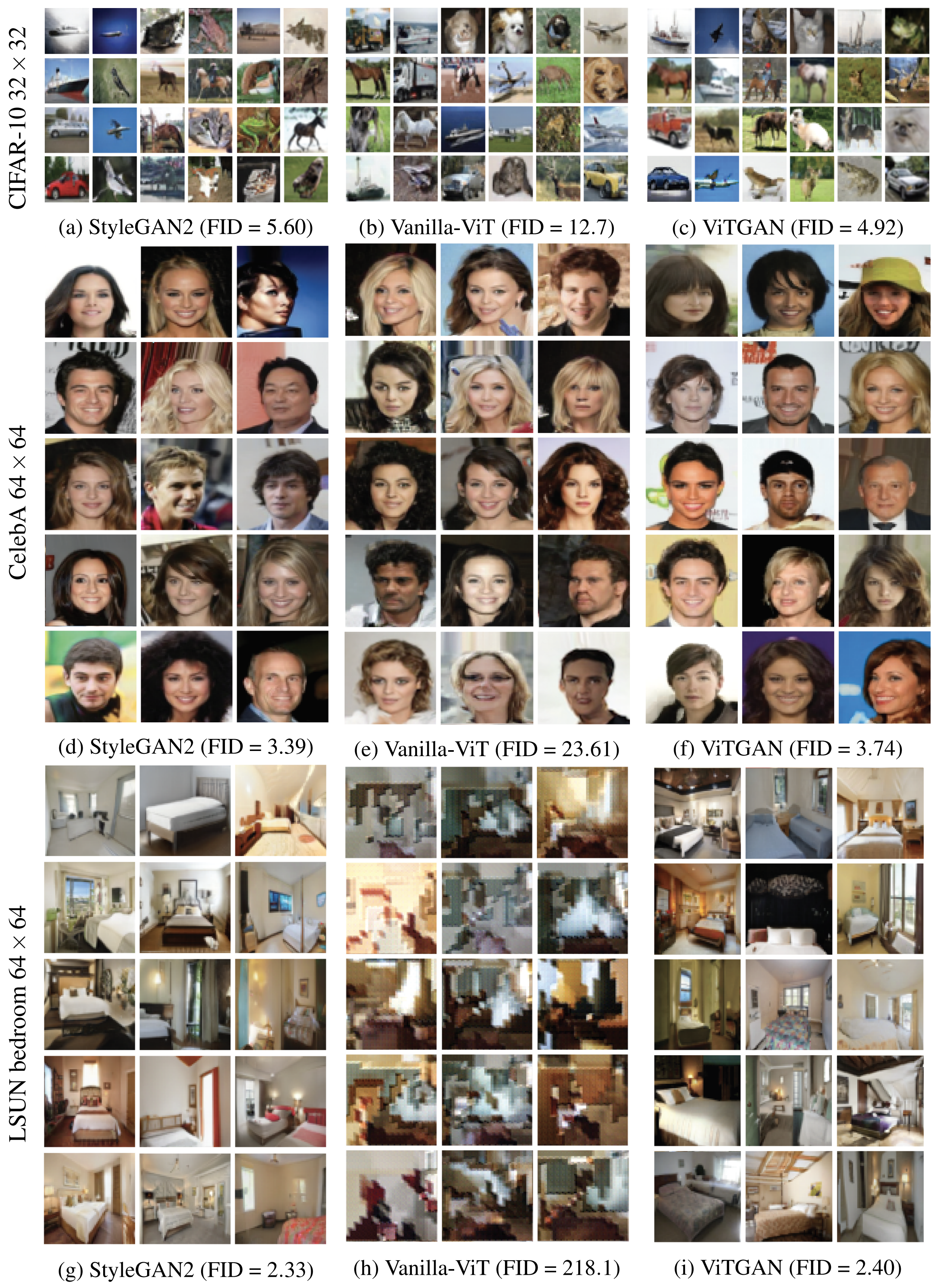
Architecture	CIFAR 10		CelebA 64x64		LSUN 64x64		LSUN 128x128	
	FID $\downarrow$	**IS $\uparrow$**	**FID $\downarrow$**	**IS $\uparrow$**	**FID $\downarrow$**	**IS $\uparrow$**	**FID $\downarrow$**	**IS $\uparrow$**
BigGAN+ DiffAug [ZLL+20]	8.59*	9.25*	-	-	-	-	-	-
StyleGAN2 [KLA+20]	5.60	9.41	**3.39**	**3.43**	2.33	2.44	3.26	2.26
TransGAN [JCW21]	9.02*	9.26*	-	-	-	-	-	-
Vanilla-ViT	12.7	8.40	20.2	2.57	218.1	2.20	-	-
ViTGAN (Ours)	4.92	9.69	3.74	3.21	2.40	2.28	2.48	2.26
StyleGAN2-D+ViTGAN-G (Ours)	**4.57**	**9.89**	-	-	**1.49**	**2.46**	**1.87**	**2.32**

Positional Embedding. Each positional embedding of ViT networks is a linear projection of patch position followed by a sine activation function. The patch positions are normalized to lie between -1.0 and 1.0.

3.5.2 Main Results

Table 3.1 shows the main results on three standard benchmarks for image synthesis. Our method is compared with the following baseline architectures. *TransGAN* [JCW21] is the only existing convolution-free GAN that is entirely built on the Transformer architecture. *Vanilla-ViT* is a ViT-based GAN that employs the generator illustrated in Fig. 3.2 (A) and a vanilla ViT discriminator without any techniques discussed in Section 3.4.1. For a fair comparison, R1 penalty and bCR [ZSL+21] + DiffAug [ZLL+20] were used for this baseline. The architecture with the generator illustrated in Fig. 3.2 (B) is separately compared in Table 3.3a. In addition, BigGAN [BDS19] and StyleGAN2 [KLA+20] are also included as state-of-the-art CNN-based GAN models.

Our ViTGAN model outperforms other Transformer-based GAN models by a large margin. This stems from the improved stable GAN training on the Transformer architecture. As shown in Fig. 3.4, our method overcomes the spikes in gradient magnitude and stabilizes the training

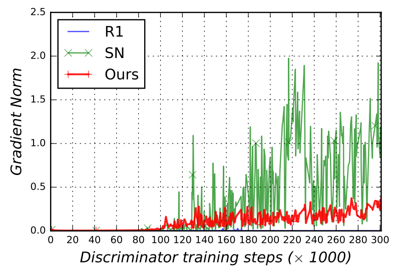


(a) StyleGAN2 (FID = 5.60) (b) Vanilla-ViT (FID = 12.7) (c) ViTGAN (FID = 4.92)

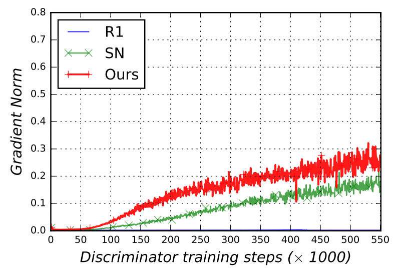
(d) StyleGAN2 (FID = 3.39) (e) Vanilla-ViT (FID = 23.61) (f) ViTGAN (FID = 3.74)

(g) StyleGAN2 (FID = 2.33) (h) Vanilla-ViT (FID = 218.1) (i) ViTGAN (FID = 2.40)

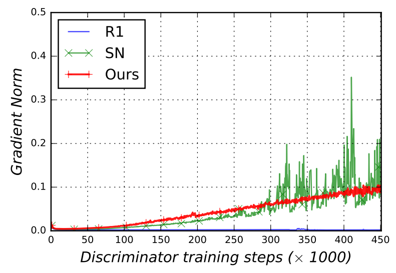
Figure 3.3: **Qualitative Comparison.** We compare our ViTGAN with StyleGAN2, and our best Transformer baseline, *i.e*, a vanilla pair of ViT generator and discriminator described in Section 3.5.2, on the CIFAR-10 32×32, CelebA 64×64 and LSUN Bedroom 64×64 datasets.



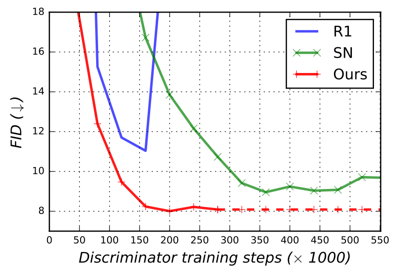
(a) Gradient L_2 Norm (CIFAR)



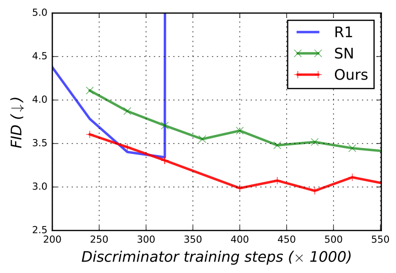
(b) Gradient L_2 Norm (CelebA)



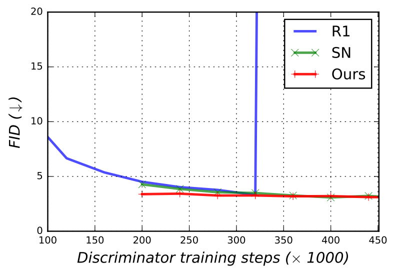
(c) Gradient L_2 Norm (LSUN)



(d) FID (CIFAR)



(e) FID (CelebA)



(f) FID (LSUN)

Figure 3.4: **(a-c) Gradient magnitude (L_2 norm over all parameters) of ViT discriminator and (d-f) FID score (lower the better)** as a function of training iteration. Our ViTGAN are compared with two baselines of Vanilla ViT discriminators with R1 penalty and spectral norm (SN). The remaining architectures are the same for all methods. Our method overcomes the spikes of gradient magnitude, and achieve lower FIDs (on CIFAR and CelebA) or a comparable FID (on LSUN).

dynamics. This enables ViTGAN to converge to either a lower or a comparable FID on different datasets.

ViTGAN achieves comparable performance to the leading CNN-based models, *i.e* BigGAN and StyleGAN2. Note that in Table 3.1, to focus on comparing the architectures, we use a generic version of StyleGAN2. As shown in Fig. 3.3, the image fidelity of the best Transformer baseline (Middle Row) has been notably improved by the proposed ViTGAN model (Last Row). Even compared with StyleGAN2, ViTGAN generates images with comparable quality and diversity. Notice there appears to be a perceivable difference between the images generated by Transformers and CNNs, *e.g*in the background of the CelebA images. Both the quantitative results and qualitative comparison substantiate the efficacy of the proposed ViTGAN as a competitive Transformer-based GAN model.

Table 3.2: **ViTGAN on CIFAR-10 when paired with CNN-based generators or discriminators**.

Generator	Discriminator	FID ↓	IS ↑
ViTGAN	ViTGAN	4.92	9.69
StyleGAN2	StyleGAN2	5.60	9.41
StyleGAN2	Vanilla-ViT	17.3	8.57
StyleGAN2	ViTGAN	8.02	9.15
ViTGAN	StyleGAN2	**4.57**	**9.89**

3.5.3 Ablation Studies

We conduct ablation experiments on the CIFAR dataset to study the contributions of the key techniques and verify the design choices in our model.

Compatibility with CNN-based GANs. In Table 3.2, we mix and match the generator and discriminator of our ViTGAN and the leading CNN-based GAN: StyleGAN2. With the StyleGAN2 generator, our ViTGAN discriminator outperforms the vanilla ViT discriminator. Besides, our ViTGAN generator still works together with the StyleGAN2 discriminator. The results show the proposed techniques are compatible with both Transformer-based and CNN-based generators and discriminators.

Generator architecture. Table 3.3a shows GAN performances under three generator architectures, as shown in Fig. 3.2. Fig. 3.2 (B) underperforms other architectures. We find that Fig. 3.2 (A) works well but lags behind Fig. 3.2 (C) due to its instability. Regarding the mapping between patch embedding and pixels, implicit neural representation (denoted as NeurRep in Table 3.3a) offers consistently better performance than linear mapping, suggesting the importance of implicit neural representation in the ViTGAN generators.

Discriminator regularization. Table 3.3b validates the necessity of the techniques discussed in Section 3.4.1. First, we compare GAN performances under different regularization methods. Training GANs with ViT discriminator under R1 penalty [MNG18] is highly unstable, as shown in Fig. 3.4, sometimes resulting in complete training failure (indicated as IS=NaN in Row 1 of Table 3.3b). Spectral normalization (SN) is better than R1 penalty. But SN still exhibits high-variance gradients and therefore suffers from low quality scores. Our L_2+ISN regularization improves the stability significantly (*c.f* Fig. 3.4) and achieves the best IS and FID scores as a consequence. On the other hand, the overlapping patch is a simple trick that yields further improvement over the L_2+ISN method. However, the overlapping patch by itself does not work well (see a comparison between Row 3 and 9). The above results validate the essential role of these techniques in achieving the final performance of the ViTGAN model.

Table 3.3: Ablation studies of ViTGAN on CIFAR-10.

Embedding	**Output Mapping**	**FID** $\downarrow$	**IS** $\uparrow$
Fig 3.2 (A)	Linear	14.3	8.60
Fig 3.2 (A)	NeurRep	11.3	9.05
Fig 3.2 (B)	Linear	328	1.01
Fig 3.2 (B)	NeurRep	285	2.46
Fig 3.2 (C)	Linear	15.1	8.58
Fig 3.2 (C)	NeurRep	**6.66**	**9.30**

(a) **Ablation studies of generator architectures.** NeurRep denotes implicit neural representation. ViT discriminator without convolutional projection is used for this ablation.

Aug.	**Reg.**	**Overlap**	**Proj.**	**FID** $\downarrow$	**IS** $\uparrow$
✗	R1	✗	✗	2e4	NaN
✗	R1	✓	✗	129	4.99
✓	R1	✓	✗	13.1	8.71
✗	SN	✗	✗	121	4.28
✓	SN	✗	✗	10.2	8.78
✓	L_2+SN	✗	✗	168	2.36
✓	ISN	✗	✗	8.51	9.12
✓	L_2+ISN	✗	✗	8.36	9.02
✓	L_2+ISN	✓	✗	**6.66**	**9.30**
✓	L_2+ISN	✓	✓	**4.92**	**9.69**

(b) **Ablation studies of discrminator regularization.** 'Aug.', 'Reg.', 'Overlap' and 'Proj.' stand for DiffAug [ZLL+20] + bCR [ZSL+21], and regularization method, overlapping image patches, and convolutional projection, respectively.

Necessity of self-attention. Two intrinsic advantages of ViTs over CNNs are (a) adaptive connection weights based on both content and position, and (b) globally contextualized representation. We show the importance of each of them by ablation analysis in Table 3.4. We conduct this ablation study on the LSUN Bedroom dataset considering that its large-scale (~3 million

Table 3.4: Ablations studies of self-attention on LSUN Bedroom 32x32.

Method	**attention**	**local**	**global**	**FID** $\downarrow$
StyleGAN2	✗	✓	✗	2.59
ViT	✓	✗	✓	1.94
MLP-Mixer	✗	✗	✓	3.23
ViT-local	✓	✓	✗	1.79
ViTGAN	✓	✓	✓	**1.57**

images) helps compare sufficiently trained vision transformers.

First, to see the importance of adapting connection weights, we maintain the network topology (*i.e*, each token is connected to all other tokens) and use fixed weights between tokens as in CNNs. We adopt the recently proposed MLP-Mixer [THK+21] as an instantiation of this idea. By comparing the 2nd and 3rd rows of the table, we see that it is beneficial to adapt connection weight between tokens. The result of MLP-Mixer on the LSUN Bedroom dataset suggests that the deficiency of weight adaptation cannot be overcome by merely training on a large-scale dataset.

Second, to see the effect of self-attention scope (global *v.s.* local), we adopt a local self-attention of window size=3×3 [RPV+19]. The ViT based on local attention (4th row in Table 3.4) outperforms the ones with global attention. This shows the importance of considering local context (2nd row in Table 3.4). The ViT with convolutional projection (last row in Table 3.4) — which considers *both local and global contexts* — outperforms other methods.

Effects of Data Augmentation. Table 3.5 presents the comparison of the Convolution-based GAN architectures (BigGAN and StyleGAN2) and our Transformer-based architecture (ViTGAN). This table complements the results in Table 1 of the main paper by a closer examination of the network architecture performance with and without using data augmentation. The differentiable data augmentation (DiffAug) [ZLL+20] is used in this study.

As shown, data augmentation plays a more critical role in ViTGAN. This is not unexpected because discriminators built on Transformer architectures are more capable of over-fitting or memorizing the data. DiffAug increases the diversity of the training data, thereby mitigating the overfitting issue in adversarial training. Nevertheless, with DiffAug, ViTGAN performs comparably to the leading-performing CNN-based GAN models: BigGAN and StyleGAN2.

In addition, Table 3.5 includes the model performance without using the balanced consistency regularization (bCR) [ZSL+21].

Table 3.5: **Effectiveness of data augmentation and regularization on the CIFAR-10 dataset.** ViTGAN results here are based on the ViT backbone without convolutional projection.

Method	**Data Augmentation**	**Conv**	**FID** $\downarrow$	**IS** $\uparrow$
StyleGAN2	None	Y	5.60	9.41
StyleGAN2	DiffAug	Y	9.89	9.40
ViTGAN	None	N	30.72	7.75
ViTGAN	DiffAug	N	6.66	9.30
ViTGAN w/o. bCR	DiffAug	N	8.84	9.02

3.6 Conclusion and Limitations

We have introduced ViTGAN, leveraging Vision Transformers (ViTs) in GANs, and proposed essential techniques to ensuring its training stability and improving its convergence. Our experiments on standard benchmarks demonstrate that the presented model achieves comparable performance to leading CNN-based GANs. Regarding the limitations, ViTGAN is a novel GAN architecture consisting solely of Transformer layers. This could be improved by incorporating advanced training (*e.g*, [JS21, SSK20]) or coarse-to-fine architectures (*e.g*, [LLC$^+$21]) into the ViTGAN framework. Besides, we have not verified ViTGAN on high-resolution images. Our paper establishes a concrete baseline of ViT on resolutions up to 128$\times$128 and we hope that it can facilitate future research to use vision transformers for high-resolution image synthesis.

3.7 Acknowledgments

This work was supported in part by Google Cloud Platform (GCP) Credit Award. We would also like to acknowledge Cloud TPU support from Google's TensorFlow Research Cloud (TRC) program.

This chapter, in part, is a reprint of the material as it appears in "ViTGAN: Training GANs with Vision Transformers," Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, Ce Liu, International Conference on Learning Representations (ICLR), 2022. The dissertation author is the primary investigator and author of this material.

Chapter 4

Meta-Learning with Differentiable Convex Optimization

4.1 Introduction

The ability to learn from a few examples is a hallmark of human intelligence, yet it remains a challenge for modern machine learning systems. This problem has received significant attention from the machine learning community recently where few-shot learning is cast as a meta-learning problem (*e.g*, [RL17, FAL17, VBL$^+$16, SSZ17]). The goal is to minimize generalization error across a distribution of tasks with few training examples. Typically, these approaches are composed of an *embedding model* that maps the input domain into a feature space and a *base learner* that maps the feature space to task variables. The meta-learning objective is to learn an embedding model such that the base learner generalizes well across tasks.

While many choices for base learners exist, nearest-neighbor classifiers and their variants (*e.g*, [SSZ17, VBL$^+$16]) are popular as the classification rule is simple and the approach scales well in the low-data regime. However, discriminatively trained linear classifiers often outperform nearest-neighbor classifiers (*e.g*, [CKY08, MGE11]) in the low-data regime as they can exploit the negative examples which are often more abundant to learn better class boundaries. Moreover, they can effectively use high dimensional feature embeddings as model capacity can be controlled by appropriate regularization such as weight sparsity or norm.

Hence, in this paper, we investigate linear classifiers as the base learner for a meta-learning based approach for few-shot learning. The approach is illustrated in Figure 4.1 where a linear support vector machine (SVM) is used to learn a classifier given a set of labeled training examples and the generalization error is computed on a novel set of examples from the same task. The key challenge is computational since the meta-learning objective of minimizing the generalization error across tasks requires training a linear classifier in the inner loop of optimization (see Section 4.3). However, the objective of linear models is convex and can be solved efficiently. We observe that two additional properties arising from the convex nature that allows efficient meta-learning: implicit differentiation of the optimization [Bar18, GFC$^+$16] and the low-rank nature of the classifier in the few-shot setting. The first property allows the use of off-the-shelf convex optimizers to estimate the optima and implicitly differentiate the optimality or Karush-Kuhn-Tucker (KKT) conditions to train embedding model. The second property means that the number of optimization variables in the *dual formation* is far smaller than the feature dimension for few-shot learning.

To this end, we have incorporated a *differentiable quadratic programming* (QP) solver [AK17] which allows end-to-end learning of the embedding model with various linear classifiers, *e.g*, multiclass support vector machines (SVMs) [CS02] or linear regression, for few-shot classification tasks. Making use of these properties, we show that our method is practical and offers substantial gains over nearest neighbor classifiers at a modest increase in computational costs (see Table 4.3). Our method achieves state-of-the-art performance on 5-way 1-shot and 5-shot classification for popular few-shot benchmarks including miniImageNet [VBL$^+$16, RL17], tieredImageNet [RRT$^+$18], CIFAR-FS [BHTV19], and FC100 [ORL18].

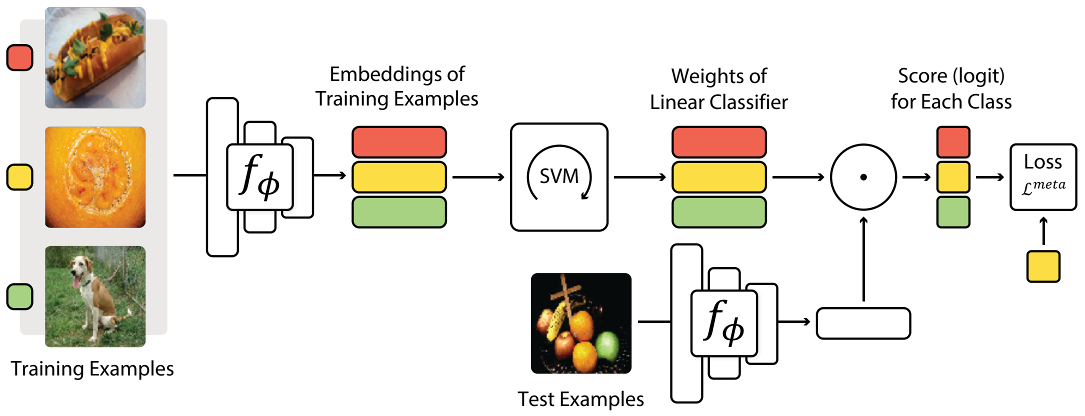

Embeddings of Training Examples
Weights of Linear Classifier
Score (logit) for Each Class
f_ϕ
SVM
Loss $\mathcal{L}^{meta}$
f_ϕ
Training Examples
Test Examples

Figure 4.1: **Overview of our approach, MetaOptNet.** Schematic illustration of our method MetaOptNet on an 1-shot 3-way classification task. The meta-training objective is to learn the parameters ϕ of a feature embedding model f_ϕ that generalizes well across tasks when used with regularized linear classifiers (*e.g*, SVMs). A task is a tuple of a few-shot training set and a test set (see Section 4.3 for details).

4.2 Related Work

Meta-learning studies what aspects of the learner (commonly referred to as *bias* or *prior*) effect generalization across a distribution of tasks [Sch87, Thr98, VD02]. Meta-learning approaches for few-shot learning can be broadly categorized these approaches into three groups. *Gradient-based methods* [RL17, FAL17] use gradient descent to adapt the embedding model parameters (*e.g*, all layers of a deep network) given training examples. *Nearest-neighbor methods* [VBL$^+$16, SSZ17] learn a distance-based prediction rule over the embeddings. For example, prototypical networks [SSZ17] represent each class by the mean embedding of the examples, and the classification rule is based on the distance to the nearest class mean. Another example is matching networks [VBL$^+$16] that learns a kernel density estimate of the class densities using the embeddings over training data (the model can also be interpreted as a form of *attention* over training examples). *Model-based methods* [MRCA18, MYMT18] learn a parameterized predictor to estimate model parameters, *e.g*, a recurrent network that predicts parameters analogous to a few steps of gradient descent in parameter space. While gradient-based methods are general, they are prone to overfitting as the embedding dimension grows [MRCA18, RRS$^+$19]. Nearest-neighbor approaches offer simplicity and scale well in the few-shot setting. However, nearest-neighbor methods have no mechanisms for feature selection and are not very robust to noisy features.

Our work is related to techniques for *backpropagation* though optimization procedures. Domke [Dom12] presented a generic method based on *unrolling* gradient-descent for a fixed number of steps and automatic differentiation to compute gradients. However, the trace of the optimizer (*i.e*, the intermediate values) needs to be stored in order to compute the gradients which can be prohibitive for large problems. The storage overhead issue was considered in more detail by Maclaurin *et al*. [MDA15] where they studied low precision representations of the optimization trace of deep networks. If the argmin of the optimization can be found analytically, such as in unconstrained quadratic minimization problems, then it is also possible to compute the gradients analytically. This has been applied for learning in low-level vision problems [TLAF07, SR14]. A concurrent and closely related work [BHTV19] uses this idea to learn few-shot models using ridge-regression base learners which have closed-form solutions. We refer readers to Gould *et al*. [GFC$^+$16] which provides an excellent survey of techniques for differentiating argmin and argmax problems.

Our approach advocates the use of linear classifiers which can be formulated as convex learning problems. In particular, the objective is a *quadratic program* (QP) which can be efficiently solved to obtain its *global optima* using gradient-based techniques. Moreover, the solution to convex problems can be characterized by their Karush-Kuhn-Tucker (KKT) conditions which allow us to *backpropagate* through the learner using the *implicit function theorem* [KP12]. Specifically, we use the formulation of Amos and Kolter [AK17] which provides efficient GPU routines for computing solutions to QPs and their gradients. While they applied this framework to learn representations for constraint satisfaction problems, it is also well-suited for few-shot learning as the problem sizes that arise are typically small.

While our experiments focus on linear classifiers with hinge loss and ℓ_2 regularization, our framework can be used with other loss functions and non-linear kernels. For example, the ridge regression learner used in [BHTV19] can be implemented within our framework allowing a direct comparison.

4.3 Meta-learning with Convex Base Learners

We first derive the meta-learning framework for few-shot learning following prior work (*e.g*, [SSZ17, RL17, FAL17]) and then discuss how convex base learners, such as linear SVMs, can be incorporated.

4.3.1 Problem formulation

Given the training set $\mathcal{D}^{train} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$, the goal of the base learner $\mathcal{A}$ is to estimate parameters θ of the predictor $y = f(\boldsymbol{x}; \theta)$ so that it generalizes well to the unseen test set $\mathcal{D}^{test} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{Q}$. It is often assumed that the training and test set are sampled from the same distribution and the domain is mapped to a feature space using an embedding model f_ϕ parameterized by ϕ. For optimization-based learners, the parameters are obtained by minimizing the empirical loss over training data along with a regularization that encourages simpler models. This can be written as:

$$\theta = \mathcal{A}(\mathcal{D}^{train}; \phi) = \arg\min_{\theta} \mathcal{L}^{base}(\mathcal{D}^{train}; \theta, \phi) + \mathcal{R}(\theta) \tag{4.1}$$

where $\mathcal{L}^{base}$ is a loss function, such as the negative log-likelihood of labels, and $\mathcal{R}(\theta)$ is a regularization term. Regularization plays an important role in generalization when training data is limited.

Meta-learning approaches for few-shot learning aim to minimize the generalization error across a distribution of tasks sampled from a task distribution. Concretely, this can be thought of as learning over a collection of tasks: $\mathcal{T} = \{(\mathcal{D}_i^{train}, \mathcal{D}_i^{test})\}_{i=1}^{I}$, often referred to as a *meta-training* set. The tuple $(\mathcal{D}_i^{train}, \mathcal{D}_i^{test})$ describes a training and a test dataset, or a task. The objective is to learn an embedding model ϕ that minimizes generalization (or test) error across tasks given a base learner A. Formally, the learning objective is:

$$\min_{\phi} \mathbb{E}_{\mathcal{T}} \left[\mathcal{L}^{meta}(\mathcal{D}^{test}; \theta, \phi), \text{where } \theta = \mathcal{A}(\mathcal{D}^{train}; \phi) \right]. \tag{4.2}$$

Figure 4.1 illustrates the training and testing for a single task. Once the embedding model f_ϕ is learned, its generalization is estimated on a set of *held-out* tasks (often referred to as a *meta-test* set) $\mathcal{S} = \{(\mathcal{D}_j^{train}, \mathcal{D}_j^{test})\}_{j=1}^{J}$ computed as:

$$\mathbb{E}_{\mathcal{S}} \left[\mathcal{L}^{meta}(\mathcal{D}^{test}; \theta, \phi), \text{where } \theta = \mathcal{A}(\mathcal{D}^{train}; \phi) \right]. \tag{4.3}$$

Following prior work [RL17, FAL17], we call the stages of estimating the expectation in Equation 4.2 and 4.3 as meta-training and meta-testing respectively. During meta-training, we keep an additional held-out *meta-validation* set to choose the hyperparameters of the meta-learner and pick the best embedding model.

4.3.2 Episodic sampling of tasks

Standard few-shot learning benchmarks such as miniImageNet [RL17] evaluate models in K-way, N-shot classification tasks. Here K denotes the number of classes, and N denotes the number of training examples per class. Few-shot learning techniques are evaluated for small values of N, typically $N \in \{1, 5\}$. In practice, these datasets do not explicitly contain tuples $(\mathcal{D}_i^{train}, \mathcal{D}_i^{test})$, but each task for meta-learning is constructed “on the fly" during the meta-training stage, commonly described as an episode.

For example, in prior work [VBL$^+$16, RL17], a task (or episode) $_i = (\mathcal{D}_i^{train}, \mathcal{D}_i^{test})$ is sampled as follows. The overall set of categories is C^{train}. For each episode, categories

C_i containing K categories from the C^{train} are first sampled (with replacement); then training (support) set $\mathcal{D}_i^{train} = \{(\boldsymbol{x}_n, y_n) \mid n = 1, \ldots, N \times K, y_n \in C_i\}$ consisting of N images per category is sampled; and finally, the test (query) set $\mathcal{D}_i^{test} = \{(\boldsymbol{x}_n, y_n) \mid n = 1, \ldots, Q \times K, y_n \in C_i\}$ consisting of Q images per category is sampled.

We emphasize that we need to sample without replacement, *i.e*, $\mathcal{D}_i^{train} \cap \mathcal{D}_i^{test} =$, to optimize the generalization error. In the same manner, meta-validation set and meta-test set are constructed on the fly from C^{val} and C^{test}, respectively. In order to measure the embedding model's generalization to *unseen categories*, C^{train}, C^{val}, and C^{test} are chosen to be mutually disjoint.

4.3.3 Convex base learners

The choice of the base learner A has a significant impact on Equation 4.2. The base learner that computes $\theta = \mathcal{A}(D^{train}; \phi)$ has to be efficient since the expectation has to be computed over a distribution of tasks. Moreover, to estimate parameters ϕ of the embedding model the gradients of the task test error $\mathcal{L}^{meta}(\mathcal{D}^{test}; \theta, \phi)$ with respect to ϕ have to be efficiently computed. This has motivated simple base learners such as nearest class mean [SSZ17] for which the parameters of the base learner θ are easy to compute and the objective is differentiable.

We consider base learners based on multi-class linear classifiers (*e.g*, support vector machines (SVMs) [CS02, WW99], logistic regression, and ridge regression) where the base-learner's objective is *convex*. For example, a K-class linear SVM can be written as $\theta = \{\boldsymbol{w}_k\}_{k=1}^{K}$. The Crammer and Singer [CS02] formulation of the multi-class SVM is:

$$\begin{aligned} &\theta = \mathcal{A}(\mathcal{D}^{train};) = \arg\min_{\{\boldsymbol{w}_k\}} \min_{\{\xi_i\}} \frac{1}{2} \sum_k ||\boldsymbol{w}_k||_2^2 + C \sum_n \xi_n \\ &\text{subject to} \\ &\boldsymbol{w}_{y_n} \cdot f_(\boldsymbol{x}_n) - \boldsymbol{w}_k \cdot f_(\boldsymbol{x}_n) \geq 1 - \delta_{y_n,k} - \xi_n, \ \forall n, k \end{aligned} \tag{4.4}$$

where $\mathcal{D}^{train} = \{(\boldsymbol{x}_n, y_n)\}$, C is the regularization parameter and $\delta_{\cdot,\cdot}$ is the Kronecker delta function.

Gradients of the SVM objective. From Figure 4.1, we see that in order to make our system end-to-end trainable, we require that the solution of the SVM solver should be differentiable with respect to its input, *i.e*, we should be able to compute $\{\frac{\partial \theta}{\partial f(\boldsymbol{x}_n)}\}_{n=1}^{N \times K}$. The objective of SVM is convex and has a unique optimum. This allows for the use of implicit function theorem (*e.g*, [KP12, DR09, Bar18]) on the optimality (KKT) conditions to obtain the necessary gradients. For the sake of completeness, we derive the form of the theorem for convex optimization problems as stated in [Bar18]. Consider the following convex optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_0(\theta, z) \\ \text{subject to} \quad & f(\theta, z) \preceq 0 \\ & h(\theta, z) = 0. \end{aligned} \tag{4.5}$$

where the vector $\theta \in \mathbb{R}^d$ is the optimization variable of the problem, the vector $z \in \mathbb{R}^e$ is the input parameter of the optimization problem, which is $\{f_\phi(\boldsymbol{x}_n)\}$ in our case. We can optimize the objective by solving for the saddle point $(\tilde{\theta}, \tilde{\lambda}, \tilde{\nu})$ of the following Lagrangian:

$$L(\theta, \lambda, \nu, z) = f_0(\theta, z) + \lambda^T f(\theta, z) + \nu^T h(\theta, z). \tag{4.6}$$

In other words, we can obtain the optimum of the objective function by solving $g(\tilde{\theta}, \tilde{\lambda}, \tilde{\nu}, z) = 0$ where

$$g(\theta, \lambda, \nu, z) = \begin{bmatrix} \nabla_\theta L(\theta, \lambda, \nu, z) \\ \mathbf{diag}(\lambda) f(\theta, z) \\ h(\theta, z) \end{bmatrix}. \tag{4.7}$$

Given a function $f(x) : \mathbb{R}^n \to \mathbb{R}^m$, denote $D_x f(x)$ as its Jacobian $\in \mathbb{R}^{m \times n}$.

Theorem 2 *(From Barratt [Bar18]) Suppose* $g(\tilde{\theta}, \tilde{\lambda}, \tilde{\nu}, z) = 0$*. Then, when all derivatives exist,*

$$D_z\tilde{\theta} = -D_\theta g(\tilde{\theta}, \tilde{\lambda}, \tilde{\nu}, z)^{-1} D_z g(\tilde{\theta}, \tilde{\lambda}, \tilde{\nu}, z). \tag{4.8}$$

This result is obtained by applying the implicit function theorem to the KKT conditions. Thus, once we compute the optimal solution $\tilde{\theta}$, we can obtain a closed-form expression for the gradient of $\tilde{\theta}$ with respect to the input data. This obviates the need for backpropagating through the entire optimization trajectory since the solution does not depend on the trajectory or initialization due to its uniqueness. This also saves memory, an advantage that convex problems have over generic optimization problems.

Time complexity. The forward pass (*i.e*, computation of Equation 4.4) using our approach requires the solution to the QP solver whose complexity scales as $O(d^3)$ where d is the number of optimization variables. This time is dominated by factorizing the KKT matrix required for primal-dual interior point method. Backward pass requires the solution to Equation 4.8 in Theorem 2, whose complexity is $O(d^2)$ given the factorization already computed in the forward pass. Both forward pass and backward pass can be expensive when the dimension of embedding f_ϕ is large.

Dual formulation. The dual formulation of the objective in Equation 4.4 allows us to address the poor dependence on the embedding dimension and can be written as follows. Let

$$\boldsymbol{w}_k(^k) = \sum_n \alpha_n^k f_\phi(\boldsymbol{x}_n) \quad \forall\, k. \tag{4.9}$$

We can instead optimize in the dual space:

$$\begin{aligned} &\max_{\{^k\}} \Big[-\frac{1}{2}\sum_k ||\boldsymbol{w}_k(^k)||_2^2 + \sum_n \alpha_n^{y_n} \Big] \\ &\text{subject to} \\ &\qquad \alpha_n^{y_n} \leq C, \quad \alpha_n^k \leq 0 \quad \forall k \neq y_n, \\ &\qquad \sum_k \alpha_n^k = 0 \quad \forall n. \end{aligned} \tag{4.10}$$

This results in a quadratic program (QP) over the dual variables $\{^k\}_{k=1}^K$. We note that the size of the optimization variable is the number of training examples times the number of classes. This is often *much smaller* than the size of the feature dimension for few-shot learning. We solve the dual QP of Equation 4.10 using [AK17] which implements a differentiable GPU-based QP solver. In practice (as seen in Table 4.3) the time taken by the QP solver is comparable to the time taken to compute features using the ResNet-12 architectures so the overall speed per iteration is not significantly different from those based on simple base learners such as nearest class prototype (mean) used in Prototypical Networks [SSZ17].

Concurrent to our work, Bertinetto *et al.* [BHTV19] employed ridge regression as the base learner which has a closed-form solution. Although ridge regression may not be best suited for classification problems, their work showed that training models by minimizing squared error with respect to one-hot labels works well in practice. The resulting optimization for ridge regression is also a QP and can be implemented within our framework as:

$$\max_{\{^k\}} \Big[-\frac{1}{2}\sum_k ||\boldsymbol{w}_k(^k)||_2^2 - \frac{\lambda}{2}\sum_k ||^k||_2^2 + 2\sum_n \alpha_n^{y_n} \Big] \tag{4.11}$$

where $\boldsymbol{w}_k$ is defined as Equation 4.9. A comparison of linear SVM and ridge regression in Section 4.4 shows a slight advantage of the linear SVM formation.

4.3.4 Meta-learning objective

To measure the performance of the model we evaluate the negative log-likelihood of the test data sampled from the same task. Hence, we can re-express the meta-learning objective of Equation 4.2 as:

$$\begin{aligned}&\mathcal{L}^{meta}(\mathcal{D}^{test};\theta,\phi,\gamma) = \\ &\sum_{(\boldsymbol{x},y)\in\mathcal{D}^{test}} [-\gamma \boldsymbol{w}_y \cdot f_\phi(\boldsymbol{x}) + \log \sum_k \exp(\gamma \boldsymbol{w}_k \cdot f_\phi(\boldsymbol{x}))]\end{aligned} \tag{4.12}$$

where $\theta = \mathcal{A}(\mathcal{D}^{train};\phi) = \{\boldsymbol{w}_k\}_{k=1}^K$ and γ is a learnable scale parameter. Prior work in few-shot learning [ORL18, BHTV19, GK18] suggest that adjusting the prediction score by a learnable scale parameter γ provides better performance under nearest class mean and ridge regression base learners.

We empirically find that inserting γ is beneficial for the meta-learning with SVM base learner as well. While other choices of test loss, such as hinge loss, are possible, log-likelihood worked the best in our experiments.

4.4 Experiments

We first describe the network architecture and optimization details used in our experiments (Section 4.4.1). We then present results on standard few-shot classification benchmarks including derivatives of ImageNet (Section 4.4.2) and CIFAR (Section 4.4.3), followed by a detailed analysis of the impact of various base learners on accuracy and speed using the same embedding network and training setup (Section 4.4.4-4.4.6).

4.4.1 Implementation details

Meta-learning setup. We use a ResNet-12 network following [ORL18, MRCA18] in our experiments. Let `Rk` denote a residual block that consists of three $\{3\times3$ convolution with `k` filters, batch normalization, Leaky ReLU(0.1)}; let `MP` denote a 2×2 max pooling. We use DropBlock regularization [GLL18], a form of structured Dropout. Let `DB(k, b)` denote a DropBlock layer with keep_rate=`k` and block_size=`b`. The network architecture for ImageNet derivatives is: `R64-MP-DB(0.9,1)-R160-MP-DB(0.9,1)-R320- MP-DB(0.9,5)-R640-MP-DB(0.9,5)`, while the network architecture used for CIFAR derivatives is: `R64-MP-DB(0.9,1)-R160-MP-DB(0.9,1) -R320-MP-DB(0.9,2)-R640-MP-DB(0.9,2)`. We do not apply a global average pooling after the last residual block.

As an optimizer, we use SGD with Nesterov momentum of 0.9 and weight decay of 0.0005. Each mini-batch consists of 8 episodes. The model was meta-trained for 60 epochs, with each epoch consisting of 1000 episodes. The learning rate was initially set to 0.1, and then changed to 0.006, 0.0012, and 0.00024 at epochs 20, 40 and 50, respectively, following the practice of [GK18].

During meta-training, we adopt horizontal flip, random crop, and color (brightness, contrast, and saturation) jitter data augmentation as in [GK18, QLSY18]. For experiments on miniImageNet with ResNet-12, we use label smoothing with $\varepsilon = 0.1$. Unlike [SSZ17] where they used higher way classification for meta-training than meta-testing, we use a 5-way classification in both stages following recent works [GK18, ORL18]. Each class contains 6 test (query) samples during meta-training and 15 test samples during meta-testing. Our meta-trained model was chosen based on 5-way 5-shot test accuracy on the meta-validation set.

Meta-training shot. For prototypical networks, we match the meta-training shot to meta-testing shot following the usual practice [SSZ17, GK18]. For SVM and ridge regression, we observe that keeping meta-training shot higher than meta-testing shot leads to better test accuracies as shown in Figure 4.2. Hence, during meta-training, we set training shot to 15 for miniImageNet with ResNet-12; 5 for miniImageNet with 4-layer CNN (in Table 4.3); 10 for tieredImageNet; 5 for CIFAR-FS; and 15 for FC100.

Base-learner setup. For linear classifier training, we use the quadratic programming

(QP) solver OptNet [AK17]. Regularization parameter C of SVM was set to 0.1. Regularization parameter λ of ridge regression was set to 50.0. For the nearest class mean (prototypical networks), we use squared Euclidean distance normalized with respect to the feature dimension.

Early stopping. Although we can run the optimizer until convergence, in practice we found that running the QP solver for a fixed number of iterations (just three) works well in practice. Early stopping acts an additional regularizer and even leads to a slightly better performance.

4.4.2 Experiments on ImageNet derivatives

The **miniImageNet** dataset [VBL$^+$16] is a standard benchmark for few-shot image classification benchmark, consisting of 100 randomly chosen classes from ILSVRC-2012 [RDS$^+$15]. These classes are randomly split into 64, 16 and 20 classes for meta-training, meta-validation, and meta-testing respectively. Each class contains 600 images of size 84×84. Since the class splits were not released in the original publication [VBL$^+$16], we use the commonly-used split proposed in [RL17].

The **tieredImageNet** benchmark [RRT$^+$18] is a larger subset of ILSVRC-2012 [RDS$^+$15], composed of 608 classes grouped into 34 high-level categories. These are divided into 20 categories for meta-training, 6 categories for meta-validation, and 8 categories for meta-testing. This corresponds to 351, 97 and 160 classes for meta-training, meta-validation, and meta-testing respectively. This dataset aims to minimize the semantic similarity between the splits. All images are of size 84×84.

Results. Table 4.1 summarizes the results on the 5-way miniImageNet and tieredImageNet. Our method achieves state-of-the-art performance on 5-way miniImageNet and tieredImageNet benchmarks. Note that LEO [RRS$^+$19] make use of encoder and relation network in addition to the WRN-28-10 backbone network to produce sample-dependent initialization of gradient descent. TADAM [ORL18] employs a task embedding network (TEN) block for each convolutional layer – which predicts element-wise scale and shift vectors.

Table 4.1: **Comparison to prior work on miniImageNet and tieredImageNet.** Average few-shot classification accuracies (%) with 95% confidence intervals on miniImageNet and tieredImageNet meta-test splits. a-b-c-d denotes a 4-layer convolutional network with a, b, c, and d filters in each layer. *Results from [RL17]. †Used the union of meta-training set and meta-validation set to meta-train the meta-learner. “RR" stands for ridge regression.

		miniImageNet 5-way		**tieredImageNet 5-way**	
model	**backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Meta-Learning LSTM* [RL17]	64-64-64-64	43.44 ± 0.77	60.60 ± 0.71	-	-
Matching Networks* [VBL$^+$16]	64-64-64-64	43.56 ± 0.84	55.31 ± 0.73	-	-
MAML [FAL17]	32-32-32-32	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	70.30 ± 1.75
Prototypical Networks*† [SSZ17]	64-64-64-64	49.42 ± 0.78	68.20 ± 0.66	53.31 ± 0.89	72.69 ± 0.74
Relation Networks* [SYZ$^+$18]	64-96-128-256	50.44 ± 0.82	65.32 ± 0.70	54.48 ± 0.93	71.32 ± 0.78
R2D2 [BHTV19]	96-192-384-512	51.2 ± 0.6	68.8 ± 0.1	-	-
Transductive Prop Nets [LLP$^+$19]	64-64-64-64	55.51 ± 0.86	69.86 ± 0.65	59.91 ± 0.94	73.30 ± 0.75
SNAIL [MRCA18]	ResNet-12	55.71 ± 0.99	68.88 ± 0.92	-	-
Dynamic Few-shot [GK18]	64-64-128-128	56.20 ± 0.86	73.00 ± 0.64	-	-
AdaResNet [MYMT18]	ResNet-12	56.88 ± 0.62	71.94 ± 0.57	-	-
TADAM [ORL18]	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	-	-
Activation to Parameter† [QLSY18]	WRN-28-10	59.60 ± 0.41	73.74 ± 0.19	-	-
LEO† [RRS$^+$19]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12	$\mathbf{66.33 \pm 0.05}$	$\mathbf{81.44 \pm 0.09}$
MetaOptNet-RR (ours)	ResNet-12	61.41 ± 0.61	77.88 ± 0.46	$\mathbf{65.36 \pm 0.71}$	$\mathbf{81.34 \pm 0.52}$
MetaOptNet-SVM (ours)	ResNet-12	62.64 ± 0.61	78.63 ± 0.46	$\mathbf{65.99 \pm 0.72}$	$\mathbf{81.56 \pm 0.53}$
MetaOptNet-SVM-trainval (ours)†	ResNet-12	$\mathbf{64.09 \pm 0.62}$	$\mathbf{80.00 \pm 0.45}$	$\mathbf{65.81 \pm 0.74}$	$\mathbf{81.75 \pm 0.53}$

We also note that [RRS$^+$19, QLSY18] pretrain the WRN-28-10 feature extractor [ZK16] to jointly classify all 64 classes in miniImageNet meta-training set; then freeze the network during the meta-training. [ORL18] make use of a similar strategy of using standard classification: they co-train the feature embedding on few-shot classification task (5-way) and standard classification task (64-way). In contrast, our system is meta-trained end-to-end, explicitly training the feature extractor to work well on few-shot learning tasks with regularized linear classifiers. This strategy allows us to clearly see the effect of meta-learning. Our method is arguably simpler and achieves strong performance.

4.4.3 Experiments on CIFAR derivatives

The **CIFAR-FS** dataset [BHTV19] is a recently proposed few-shot image classification benchmark, consisting of all 100 classes from CIFAR-100 [KNHb]. The classes are randomly split into 64, 16 and 20 for meta-training, meta-validation, and meta-testing respectively. Each class contains 600 images of size 32×32.

The **FC100** dataset [ORL18] is another dataset derived from CIFAR-100 [KNHb], containing 100 classes which are grouped into 20 superclasses. These classes are partitioned into 60 classes from 12 superclasses for meta-training, 20 classes from 4 superclasses for meta-validation, and 20 classes from 4 superclasses for meta-testing. The goal is to minimize semantic overlap between classes similar to the goal of tieredImageNet. Each class contains 600 images of size 32×32.

Results. Table 4.2 summarizes the results on the 5-way classification tasks where our method MetaOptNet-SVM achieves the state-of-the-art performance. On the harder FC100 dataset, the gap between various base learners is more significant, which highlights the advantage of complex base learners in the few-shot learning setting.

4.4.4 Comparisons between base learners

Table 4.3 shows the results where we vary the base learner for two different embedding architectures. When we use a standard 4-layer convolutional network where the feature dimension is low (1600), we do not observe a substantial benefit of adopting discriminative classifiers for few-shot learning. Indeed, nearest class mean classifier [MVPC13] is proven to work well under a low-dimensional feature as shown in Prototypical Networks [SSZ17]. However, when the embedding dimensional is much higher (16000), SVMs yield better few-shot accuracy than other base learners. Thus, regularized linear classifiers provide robustness when high-dimensional features are available.

Table 4.2: **Comparison to prior work on CIFAR-FS and FC100.** Average few-shot classification accuracies (%) with 95% confidence intervals on CIFAR-FS and FC100. a-b-c-d denotes a 4-layer convolutional network with a, b, c, and d filters in each layer. *CIFAR-FS results from [BHTV19]. †FC100 result from [ORL18]. ¶Used the union of meta-training set and meta-validation set to meta-train the meta-learner. "RR" stands for ridge regression.

		CIFAR-FS 5-way		FC100 5-way	
model	**backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
MAML* [FAL17]	32-32-32-32	58.9 ± 1.9	71.5 ± 1.0	-	-
Prototypical Networks*† [SSZ17]	64-64-64-64	55.5 ± 0.7	72.0 ± 0.6	35.3 ± 0.6	48.6 ± 0.6
Relation Networks* [SYZ+18]	64-96-128-256	55.0 ± 1.0	69.3 ± 0.8	-	-
R2D2 [BHTV19]	96-192-384-512	65.3 ± 0.2	79.4 ± 0.1	-	-
TADAM [ORL18]	ResNet-12	-	-	40.1 ± 0.4	56.1 ± 0.4
ProtoNets (our backbone) [SSZ17]	ResNet-12	**72.2** ± **0.7**	83.5 ± 0.5	37.5 ± 0.6	52.5 ± 0.6
MetaOptNet-RR (ours)	ResNet-12	**72.6** ± **0.7**	**84.3** ± **0.5**	40.5 ± 0.6	55.3 ± 0.6
MetaOptNet-SVM (ours)	ResNet-12	**72.0** ± **0.7**	**84.2** ± **0.5**	41.1 ± 0.6	55.5 ± 0.6
MetaOptNet-SVM-trainval (ours)¶	ResNet-12	**72.8** ± **0.7**	**85.0** ± **0.5**	**47.2** ± **0.6**	**62.5** ± **0.6**

The added benefits come at a modest increase in computational cost. For ResNet-12, compared to nearest class mean classifier, the additional overhead is around 13% for the ridge regression base learner and around 30-50% for the SVM base learner. As seen in from Figure 4.2, the performance of our model on both 1-shot and 5-shot regimes generally increases with increasing meta-training shot. This makes the approach more practical as we can meta-train the embedding once with a high shot for all meta-testing shots.

As noted in the FC100 experiment, SVM base learner seems to be beneficial when the semantic overlap between test and train is smaller. We hypothesize that the class embeddings are more significantly more compact for training data than test data (*e.g*, see [YCBL14]); hence flexibility in the base learner allows robustness to noisy embeddings and improves generalization.

4.4.5 Reducing meta-overfitting

Augmenting meta-training set. Despite sampling tasks, at the end of meta-training MetaOptNet-SVM with ResNet-12 achieves nearly 100% test accuracy on all the meta-training

Table 4.3: **Effect of the base learner and embedding network architecture.** Average few-shot classification accuracy (%) and forward inference time (ms) per episode on miniImageNet and tieredImageNet with varying base learner and backbone architecture. The former group of results used the standard 4-layer convolutional network with 64 filters per layer used in [VBL+16, SSZ17], whereas the latter used a 12-layer ResNet without the global average pooling. "RR" stands for ridge regression.

	miniImageNet 5-way				**tieredImageNet 5-way**			
	1-shot		**5-shot**		**1-shot**		**5-shot**	
model	**acc. (%)**	**time (ms)**	**acc. (%)**	**time (ms)**	**acc. (%)**	**time (ms)**	**acc. (%)**	**time (ms)**
4-layer conv (feature dimension=1600)								
Prototypical Networks [MVPC13, SSZ17]	53.47$_{\pm0.63}$	6$_{\pm0.01}$	70.68$_{\pm0.49}$	7$_{\pm0.02}$	54.28$_{\pm0.67}$	6$_{\pm0.03}$	71.42$_{\pm0.61}$	7$_{\pm0.02}$
MetaOptNet-RR (ours)	53.23$_{\pm0.59}$	20$_{\pm0.03}$	69.51$_{\pm0.48}$	27$_{\pm0.05}$	54.63$_{\pm0.67}$	21$_{\pm0.05}$	72.11$_{\pm0.59}$	28$_{\pm0.06}$
MetaOptNet-SVM (ours)	52.87$_{\pm0.57}$	28$_{\pm0.02}$	68.76$_{\pm0.48}$	37$_{\pm0.05}$	54.71$_{\pm0.67}$	28$_{\pm0.07}$	71.79$_{\pm0.59}$	38$_{\pm0.08}$
ResNet-12 (feature dimension=16000)								
Prototypical Networks [MVPC13, SSZ17]	59.25$_{\pm0.64}$	60$_{\pm17}$	75.60$_{\pm0.48}$	66$_{\pm17}$	61.74$_{\pm0.77}$	61$_{\pm17}$	80.00$_{\pm0.55}$	66$_{\pm18}$
MetaOptNet-RR (ours)	61.41$_{\pm0.61}$	68$_{\pm17}$	**77.88**$_{\pm0.46}$	75$_{\pm17}$	**65.36**$_{\pm0.71}$	69$_{\pm17}$	**81.34**$_{\pm0.52}$	77$_{\pm17}$
MetaOptNet-SVM (ours)	**62.64**$_{\pm0.61}$	78$_{\pm17}$	**78.63**$_{\pm0.46}$	89$_{\pm17}$	**65.99**$_{\pm0.72}$	78$_{\pm17}$	**81.56**$_{\pm0.53}$	90$_{\pm17}$

datasets except the tieredImageNet. To alleviate overfitting, similarly to [RRS+19, QLSY18], we use the union of the meta-training and meta-validation sets to meta-train the embedding, keeping the hyperparameters, such as the number of epochs, identical to the previous setting. In particular, we terminate the meta-training after 21 epochs for miniImageNet, 52 epochs for tieredImageNet, 21 epochs for CIFAR-FS, and 21 epochs for FC100. Tables 4.1 and 4.2 show the results with the augmented meta-training sets, denoted as MetaOptNet-SVM-trainval. On minImageNet, CIFAR-FS, and FC100 datasets, we observe improvements in test accuracies. On tieredImageNet dataset, the difference is negligible. We suspect that this is because our system has not yet entered the regime of overfitting (In fact, we observe $\sim$94% test accuracy on tieredImageNet meta-training set). Our results suggest that meta-learning embedding with more meta-training *"classes"* helps reduce overfitting to the meta-training set.

Various regularization techniques. Table 4.4 shows the effect of regularization methods on MetaOptNet-SVM with ResNet-12. We note that early works on few-shot learning [SSZ17, FAL17] did not employ any of these techniques. We observe that without the use of regularization,

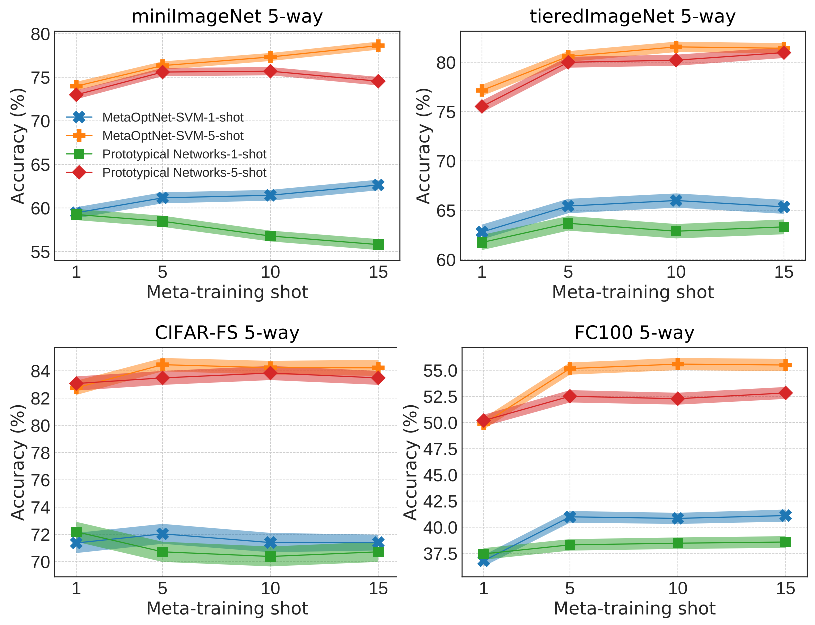


Figure 4.2: **Test accuracies (%) on meta-test sets with varying meta-training shot**. Shaded region denotes 95% confidence interval. In general, the performance of MetaOptNet-SVM on both 1-shot and 5-shot regimes increases with increasing meta-training shot.

the performance of ResNet-12 reduces to the one of the 4-layer convolutional network with 64 filters per layer shown in Table 4.3. This shows the importance of regularization for meta-learners. We expect that performances of few-shot learning systems would be further improved by introducing novel regularization methods.

4.4.6 Efficiency of dual optimization

To see whether the dual optimization is indeed effective and efficient, we measure accuracies on meta-test set with varying iteration of the QP solver. Each iteration of QP solver [AK17] involves computing updates for primal and dual variables via LU decomposition of KKT matrix. The results are shown in Figure 4.3. The QP solver reaches the optima of ridge regression objective in just one iteration. Alternatively one can use its closed-form solution as used in [BHTV19]. Also, we observe that for 1-shot tasks, the QP SVM solver reaches optimal accuracies in 1 iteration, although we observed that the KKT conditions are not exactly satisfied

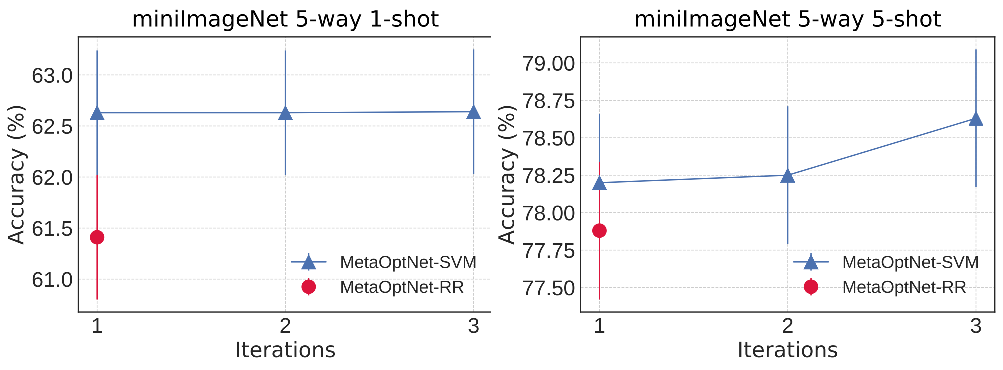


Figure 4.3: **Test accuracies (%) on miniImageNet meta-test set with varying iterations of QP solver.** The error bar denotes 95% confidence interval. Ridge regression base learner (MetaOptNet-RR) converges in 1 iteration; SVM base learner (MetaOptNet-SVM) was run for 3 iterations.

yet. For 5-shot tasks, even if we run QP SVM solver for 1 iteration, we achieve better accuracies than other base learners. When the iteration of SVM solver is limited to 1 iteration, 1 episode takes 69 ± 17 ms for an 1-shot task, and 80 ± 17 ms for a 5-shot task, which is on par with the computational cost of the ridge regression solver (Table 4.3). These experiments show that solving dual objectives for SVM and ridge regression is very effective under few-shot settings.

4.5 Conclusion

In this paper, we presented a meta-learning approach with convex base learners for few-shot learning. The dual formulation and KKT conditions can be exploited to enable computational and memory efficient meta-learning that is especially well-suited for few-shot learning problems. Linear classifiers offer better generalization than nearest-neighbor classifiers at a modest increase in computational costs (as seen in Table 4.3). Our experiments suggest that regularized linear models allow significantly higher embedding dimensions with reduced overfitting. For future work, we aim to explore other convex base-learners such as kernel SVMs. This would allow the ability to incrementally increase model capacity as more training data becomes available for a task.

Table 4.4: **Ablation study.** Various regularization techniques improves test accuracy (%) on 5-way miniImageNet benchmark. We use MetaOptNet-SVM with ResNet-12 for results. 'Data Aug.', 'Label Smt.', and 'Larger Data' stand for data augmentation, label smoothing on the meta-learning objective, and merged dataset of meta-training split and meta-test split, respectively.

Data Aug.	Weight Decay	Drop Block	Label Smt.	Larger Data	1-shot	5-shot
					51.13	70.88
✓					55.80	75.76
	✓				56.65	73.72
✓	✓				60.33	76.61
✓	✓	✓			61.11	77.40
✓	✓	✓	✓		62.64	78.63
✓	✓	✓	✓	✓	64.09	80.00

4.6 Acknowledgements

This chapter, in full, is a reprint of the material as it appears in "Meta-Learning with Differentiable Convex Optimization," Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, Stefano Soatto, Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2019. The dissertation author is the primary investigator and author of this material.

Chapter 5

Conclusion

In this dissertation, I introduced approaches for learning priors on visual data and models. First, I presented approached for generative modeling of visual data: Wasserstein introspective neural networks and ViTGAN. In Chapter 2, we developed Wasserstein introspective neural networks (WINN) that significantly improves training stability of prior energy-based models [JLT17]. WINN is the first approach to produce near state-of-the-art as a generator and a discriminative classifier at the same time. Our models' significant error reduction against adversarial examples suggest that learning priors on natural images is beneficial for discriminative classifiers. In Chapter 3, we have introduced ViTGAN, leveraging Vision Transformers (ViTs) in GANs. We developed essential techniques to improving its training stability and convergence. Our experiments on standard benchmarks demonstrate that the presented model achieves comparable performance to leading CNN-based GANs. Our method is the first one to show that generic ViTs without image-specific architectural priors such as convolutions or pooling can be used for learning image priors. Our paper establishes a concrete baseline of ViT on resolutions up to 128×128 and we hope that it can facilitate future research to use vision transformers for high-resolution image generation.

Second, I presented an approach for learning priors for visual learners. In Chapter 4, we developed a meta-learning approach with convex base learners for few-shot learning. We showed the dual formulation and KKT conditions can be used to enable computational and memory efficient meta-learning that is especially well-suited for few-shot learning problems. Linear classifiers offer better generalization than nearest-neighbor classifiers at a modest increase in computational costs. Our experiments suggest that regularized linear models allow significantly higher embedding dimensions with reduced overfitting. After our work was published, [RFKL19, LVD19] discovered that implicit differentiation can be used for general meta-learning and hyperparameter optimization.

Bibliography

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[ADH$^+$21] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021.

[ADK$^+$21] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *CVPR*, 2021.

[AK17] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017.

[Bal12] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49, 2012.

[Bar18] Shane Barratt. On the Differentiability of the Solution to Convex Optimization Problems. arXiv:1804.05098, 2018.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019.

[BHTV19] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

[BKH16] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[BKT$^+$20] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward transformer-based object detection. *arXiv preprint arXiv:2012.09958*, 2020.

[BMR$^+$20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[BWT21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021.

[CDS19] Aidan Clark, Jeff Donahue, and Karen Simonyan. Efficient video generation on complex datasets. *CoRR*, abs/1907.06571, 2019.

[CKY08] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *ICML*, 2008.

[CLHG19] Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. In *ICLR*, 2019.

[CRC$^+$20] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

[CS02] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002.

[CWG$^+$20] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *ICML*, 2020.

[CXH21] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *CoRR*, abs/2104.02057, 2021.

[DBK$^+$21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[DBP$^+$17] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

[DCL21] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*, 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

[DF18] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018.

[Dom12] Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, 2012.

[DR09] Asen L. Dontchev and R. Tyrrell Rockafellar. Implicit functions and solution mappings. *Springer Monogr. Math.*, 2009.

[DSB15] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[ERO21] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[FRL$^+$18] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M. Dai, Shakir Mohamed, and Ian J. Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. In *ICLR*, 2018.

[FS97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and Sys. Sci.*, 55(1), 1997.

[GAA$^+$17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[GEB15a] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[GEB15b] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[GFC$^+$16] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

[GHW$^+$21] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers, 2021.

[GK18] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.

[GLL18] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, 2018.

[GPAM$^+$14a] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[GPAM$^+$14b] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[GSS15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[HLP$^+$17] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *CVPR*, 2017.

[HLvdMW17] Gao Huang*, Zhuang Liu*, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[HLZW17] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *AAAI*, 2017.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[HRU$^+$17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[HS86] G. E. Hinton and T. Sejnowski. Learning and relearning in boltzmann machines. In *Parallel distributed processing: Explorations in the microstructure of cognition*, pages 282–317–. MIT Press, Cambridge, MA, 1986.

[HSA84] Geoffrey E. Hinton, Terrence J. Sejnowski, and David H. Ackley. Boltzmann machines: Constraint satisfaction networks that learn. Technical Report CMU-CS-84-119, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1984.

[HZ21] Drew A Hudson and C. Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint:2103.01209*, 2021.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[JCW21] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 2021.

[JLT17] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classification with convolutional nets. In *NIPS*, 2017.

[JS21] Jongheon Jeong and Jinwoo Shin. Training {gan}s with stronger augmentations via contrastive discriminator. In *ICLR*, 2021.

[KAH$^+$20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020.

[KAHK17] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[KALL18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[KGB17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017.

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[KLA$^+$20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

[KNHa] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[KNHb] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[KP12] Steven G. Krantz and Harold R. Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

[KPM21] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *ICML*, 2021.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[KW14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[KWKT15] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[LBC17] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and convergence properties of generative adversarial learning. In *NIPS*, 2017.

[LBD$^+$89a] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[LBD$^+$89b] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.

[LJT17] Justin Lazarow*, Long Jin*, and Zhuowen Tu. Introspective neural networks for generative modeling. In *ICCV*, 2017.

[LLC$^+$21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[LLP$^+$19] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. In *ICLR*, 2019.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[LVD19] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. 2019.

[LXG$^+$15] Chen-Yu Lee*, Saining Xie*, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[MDA15] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

[MGE11] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[MNG18] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *ICML*, 2018.

[MON$^+$19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

[MRCA18] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[Mül97] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, pages 429–443, 1997.

[MVPC13] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriella Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2624–2637, November 2013.

[MYMT18] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *ICML*, 2018.

[NCKN11] Jiquan Ngiam, Zhenghao Chen, Pang Wei Koh, and Andrew Y. Ng. Learning deep energy models. In *ICML*, 2011.

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *NeurIPS*, 2016.

[NWC$^+$11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[ORL18] Boris N. Oreshkin, Pau Rodríguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

[PFS$^+$19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[Pin60] Mark S Pinsker. Information and information stability of random variables and processes. 1960.

[QLSY18] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L. Yuille. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 2018.

[RBK21] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[RDS$^+$15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015.

[RFKL19] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

[RL17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[RMC16a] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[RMC16b] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[RPG$^+$21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

[RPV$^+$19] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[RRS$^+$19] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

[RRT$^+$18] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

[Sch87] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987.

[SE20] Jiaming Song and Stefano Ermon. Bridging the gap between f-gans and wasserstein gans. In *ICML*, 2020.

[SGLS21] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021.

[SGZ$^+$16a] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[SGZ$^+$16b] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[SMB+20] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

[SR14] Uwe Schmidt and Stefan Roth. Shrinkage fields for effective image restoration. In *CVPR*, 2014.

[SSK20] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *CVPR*, 2020.

[SSZ17] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[SV16] Igal Sason and Sergio Verdú. f-divergence inequalities. *IEEE Transactions on Information Theory*, 62(11):5973–6006, 2016.

[SYZ+18] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[TCD+20] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[THK+21] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[Thr98] Sebastian Thrun. *Lifelong Learning Algorithms*, pages 181–209. Springer US, Boston, MA, 1998.

[TJL+21] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *CVPR*, 2021.

[TLAF07] Marshall F. Tappen, Ce Liu, Edward H. Adelson, and William T. Freeman. Learning gaussian conditional random fields for low-level vision. In *CVPR*, 2007.

[TLYK18] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *CVPR*, 2018.

[TSM+20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.

[TTN$^+$21] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021.

[Tu07] Zhuowen Tu. Learning generative models via discriminative approaches. In *CVPR*, 2007.

[ULVL16] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[VBL$^+$16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[VD02] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, Jun 2002.

[VSP$^+$17] 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.

[WFB17] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *ICLR*, 2017.

[WT11] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[WW99] Jason Weston and Chris Watkins. Support Vector Machines for Multiclass Pattern Recognition. In *European Symposium On Artificial Neural Networks*, 1999.

[WXC$^+$21] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[WZH02] Max Welling, Richard S Zemel, and Geoffrey E Hinton. Self supervised boosting. In *NIPS*, 2002.

[XGD$^+$17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[XLG$^+$18] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *AAAI*, 2018.

[XLZW16] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *ICML*, 2016.

[XSM$^+$21] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better, 2021.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[YZS$^+$15] 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 preprint arXiv:1506.03365*, 2015.

[ZGMO19] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[ZLL$^+$20] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *NeurIPS*, 2020.

[ZLS$^+$19] Zhiming Zhou, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang. Lipschitz generative adversarial nets. In *ICML*, 2019.

[ZSL$^+$21] Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved consistency regularization for gans. In *AAAI*, 2021.

[ZZC$^+$20] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for GAN training. *arXiv preprint:2006.02595*, 2020.

[ZZOL20] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *ICLR*, 2020.