ENHANCING GAN PERFORMANCE THROUGH NEURAL ARCHITECTURE SEARCH AND TENSOR DECOMPOSITION

Prasanna Reddy Pulakurthi[1], *Mahsa Mozaffari*[1], *Sohail A. Dianat*[1], *Majid Rabbani*[1], *Jamison Heard*[1], and *Raghuveer Rao*[2]

[1]Rochester Institute of Technology, USA
[2]DEVCOM Army Research Laboratory, USA

ABSTRACT

Generative Adversarial Networks (GANs) have emerged as a powerful tool for generating high-fidelity content. This paper presents a new training procedure that leverages Neural Architecture Search (NAS) to discover the optimal architecture for image generation while employing the Maximum Mean Discrepancy (MMD) repulsive loss for adversarial training. Moreover, the generator network is compressed using tensor decomposition to reduce its computational footprint and inference time while preserving its generative performance. Experimental results show improvements of **34%** and **28%** in the FID score on the CIFAR-10 and STL-10 datasets, respectively, with corresponding footprint reductions of **14×** and **31×** compared to the best FID score method reported in the literature. The implementation code is available at: *https://github.com/PrasannaPulakurthi/MMD-AdversarialNAS*.

***Index Terms*—** Neural Architecture Search, Maximum Mean Discrepancy, Generative Adversarial Networks

1. INTRODUCTION

Generative Adversarial Networks (GANs) are a branch of generative models that can generate high-quality samples from a desired distribution. They have been successfully applied in the field of computer vision [1] and deep learning [2] to various tasks such as image-to-image translation [3], text-to-image synthesis [4], human pose synthesis [5], image generation [6], and speech synthesis [7].

The GAN generative performance is mainly influenced by the network architecture and the loss function. This work proposes a new training procedure that combines the advantages of the MMD-GAN repulsive loss [8] and Adversarial Neural Architecture Search (AdversarialNAS) [9]. The MMD-GAN repulsive loss proves superior due to its ability to account for the differences among real images during discriminator training, which enables the generator to capture intricate details more effectively. The AdversarialNAS uses a differentiable approach to simultaneously search for the optimal generator and discriminator architectures. The optimal architecture is trained using the MMD-GAN repulsive loss function along with our proposed training procedure, resulting in a significantly enhanced generative performance compared to the best reported in the literature. However, it also comes with a larger footprint. Therefore, tensor decomposition-based techniques are applied to reduce the footprint of the generator.

Our contributions are two-fold: **First**, the **MMD-GAN repulsive loss** is integrated into the **AdversarialNAS** framework, and a new **training procedure** is introduced that achieves state-of-the-art image generation performance using GANs. **Second**, a **GAN compression** technique based on **tensor decomposition** is implemented to reduce the generator's footprint while preserving its image generation capabilities through fine-tuning. The proposed method is applied to unsupervised image generation tasks using standard datasets such as CIFAR-10 and STL-10. Experimental results on widely accepted quantitative performance metrics like Inception Score (IS) [10] and Fréchet Inception Distance (FID) [11] show that our method substantially outperforms state-of-the-art techniques (both with uncompressed and compressed networks). Our tensor decomposition-based method achieves **3.40×** compression on the **CIFAR-10** and **3.64×** compression on the **STL-10** datasets, while significantly outperforming the state-of-the-art GAN techniques for image generation.

2. RELATED WORK

Training Loss Function: Various loss functions have been proposed in the literature such as minimax loss, non-saturation loss [12], hinge loss [13], Wasserstein loss [14, 15], and Maximum Mean Discrepancy (MMD) [16, 17] loss. Notably, the MMD-GAN repulsive loss function [8] has gained attention due to its ability to actively learn the differences among the real data, thus enabling the generator to learn these differences and generate a diverse set of high-quality images.

Network Architecture: Neural Architecture Search (NAS) has been investigated in literature due to its ability to automatically search for the optimal architecture. Different

This research was funded by the DEVCOM Army Research Laboratory under contract W911QX-21-D-0001.

NAS approaches employ various strategies for architecture optimization. For instance, AutoGAN [18] and AGAN [19] utilize metrics such as FID or IS as rewards in a reinforcement learning framework to guide architecture updates. AdversarialNAS [9] introduces a differentiable approach to simultaneously search for generator and discriminator architectures. However, NAS techniques primarily rely on minimax or hinge loss for searching and training the GAN networks.

GAN Compression: Recently, a growing interest in the compression of GANs has emerged [20, 21, 22, 23, 24, 25], and various strategies such as knowledge distillation [22, 23], pruning [20], and the Lottery Ticket Hypothesis (LTH) [24] have been employed. This is primarily driven by the substantial memory and computational demands associated with state-of-the-art GAN models. Tensor decomposition [26], a technique that has gained significant traction in recent years, has proven highly effective for compressing neural networks [27, 28, 29, 30]. This method has been predominantly employed in the context of Convolutional Neural Networks (CNNs), and its utility within the domain of GANs designed for image generation remains largely unexplored.

3. METHOD

This section introduces the training procedure, the loss function, and the tensor decomposition method. Consider the given data with N samples represented as $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the i^{th} sample and $\boldsymbol{X}$ has a distribution $P_{\boldsymbol{X}}$. The samples $\boldsymbol{Z}$ from a fixed distribution $P_{\boldsymbol{Z}}$ are input to the generator ($\boldsymbol{G}$) to obtain the output samples $\boldsymbol{Y} = \boldsymbol{G}(\boldsymbol{Z})$, which follows a distribution $P_{\boldsymbol{Y}}$. The generator is trained to generate output samples that mimic the real data distribution $P_{\boldsymbol{X}}$, that is, the generated distribution $P_{\boldsymbol{Y}} \approx P_{\boldsymbol{X}}$. The GAN [12] consists of two networks: the generator network ($\boldsymbol{G}$) and the discriminator network ($\boldsymbol{D}$). The learning is done by a two-player optimization game in which the generator aims to generate adversarial samples that fool the discriminator into thinking they are real, and the discriminator tries to distinguish between the real and fake samples.

3.1. Training Loss

To train the discriminator, we choose the MMD-GAN repulsive loss [8] given by,

$$\min_{\boldsymbol{D}} L_{\boldsymbol{D}} = \mathbb{E}_{P_{\boldsymbol{X}}}[k_{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{x}')] - \mathbb{E}_{P_{\boldsymbol{Y}}}[k_{\boldsymbol{D}}(\boldsymbol{y},\boldsymbol{y}')] \qquad (1)$$

The discriminator loss contracts the discriminator output of the generated samples $\boldsymbol{D}(\boldsymbol{Y})$ by maximizing $\mathbb{E}_{P_{\boldsymbol{Y}}}[k_{\boldsymbol{D}}(\boldsymbol{y},\boldsymbol{y}')]$ and expands the discriminator output of the real samples $\boldsymbol{D}(\boldsymbol{X})$ by minimizing $\mathbb{E}_{P_{\boldsymbol{X}}}[k_{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{x}')]$. The generator loss aims to minimize the Maximum Mean Discrepancy (MMD) between the generated $P_{\boldsymbol{Y}}$ and real distribution $P_{\boldsymbol{X}}$, given by,

$$\min_{\boldsymbol{G}} L_{\boldsymbol{G}} = M^2_{k_{\boldsymbol{D}}}(P_{\boldsymbol{X}}, P_{\boldsymbol{Y}}) = \mathbb{E}_{P_{\boldsymbol{X}}}[k_{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{x}')] + \mathbb{E}_{P_{\boldsymbol{Y}}}[k_{\boldsymbol{D}}(\boldsymbol{y},\boldsymbol{y}')] - 2\mathbb{E}_{P_{\boldsymbol{X}},P_{\boldsymbol{Y}}}[k_{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{y})] \qquad (2)$$

where $k_{\boldsymbol{D}}(\boldsymbol{a},\boldsymbol{b})$ is a characteristic kernel that measures the similarity between two samples $\boldsymbol{a}$ and $\boldsymbol{b}$ [31]. To stabilize MMD-GAN training, the authors in [8] propose the bounded Gaussian kernel for $\boldsymbol{D}(\boldsymbol{X})$ and $\boldsymbol{D}(\boldsymbol{Y})$:

$$k_{\boldsymbol{D}}(\boldsymbol{a},\boldsymbol{b}) = \begin{cases} \exp(-\frac{1}{2\sigma^2}\max(\|\boldsymbol{a}-\boldsymbol{b}\|^2, b_l)) & \boldsymbol{a},\boldsymbol{b} \in \boldsymbol{D}(\boldsymbol{Y}) \\ \exp(-\frac{1}{2\sigma^2}\min(\|\boldsymbol{a}-\boldsymbol{b}\|^2, b_u)) & \boldsymbol{a},\boldsymbol{b} \in \boldsymbol{D}(\boldsymbol{X}) \end{cases}$$

where b_l and b_u are the lower and upper bounds, and in [8], the values are set to σ = 1, b_l = 0.25, and b_u = 4. The authors in [8] defer the fine-tuning of these parameters to future work. Our extensive simulations, summarized in Table 1, indicate that the GAN generative performance is increased by progressively increasing the value of b_u during training. In our training procedure outlined in Algorithm 1, we start with a b_u of 4 and increase it if no further improvement in the FID score is observed within a fixed number of epochs.

Table 1. Ablation study comparing fixed upper bound training versus our proposed training procedure.

Method	b_u	IS ↑	FID ↓
Fixed	32	$9.60_{\pm 0.14}$	7.06
Fixed	4	$9.80_{\pm 0.12}$	6.52
Proposed	4 to 32	$9.92_{\pm 0.07}$	**6.08**

Algorithm 1 Proposed training procedure

Input: B batch size, N_E number of epoch. Initialize generator parameters $\boldsymbol{G}$ and discriminator parameters $\boldsymbol{D}$ using Xavier initialization [32]. Initialize the upper bound $b_u = 4$, $reset = 30$, $counter = 30$, and initial $best\_FID = 1000$.

```
while $epoch < N_E$ do
    1. Sample a mini-batch $\{\boldsymbol{x}_i\}_{i=1}^B$ from data distribution $P_X$ and $\{\boldsymbol{z}_i\}_{i=1}^B$ from noise distribution $P_{\boldsymbol{Z}}$.
    2. Update the generator weights by Adam optimizer using the gradient: $\nabla(\boldsymbol{G}) \leftarrow \nabla_{\boldsymbol{G}} L_{\boldsymbol{G}}$
    3. Update the discriminator weights by Adam optimizer using the gradient: $\nabla(\boldsymbol{D}) \leftarrow \nabla_{\boldsymbol{D}} L_{\boldsymbol{D}}$
    4. Sample 50K images to compute $FID$ and $IS$ metrics.
    if $FID < best\_FID$ then
        $best\_FID \leftarrow FID$
        $counter \leftarrow reset$
    else
        $counter \leftarrow counter - 1$
    end
    if $counter == 0$ then
        $b_u \leftarrow b_u \times 2$
        $counter \leftarrow reset$
    end
    $epoch \leftarrow epoch + 1$
end
```

3.2. Network Architecture

The MMD-GAN repulsive loss [8] is incorporated into the AdversarialNAS [9] framework to search for the optimal architecture, which is shown in Figure 1. The deeper layers

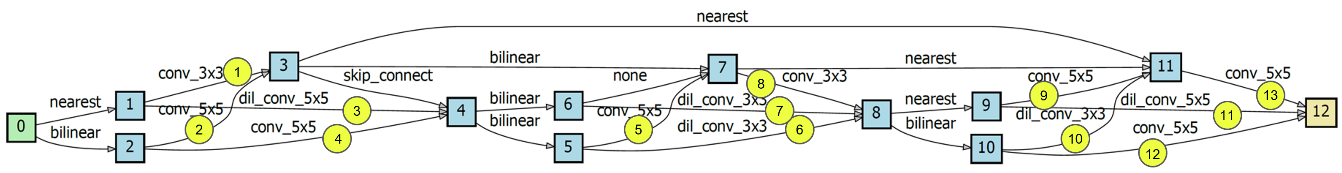

Fig. 1. The optimal generator architecture discovered by MMD-NAS for CIFAR-10 dataset.

tend to have denser connections, and the majority of them have a larger kernel size of 5×5, which underscores their importance for generating fine details in the image. Therefore, compressing the deeper layers is expected to introduce a larger performance degradation compared to the initial layers.

3.3. GAN Compression

After training the selected GAN architecture according to Algorithm 1, a tensor compression technique is applied to reduce the footprint of the generator while preserving its generative capabilities. The compression relies on Canonical Polyadic Decomposition [26] (CPD) and entails approximating the convolutional layer weights with a low-rank weight. Although compression inevitably results in information loss within the network, the network performance can be regained substantially by fine-tuning the compressed parameters.

Convolution and Fully Connected Layer Compression via CPD: Each convolutional layer within the generator can be described as a 4-dimensional tensor $\mathcal{W} \in \mathbb{R}^{T \times C \times H \times W}$, where T and C represent the number of output and input channels, while H and W are the height and width of the convolution kernel. Given a 3D input tensor $\mathcal{Y}$ to a convolution layer, the output tensor $\mathcal{Z}$ is calculated by convolving $\mathcal{Y}$ with $\mathcal{W}$. A rank-R CPD is employed to approximate the weight tensor, with a low-rank tensor in the CP format, given by, $\mathcal{W}_{t,c,h,w} \approx \sum_{r=1}^{R} \mathbf{A}_{t,r}\mathbf{B}_{c,r}\mathbf{C}_{h,r}\mathbf{D}_{w,r}$, where $\mathbf{A} \in \mathbb{R}^{T \times R}$, $\mathbf{B} \in \mathbb{R}^{C \times R}$, $\mathbf{C} \in \mathbb{R}^{H \times R}$ and $\mathbf{D} \in \mathbb{R}^{W \times R}$, are the factor matrices obtained by the CPD. Decomposing the weight tensor according to CPD allows for replacing the convolution layer with a sequence of four smaller convolution layers, as shown in Figure 2. The bias is integrated into the final layer within this sequence. Similarly, the fully connected layer is replaced with a sequence of two smaller fully connected layers, whose weights are the factor matrices computed by CPD.

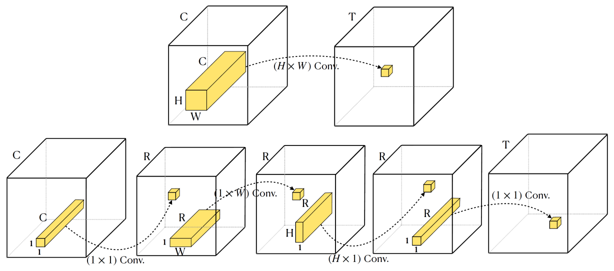

Fig. 2. a) Convolution operation (top). b) Compressed convolution operation by CPD (bottom).

Compression of the Generator and Selection Criteria: To compress the generator network, CPD is applied to specific layers of the generator. The sensitivity of each layer to compression, as measured by the FID score, is analyzed independently for each generator layer, as shown in Figure 3, and CPD is applied only to those layers with little performance loss. Our analysis across all datasets confirms our earlier conjecture that the compression of deeper layers (9 and 13, colored orange in Figure 3) results in a larger performance degradation relative to the remaining layers. These results may vary for different architectures found using NAS. As a final step, the fully connected layers are compressed without any fine-tuning because of stability issues. Also, the first fully connected layer is left uncompressed as it leads to major performance degradation.

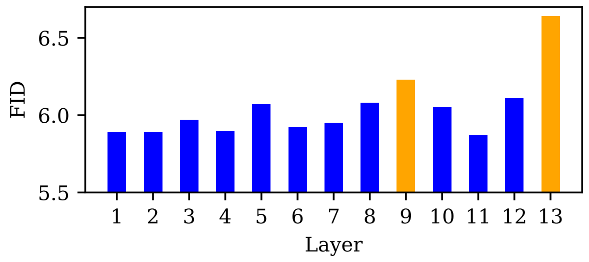

Fig. 3. FID of single layer compression without fine-tuning.

4. EXPERIMENTS

Datasets: Our method is evaluated using the CIFAR-10 [37] (50K images, 32×32 pixels), and the STL-10 [38] (100K images, 48×48 pixels) datasets. All the images were rescaled to a range [-1,1]. The images were generated from random noise samples from the normal distribution.

Networks: Two networks (large and small) are introduced, which are created by different numbers of generator channels, the large network with 256 channels and the small network with 128 channels. The number of channels for the discriminator is fixed at 512. These networks are compressed, resulting in a total of four networks.

Hyper-Parameters: The Adam [39] optimizer is used for our model, and the gradient decay factor is set to 0, squared gradient decay factor to 0.9, learning rate to 0.0002, and number of epochs to 500. The chosen Adam parameters are widely used in the literature and have shown to be effective in a wide range of optimization tasks, including image generation.

Evaluation metrics: FID [11] and IS [10] are used for quantitative evaluation using 50K real and 50k generated images.

Table 2. The quantitative comparisons of various methods and datasets for image generation tasks against state-of-the-art approaches, using IS and FID as evaluation metrics.

Method		CIFAR-10			STL10		
		Size (MB)	IS ↑	FID ↓	Size (MB)	IS ↑	FID ↓
SN-GAN [33]		4.3	$8.22_{\pm0.05}$	21.7	-	$9.16_{\pm0.12}$	40.1
AGAN [19]		20.1	$8.29_{\pm0.09}$	30.5	-	$9.23_{\pm0.08}$	52.7
Improved MMD-GAN [8]		-	8.29	16.21	-	9.34	37.63
AutoGAN [18]		4.4	$8.55_{\pm0.10}$	12.42	5.06	$9.16_{\pm0.12}$	31.01
AdversarialNAS [9]		8.84	$8.74_{\pm0.07}$	10.87	8.88	$9.63_{\pm0.19}$	26.98
Progressive GAN [34]		-	$8.80_{\pm0.05}$	15.52	-	-	-
StyleGAN-V2 [35]		-	9.18	11.07	-	10.21	20.84
TransGAN [36]		83.06	$9.02_{\pm0.12}$	9.26	167.85	$10.43_{\pm0.16}$	18.28
Our	**Large**	20.18	$\mathbf{9.62}_{\pm0.13}$	**5.83**	19.47	$\mathbf{11.60}_{\pm0.10}$	**12.91**
	Compressed Large	5.92	$\mathbf{9.66}_{\pm0.10}$	**6.09**	5.35	$\mathbf{11.28}_{\pm0.15}$	**13.06**
	Small	5.93	$\mathbf{9.32}_{\pm0.15}$	**7.48**	5.57	$\mathbf{11.39}_{\pm0.19}$	**14.57**
	Compressed Small	**1.63**	$\mathbf{9.38}_{\pm0.10}$	**8.18**	**1.71**	$\mathbf{11.66}_{\pm0.20}$	**14.84**

Compression: The rank R, which affects the degree of compression, is maintained across all layers of the network. In the CIFAR-10 dataset, R is set to 256 and 128 for compressing the large and small networks, respectively. In the STL-10 dataset, R is set to 128. The rank R for the fully connected layers is set to 4. Tensorly software package [40] is used for applying CPD to the layer weights.

Fine-tuning: After compression, the generator architecture is fine-tuned with the MMD-GAN repulsive loss function. To ensure the stability of the fine-tuning, a low learning rate of 10^{-5} is used for the new compressed layers. The learning rate and states of the Adam optimizer are preserved for the uncompressed generator layers and the discriminator.

5. RESULTS

CIFAR-10 dataset: The quantitative comparisons with previous state-of-the-art methods are given in Table 2. From the table it is observed that our method, with the large network of size 20.1 MB, achieves an FID of **5.83** and an IS of **9.62 ± 0.13** (SD), outperforming all the previous methods, including large transformer-based generative model [36] which has **4.2**× the footprint. Our compressed small network of size **1.63** MB outperforms all previous methods with an FID of **8.18** and an IS of **9.38 ± 0.10** while having the smallest footprint. Additionally, the 25 images shown in Figure 4a, which are randomly generated (without cherry-picking) by the large network, demonstrate the ability of our method to generate a diverse set of images with no sign of model collapse.

Transferability to the STL-10 dataset: The same optimal generator architecture searched using the CIFAR-10 dataset can be used to generate images from different datasets as is evidenced by the quantitative results in Table 2. The large generator network of size 19.47 MB achieves an FID of **12.91** and an IS of **11.60 ± 0.10**. The compressed small network achieves an FID of **14.84**, which is still **18.8%** better than the best result reported in the literature [36], while having a **98**× smaller footprint. The qualitative results of the large generator network for the STL-10 dataset are shown in Figure 4b.

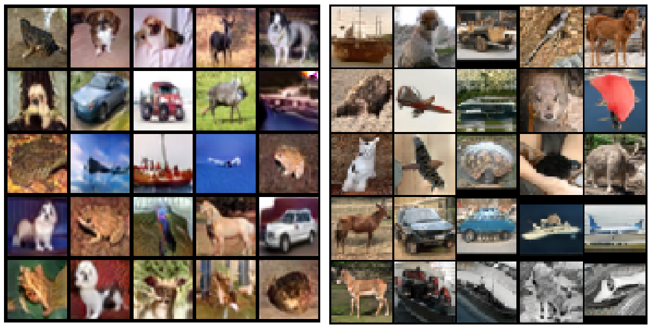
Fig. 4. a) The CIFAR-10 generated images by the discovered generator (left). b) The STL-10 generated images with the discovered generator using CIFAR-10 (right).

6. CONCLUSION

In this work, a novel approach has been presented that combines the advantages of MMD-GAN repulsive loss for capturing intricate details, the capabilities of Neural Architecture Search (NAS) to discover optimal architectures, and the advantages of tensor decomposition to reduce the network size while preserving its performance and allowing it to be deployed on edge devices. The proposed method significantly enhances image generation performance compared to the state-of-the-art, as measured by quantitative metrics such as FID and IS, with only a fraction of the footprint. As an example, on the CIFAR-10 dataset, the proposed technique achieves a **34%** improvement in FID score compared to the state-of-the-art [36] with a **14**× reduction in network size.

7. REFERENCES

[1] Zhengwei Wang, Qi She, and Tomas E Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[2] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3313–3332, 2023.

[3] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu, "Contrastive learning for unpaired image-to-image translation," in *Computer Vision–ECCV 2020*. Springer, 2020, pp. 319–345.

[4] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *2018 IEEE/CVF Conference on CVPR*, 2018, pp. 1316–1324.

[5] Wen Liu, Zhixin Piao, Zhi Tu, Wenhan Luo, Lin Ma, and Shenghua Gao, "Liquid warping gan with attention: A unified framework for human image synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[6] Minguk Kang and Jaesik Park, "Contragan: Contrastive learning for conditional image generation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21357–21369, 2020.

[7] Matthew Baas and Herman Kamper, "Gan you hear me? reclaiming unconditional speech synthesis from diffusion models," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023.

[8] Wei Wang, Yuan Sun, and Saman K. Halgamuge, "Improving MMD-GAN training with repulsive loss function," *CoRR*, vol. abs/1812.09916, 2018.

[9] Chen Gao, Yunpeng Chen, Si Liu, Zhenxiong Tan, and Shuicheng Yan, "Adversarialnas: Adversarial neural architecture search for gans," in *Proceedings of the IEEE/CVF Conference on CVPR*, 2020.

[10] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[11] 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," *Advances in neural information processing systems*, vol. 30, 2017.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial networks," 2014.

[13] Dustin Tran, Rajesh Ranganath, and David Blei, "Hierarchical implicit models and likelihood-free variational inference," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[15] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[16] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos, "Mmd gan: Towards deeper understanding of moment matching network," *Advances in neural information processing systems*, vol. 30, 2017.

[17] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton, "Demystifying MMD GANs," in *ICLR*, 2018.

[18] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang, "Autogan: Neural architecture search for generative adversarial networks," in *Proceedings of the IEEE/CVF ICCV*, 2019, pp. 3224–3234.

[19] Hanchao Wang and Jun Huan, "Agan: Towards automated design of generative adversarial networks," *arXiv:1906.11080*, 2019.

[20] Yuchen Liu, Zhixin Shu, Yijun Li, Zhe Lin, Federico Perazzi, and Sun-Yuan Kung, "Content-aware gan compression," in *Proceedings of the IEEE/CVF Conference on CVPR*, 2021, pp. 12156–12166.

[21] Shaojie Li, Jie Wu, Xuefeng Xiao, Fei Chao, Xudong Mao, and Rongrong Ji, "Revisiting discriminator in gan compression: A generator-discriminator cooperative compression scheme," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28560–28572, 2021.

[22] Yuxi Ren, Jie Wu, Xuefeng Xiao, and Jianchao Yang, "Online multi-granularity distillation for gan compression," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[23] Tie Hu, Mingbao Lin, Lizhou You, Fei Chao, and Rongrong Ji, "Discriminator-cooperated feature map distillation for gan compression," in *Proceedings of the IEEE/CVF Conference on CVPR*, 2023.

[24] Neha Mukund Kalibhat, Yogesh Balaji, and Soheil Feizi, "Winning lottery tickets in deep generative models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 8038–8046.

[25] Xuxi Chen, Zhenyu Zhang, Yongduo Sui, and Tianlong Chen, "Gans can play lottery tickets too," *arXiv preprint arXiv:2106.00134*, 2021.

[26] Tamara G Kolda and Brett W Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[27] Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki, "Stable low-rank tensor decomposition for compression of convolutional neural network," in *CV–ECCV*, 2020.

[28] Miao Yin, Yang Sui, Siyu Liao, and Bo Yuan, "Towards efficient tensor decomposition-based dnn model compression with optimization framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10674–10683.

[29] Weize Sun, Shaowu Chen, Lei Huang, Hing Cheung So, and Min Xie, "Deep convolutional neural network compression via coupled tensor decomposition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 3, pp. 603–616, 2020.

[30] Bijiao Wu, Dingheng Wang, Guangshe Zhao, Lei Deng, and Guoqi Li, "Hybrid tensor decomposition in neural network compression," *Neural Networks*, vol. 132, pp. 309–320, 2020.

[31] Arthur Gretton, Karsten Borgwardt, Malte J Rasch, Bernhard Scholkopf, and Alexander J Smola, "A kernel method for the two-sample problem," *arXiv preprint arXiv:0805.2368*, 2008.

[32] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[33] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[34] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[35] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han, "Differentiable augmentation for data-efficient gan training," *Advances in neural information processing systems*, vol. 33, pp. 7559–7570, 2020.

[36] Yifan Jiang, Shiyu Chang, and Zhangyang Wang, "Transgan: Two pure transformers can make one strong gan, and that can scale up," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[37] Alex Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[38] Adam Coates, Andrew Ng, and Honglak Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[39] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[40] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic, "Tensorly: Tensor learning in python," *arXiv:1610.09555*, 2016.