Re-GAN: Data-Efficient GANs Training via Architectural Reconfiguration

Divya Saxena, Jiannong Cao, Jiahao Xu, Tarun Kulshrestha
The Hong Kong Polytechnic University, Hong Kong
{divsaxen, csjcao}@comp.polyu.edu.hk,
jiahaoxxuu@gmail.com, tarun.kulshrestha@polyu.edu.hk

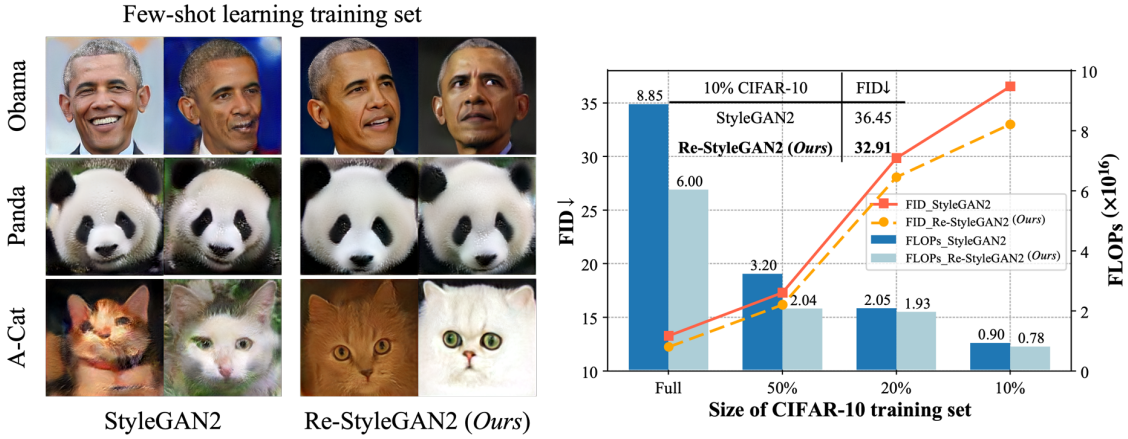

Figure 1: Results of our proposed Re-GAN where dynamically GANs architecture is reconfigured to explore different GANs sub-network structures during training time. (left) Image generation trained on multiple few-shot generation datasets, such as 100-shot Obama [1], Panda [1], and Animal Face-Cat (A-Cat) [2]; (right) FID scores vs. CIFAR-10 [3] training set size with training cost, FLOPs. Best viewed in color.

Abstract

Training Generative Adversarial Networks (GANs) on high-fidelity images usually requires a vast number of training images. Recent research on GAN tickets reveals that dense GANs models contain sparse sub-networks or "lottery tickets" that, when trained separately, yield better results under limited data. However, finding GANs tickets requires an expensive process of train-prune-retrain. In this paper, we propose Re-GAN, a data-efficient GANs training that dynamically reconfigures GANs architecture during training to explore different sub-network structures in training time. Our method repeatedly prunes unimportant connections to regularize GANs network and regrows them to reduce the risk of prematurely pruning important connections. Re-GAN stabilizes the GANs models with less data and offers an alternative to the existing GANs tickets and progressive growing methods. We demonstrate that Re-GAN is a generic training methodology which achieves stability on datasets of varying sizes, domains, and resolutions (CIFAR-10, Tiny-ImageNet, and multiple few-shot generation datasets) as well as different GANs architectures (SNGAN, ProGAN, StyleGAN2 and AutoGAN). Re-GAN also improves performance when combined with the recent augmentation approaches. Moreover, Re-GAN requires fewer floating-point operations (FLOPs) and less training time by removing the unimportant connections during GANs training while maintaining comparable or even generating higher-quality samples. When compared to state-of-the-art StyleGAN2, our method outperforms without requiring any additional fine-tuning step. Code can be found at this link: https://github.com/IntellicentAI-Lab/Re-GAN

1. Introduction

In recent years, Generative adversarial networks (GANs) [4]–[7] have made great strides in generating high-fidelity images. The GANs models serve as the backbone of several vision applications, such as data augmentation [5], [8], [9], domain adaptation [10], [11], and image-to-image translation [14]–[16].

The success of the GANs methods largely depends on a massive quantity of diverse training data, which is often time-consuming and challenging to collect [17]. Figure 1 shows how the performance of the StyleGAN2 [18] model drastically declines under the limited training data. As a

result, various new methods [1], [19], [20] have emerged to deal with the problem of insufficient data. Dynamic data-augmentation [1], [19]–[21] fills in the gap and stabilizes GANs training with less data. Very recently, [22], [23] introduced the lottery ticket hypothesis (LTH) in GANs (called "GANs tickets"), a complementary to the existing augmentation techniques. LTH identifies sparse sub-networks (called "winning tickets") that can be trained in isolation to match or even surpass the performance of unpruned models. [24] demonstrated that an identified GANs ticket can be used as a sparse structural prior to alleviate the problem of limited data in GANs. However, identifying these winning tickets requires many iterations of a time-consuming and computationally expensive train-prune-retrain process. This results in high training time and a number of floating-point operations (FLOPs) than training a dense GANs models, such as StyleGAN2 [18] and BigGAN [5]. In addition, these methods train a full-scale model before pruning, and then, after the pruning process, they engage in an extra fine-tuning to improve the performance. Given this perspective, we ask:

Is there any way to achieve training efficiency w.r.to both data and computation in GANs while preserving or even improving its performance?

One potential solution is network pruning during training, which can allow the exploration of different sub-network structures in training-time. Network structure exploration during training has shown to be effective in a variety of domains [25], [26], and its properties have been the subject of a significant amount of research [27], [28]. However, network pruning is never introduced to GANs training; as a result, the investigation of different sub-network structures exploration during GANs training remains mysterious.

To address this gap in the literature, we investigate and introduce the network pruning, i.e., connections, in GANs training by dynamically reconfiguring GANs architecture to allow the exploration of different sub-network structures in training time, dubbed as Re-GAN. However, on the other hand, it is common knowledge that the learning capabilities of the two competing networks—a generator (G) and a discriminator (D), need to be carefully maintained equilibrium in their respective capabilities for learning. Hence to build Re-GAN, the first question is: *how to explore different network structures during GANs training?* Network pruning during training regularizes the G to allow a robust gradient flow through G. This stabilizes the GANs models under limited training data and improves training efficiency. Re-GAN repeatedly prunes and grows the connections during the training process to reduce the risk of pruning important connections prematurely and prevent the model from losing its representational capabilities early in the training process. As a result, network growing provides a second opportunity to reinitialize pruned connections by reusing information from previously explored sub-network structures.

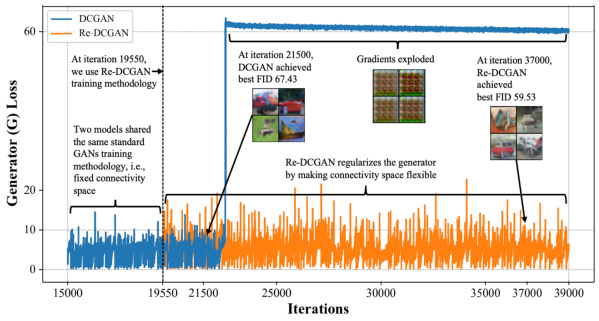

Figure 2: Conventional GANs training has fixed connectivity space. Re-GAN uses network pruning and growing during training to make connectivity space flexible that helps in the propagation of robust gradients. Best viewed in color.

The second question is: *how to explore different sub-network structures in G and D simultaneously?* On the one hand, if we employ a pretrained D (or G) and prune solely for G (or D), it can quickly incur an imbalance between the capabilities of D or G (particularly in the early stage of training), resulting in slow convergence. While it is possible to prune for G and D simultaneously, empirical experiments show that doing so significantly degrades the initial unstable GANs training, resulting in highly fluctuating training curves and, in many cases, a failure to converge. As a trade-off, we propose expanding D as per standard GANs training while applying pruning exclusively to G's architecture.

Additionally, our method is robust, working well with a wide range of GANs architectures (ProGANs [29], SNGAN [30], StyleGAN2, and AutoGAN [31], [32]) and datasets (CIFAR-10 [3], Tiny-ImageNet [33], Flickr Faces HQ [34], and many few-shot generation datasets). We find that exploring different sub-network structures during training accounts for a significant decrease in FID score compared to the vanilla DCGAN [35] architecture without a pre-trained model or fine-tuning the pruned model (see Figure 2). Our method delivers higher performance in less training time to state-of-the-art (SOTA) methods on most available datasets without additional hyperparameters that progressive growing method introduces, such as training schedules and learning rates for different generation stages (resolutions). This robustness allows the Re-GAN to be easily generalized on unseen datasets.

To the best of our knowledge, Re-GAN is the first attempt to incorporate network pruning during GANs training. Our technical innovations are as follows:

- We conduct the first in-depth study on taking a unified approach of incorporating pruning in GANs training without pre-training a large model or fine-tuning the pruned model.
- Our method repeatedly prunes and grows the connections during training to reduce the possibility of

pruning important connections and helps the model to maintain its representation ability early in training. Thus, network growing gives another chance to reinitialize pruned connections from the explored network sub-structures.

- Extensive experiments are conducted across a wide range of GANs architectures and demonstrated that Re-GAN could be easily applied on these GANs architectures to improve their performances, both in regular and low-data regime setups. For example, for the identified winning GANs ticket, ProGAN and StyleGAN2 on full CIFAR-10, we achieve 70.23%, 18.81%, and 19% training FLOPs savings, respectively, while improved generated sample quality for both full and 10% training data. Re-GAN presents a viable alternative to the GANs tickets and progressive growing techniques. Additionally, the performance of Re-GAN is enhanced when integrated with recent augmentation techniques.

2. Related Works

Stabilize the GANs training. In recent years, different loss functions [4], [36], [37], regularization [30], [38], and architectural designs [39], [40] have all been proposed as ways to enhance GANs [6]. Our efforts are in the category of network architecture design. Recently, state-of-the-art (SOTA) models like StyleGAN2 and BigGAN have suggested making networks deeper and wider. They have also shown that training deep GANs networks usually leads to better generalization. However, a deeper model with more convolution layers results in a longer training time for GANs. This is because a deeper model contains a greater number of model parameters and a weaker gradient flow via G [29], [41], [42]. The findings of Progressive GAN (ProGAN) [29] show that gradually growing and training both networks together from a lower resolution, stabilizes the training and enhances the generation quality. MSG-GAN [39] introduced a solution to the gradient flow issue in which G gets feedback from a number of different resolutions simultaneously. Nevertheless, these methods further add to the computational cost, necessitating an even high amount of GPU memory and more training time.

GANs compression. GANs, like other deep neural networks, excel in image generation and translation tasks [29], [43]–[45], but suffer from high computational complexity and memory requirements. Han et al. [46] introduced a co-evolutionary pruning technique for GANs compression, while Wang et al. [47] proposed quantizing GANs to 1 or 2 bits. Li et al. [48] used distillation to improve compressed D with a pre-trained GANs model, and Wang et al. [49] combined these three techniques into one framework. Although impressive results were achieved in training compressed G with a pre-trained D [50], existing GANs compression approaches require pre-training an over-parameterized model, limiting training efficiency. Meanwhile, [51] proposed dynamic network size modification during training, but design space exploration is constrained by network augmentation. Our approach injects sparsity during training to enhance generalization, stabilize GANs with limited data, and improve training efficiency, distinguishing it from these inference-focused techniques.

Lottery Ticket Hypothesis (LTH). Recently, [22]–[24] have shown the existence of winning tickets in the min-max game beyond minimization by extending LTH to GANs. [52] claims the existence of independently trainable sparse sub-networks capable of performing at the same level as dense networks and in some instances, even better. While [53], [54] scaled up LTH by rewinding [55], [56]. However, these methods do not restore key connections that have been prematurely pruned, which restricts the model's capacity. In contrast, the proposed method can restore connections that have been prematurely pruned and can better preserve the capacity of the model.

On the other hand, the total number of FLOPs necessary to locate and train a winning ticket at a sparsity level of 80% is more than four times as many as the number of FLOPs necessary to train a dense model [57]. As the size of SOTA models have grown (e.g., BigGAN and StyleGAN2), such huge resource needs would cause financial and environmental problems [58]–[60]. Being unique and orthogonal efforts from ours, these methods do not focus on the training efficiency yet.

3. Methodology

3.1. Design motivation

The synaptic connection topology of the brain is very dynamic, yet the brain still maintains a stable and efficient computing function [61], [62]. It has been shown that the underlying process of synaptic rewiring plays a crucial part in learning [63]. We consider the training of GANs to be akin to such biological learning processes and combine it with different training objectives under sparsity constraint.

In contrast to conventional GANs training with predefined static connectivity, we prune and grow the connections during the GANs training. We conduct a thorough investigation to find that introducing network pruning and growing into GANs training yields desirable results on many fronts, specifically: (1) stabilizes GANs across various datasets, resolutions, domains, and architectures; (2) maintaining comparable or even generating higher-quality samples; and (3) enhancing the GANs training efficiency w.r.to both data and computation by removing the unimportant connections.

In the next sub-section, we review the traditional GANs models, then describe in detail our proposed GANs training methodology via architectural reconfiguration.

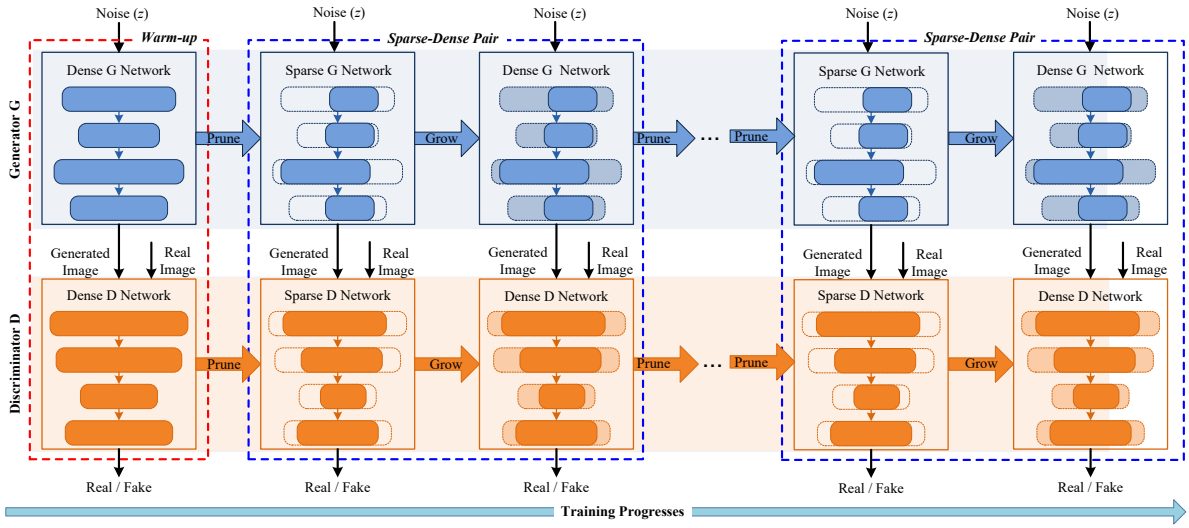

Figure 3: Architecture of Re-GAN, shown here on the base model proposed in GANs [6]. The key feature of Re-GAN is that it provides sparse–dense model pairs at the end of the training process. This is beneficial in practice, as it may be desired to deploy compressed variant in resource-constrained environments without redoing the full training process.

3.2. Generative Adversarial Networks

A GANs model consists of a discriminator D and a generator G. The training objectives of the D and G can be represented as θ_D and θ_G, respectively. The GANs framework can be as follows:

$$\max_{\theta_D} \mathbb{E}_{x\sim p_{data}}[f_D(D(x))] + \mathbb{E}_{z\sim p_z}[f_G\left(D(G(z))\right)] \quad (1)$$

$$\min_{\theta_G} \mathbb{E}_{z\sim p_z}[g_G\left(D(G(z))\right)], \quad (2)$$

where p_z is the prior distribution (e.g., $N(0, I)$) and p_{data} is the real training data used to approximate the data distribution. The notations f_D, f_G, and g_G in Eq. (1) and (2) represent the mapping functions from which various GANs losses can be derived [64].

3.3. GANs training with architectural reconfiguration

A network's topology is equivalent to a directed acyclic graph with a predetermined order of nodes. Each node X_{in} serves as an input feature, and each edge functions as a computation cell with hyperparameters. We parameterize architectural space by associating a mask variable $m \in \{0,1\}$ with each computation cell to enable training time pruning $(m = 1 \rightarrow 0)$ and growing $(m = 0 \rightarrow 1)$. We consider a single-level configuration space for GANs architecture that enables dynamic pruning and growing networks width-wise.

Revisiting GANs training. Re-GAN reconfigures G's architecture to explore different sub-network structures during training. As shown in Algorithm 1, we start the GANs training on the dense network for several iterations, warm-up phase, and learn the connection weights to know their importance. To achieve the sparse structure, we prune the least important weights based on the pre-defined pruning criterion. We then activate the pruned connections to further grow the network after a series of iterative optimizations. Once the network topology has been changed from sparse to dense, it will train the new network until the next connectivity update. The overview of Re-GAN is shown in Figure 3. The main factors of Re-GAN training are as follows: 1) sparsity distribution, 2) update schedule, 3) pruning, and 4) grow.

GANs sub-networks exploration. We prune the low-weight connections by using unstructured magnitude pruning [65], [66] by using the binary masks m_g and m_d. The pruning ratio ρ determines the amount of weights removed during the pruning phase. We use ρ = 10% in all experiments except StyleGAN2 for FFHQ (ρ = 30%). We use the same sparsity across all layers, i.e., uniform sparsity distribution. First, we sort the weights and produced a binary mask to exclude weights smaller than λ. If the mask of each layer that has to be regularized is calculated, then the mask of the whole parameter space, M, can be obtained for both G and D. Consequently, Eq. 1 and 2 here become:

$$\max_{\theta_D} \mathbb{E}_{x\sim p_{data}}[f_D(D(x, \theta_D \odot M_D))] + \mathbb{E}_{z\sim p_z}[f_G\left(D(G(z, \theta_G \odot M_G))\right)] \quad (3)$$

$$\min_{\theta_G} \mathbb{E}_{z\sim p_z}[g_G\left(D(G(z, \theta_G \odot M_G))\right)], \quad (4)$$

where θ_G, M_G, θ_D and M_D are the corresponding weights and masks for G and D, $\odot$ stands for Hadamard product.

The update schedule contains two key factors, (i) the update interval (g) is the number of training iterations

between the pruning and growing phase and vice versa; (ii) the learning rate, we set 1/10 the initial learning rate during the network growing phase. Other hyperparameters remained the same as per the given GANs network architecture.

During each connectivity update, we use the weight magnitude as the indicator for pruning and make the following state transitions: if an indicator changes from 1 to 0, we remove the corresponding connection from the computational graph; During the network growing, all the pruned connections return to 1, indicating that the network will be grown back, initialized weights with 0 and trained. The network growing phase boosts the network's model capacity and makes it easier to get a better local minimum than with a sparse network topology. Similarly, growing new connections in G allows for the search for optimal connectivity and escape from undesired local minima. In addition, by deleting the least significant connections, our method considerably reduces both the training memory cost and the training time, hence enhancing training efficiency. We experiment with three distinct loss functions for the d_{critic} function namely, WGAN-GP [67] which is used by ProGAN, hinge loss is used by SNGAN and non-saturating GAN loss with 1-sided GP [6] which is used by StyleGAN2.

4. Experimental Results

In this section, we perform experiments on various datasets that contain a wide range of content categories. We evaluate CIFAR-10 and Tiny-ImageNet at 64×64 resolution, based on unconditional ProGAN [29][68], SNGAN [30][69], and SOTA StyleGAN2 [18][70]. We use the widely adopted evaluation metrics, such as the Fréchet Inception Distance (FID) [71] and Inception Score (IS) [4], and also provide the number of real images (#RI (in Millions (M)) [29][30]. If model *A* is taking less number of real images and is achieving comparable or better results than model *B*, it shows that model *A* is efficient. Note that higher performing GANs models are indicated by lower FID (↓), and larger IS (↑). We further extend our study on 256^2 resolution, we test on Animal-Face Dog and Cat [2], 100-Shot-Obama [1], Panda [1], and Grumpy-cat [1], and Oxford-flowers [72]. On 1024^2 resolution, we test on 1k, 5k, 10k and 70k Flickr-Face-HQ (FFHQ) [34]. We also perform extensive ablation study to analyze the effectiveness of each component in Re-GAN.

4.1. Implementation details

For ProGAN and StyleGAN2, for each dataset, we use the same 512-dimensional initial latent space, derived from a standard normal distribution $N(0, 1)$, followed by hypersphere normalization [29]. For SNGAN, the latent variable dimension is 128 for all datasets. For all experiments, we use the same hyperparameter settings. However, we reduce the learning rate ten times during network growing phase for better learning. We initialize parameters using standard normal distribution $N(0, 1)$.

We trained all models ourselves and used the same hardware to provide fair comparisons of the training times for the corresponding set of experiments. We use NVIDIA RTX 3090 GPU with 24 GB for all 64×64, and 256×256 experiments, and NVIDIA Quadro RTX 8000 GPU with 48 GB for all 1024×1024 experiments. We train for the same number of iterations predefined by the baseline models. We save the checkpoints every 10k iterations during training and report the best FID as well as the time it took to obtain that score from the checkpoints (e.g., happens at least after 15 hours of training for StyleGAN2).

Algorithm1 Re-GAN training

Input: An L-layer GANs model; target sparsity k; total training iterations T_{total}; training iterations between two consecutive steps, update interval g; training set S; learning rate α;
Output: Converged G and D

1. Randomly initialize all model weights $\theta_G \in \mathbb{R}^N, \theta_D \in \mathbb{R}^N$;
2. **Train** the weights θ_G and θ_D for Δ_W epochs // *warm-up phase*
3. **while** each training iteration t ∈ [T_{total}] **do**
4. **if** mod (t, g) = 0 **then**
   // entered a pruning phase
5. $\theta_G \leftarrow T_k(\theta_G, k)$ // *Apply compression (top-k)-*
6. $\theta_D \leftarrow T_k(\theta_D, k)$ // *operations on weights*
7. $m_G, m_D \leftarrow 1[\theta_i \neq 0]$ // *Create masks*
8. **else**
   // entered a growing phase
9. $\alpha = \alpha * 0.1$ // reduce the learning rate by 10
10. $m_G, m_D \leftarrow 1_N$ // reset all masks
11. **end if**
12. Sample a mini-batch from S;
13. $\theta_G, \theta_D \leftarrow \theta_G \odot m_G, \theta_D \odot m_D$ // Update θ_G, θ_D by Eq. (3)
14. **end while** // and (4), respectively

For StyleGAN2, we find that using default settings is hardly to converge on CIFAR-10 and Tiny-ImageNet, so we follow the settings in [4] for CIFAR-10, in which both path length regularization and lazy regularization are not used, the coefficient for R1 regularization γ is set to 0.1 for CIFAR-10, the output channel at final size level is 128 instead of 512 and doubled at each coarser level with a maximum 512. We use a similar setting for Tiny-ImageNet except setting γ to 1. For few-shot learning, FFHQ experiments, we set γ to 10. We note a discrepancy between the results the author described scores and what we could able to accomplish using the author's provided code. This could be the result of subtle hardware variations or variations between runs.

4.2. Comparison to state-of-the-art

We use the best-performing model ProGAN and StyleGAN2 and perform experiments at 64 × 64 resolution on CIFAR-10 and Tiny-ImageNet. We evaluate our method with 100% data available and with 10%, 20% and 50% data available, with the findings summarized in Table

1 and 2, respectively.

The following observations can be drawn: First, Re-ProGAN and Re-StyleGAN2 can achieve consistently improved performance for all training cases, from complete data to limited data. In particular, likely to gain more when there is less training data available (e.g., 10%), which is consistent with our design principle. With only 10% training data available, Re-ProGAN and Re-StyleGAN2 obtain massive gains of 29.33 FID/7.69 IS and 32.91 FID/7.35 IS on CIFAR-10, respectively, and 79.08 FID/6.45 IS and 82.63 FID/6.47 IS on Tiny-ImageNet, respectively. Second, due to removing unimportant connections of low weights, Re-GAN consistently reduces the cost of computation associated with training and requires fewer total hours of GPU training time.

Qualitative results and precision-recall curve for Re-StyleGAN2 on CIFAR-10 is in the supplementary material. We also provide a study in the supplementary material to show that Re-GAN helps to alleviate mode collapse.

Table 1: FID comparison on few-shot datasets at 64×64 resolution. FID and IS are calculated using 10k randomly generated samples, with the test data (10k) serving as the reference distribution. The best performance is highlighted.

Dataset	Size	Methods	#Real Images (RI)(in M)	Training Time (hour)	FLOPs $\times 10^9$	IS↑	FID↓
CIFAR-10	50K	ProGAN	19.5	11.6	12.97	8.05 ± 0.20	26.14
		Re-ProGAN	**15.9**	**9.8**	**10.53**	**8.22 ± 0.19**	**21.17**
		StyleGAN2	17.7	14.2	5.0	8.76 ± 0.22	13.19
		Re-StyleGAN2	**12.1**	**10.1**	**4.05**	**8.88 ± 0.24**	**12.16**
Tiny-ImageNet	100K	ProGAN	9.4	53	27.47	9.44 ± 0.31	40.27
		Re-ProGAN	**8.7**	**48.9**	**22.18**	**9.57 ± 0.35**	**37.47**
		StyleGAN2	19.2	52.5	21.07	11.75 ± 0.25	20.95
		Re-StyleGAN2	**17.8**	**46.7**	**17.04**	**12.10 ± 0.28**	**20.21**

Table 2: Experiments on 64×64 resolution 50%, 20% and 10% of training datasets. The best performance is highlighted.

Dataset	Methods	50% data			20% data			10% data		
		#RI (in M)	IS↑	FID↓	#RI (in M)	IS↑	FID↓	#RI (in M)	IS↑	FID↓
CIFAR-10	ProGAN	10.5	7.75±0.19	28.33	7.2	7.51±0.19	28.64	5.6	7.47±0.16	30.08
	Re-ProGAN	**9.4**	**7.97±0.13**	**25.74**	**6.1**	**7.63±0.18**	**27.19**	**4.9**	**7.69±0.23**	**29.33**
	StyleGAN2	6.4	8.39±0.15	17.22	4.1	7.36±0.27	29.77	3.2	7.20±0.24	36.45
	Re-StyleGAN2	**4**	**8.48±0.16**	**16.09**	**3.8**	**7.46±0.23**	**27.98**	**2.7**	**7.35±0.13**	**32.91**
Tiny-ImageNet	ProGAN	9.6	8.71±0.27	49	8	6.84±0.12	77.86	7.7	6.38±0.19	83.27
	Re-ProGAN	**8.3**	**8.84±0.25**	**46.78**	**7.7**	**6.92±0.15**	**75.22**	**7.4**	**6.45±0.22**	**79.08**
	StyleGAN2	14.5	11.02±0.27	30.03	5.1	8.32±0.24	65.26	3.2	6.32±0.10	84.86
	Re-StyleGAN2	**13.2**	**11.25±0.31**	**29.19**	**4.2**	**8.53±0.29**	**62.34**	**1.3**	**6.47±0.10**	**82.63**

Table 3: FID comparison to NAS-based GANs, i.e., AutoGAN for full CIFAR-10 dataset.

Methods	#RI (M)	Training time (hour)	Flops ($\times 10^{16}$)	FID↓
				Full data
AutoGAN (Top A)	4.7	6.7	19M	17.70
Re-AutoGAN (Top A)	**4.1**	**5.72**	**16.5M**	**17.61**
AutoGAN (Top B)	5.2	7.75	23M	18.08
Re-AutoGAN (Top B)	**4.2**	**5.95**	**18M**	**17.30**
AutoGAN (Top C)	5.7	8.12	23M	17.09
Re-AutoGAN (Top C)	**5**	**6.73**	**20M**	**16.61**

We extend our method comparison with the NAS-based GANs method, i.e., AutoGAN (Top A, B and C architectures) [31], [32]. The result in Table 3 shows that Re-GAN not only improves the performance for human-designed networks (e.g., SNGAN, ProGAN, StyleGAN2) but also boosts the performance and reduces training time and FLOPs for automatically searched architecture, AutoGAN. It also minimizes training time and FLOPs for all top three searched GANs architectures.

4.3. Comparison with GANs tickets

Recently, Iterative Magnitude Pruning (IMP), an unstructured magnitude pruning [24] have shown performing well in finding lottery tickets in GANs than other pruning methods [66], [73] to find GANs tickets [22], [23] [24]. We find GANs tickets using IMP at 20% and 46% pruning ratio for the full training dataset, named ST_SNGAN@20% and ST_SNGAN@46%, respectively. We also include growing with random initialization (RI), *Re-SNGAN-RI* in which weights are randomly initialized during the growing rather than zero initialization. Also, we set same learning rate for both dense and sparse phases, *Re-SNGAN-lr*. The results are summarized in Table 4. The results show that our proposed Re-SNGAN performs well w.r.to FID. Re-SNGAN-RI results in longer convergence times, as network has to explore more diverse weight configurations to find best solution. While, setting 1/10 learning rate during dense phase performs better than same learning rate in both phases as it promotes stability by

avoiding large, sudden weight updates. Also, finding GANs tickets is highly time consuming. While the training time for exploring different sub-network structures during training, even only at 10% pruning ratio, is comparable to the dense model; In addition, Re-SNGAN consumes less training FLOPs than the GANs tickets.

Table 4: Comparison to GANs tickets methods on CIFAR-10.

Methods	#RI (M)	Training time (hour)	Flops ($\times 10^{17}$)	FID↓ Full data
SNGAN@0%	**30**	**6.2**	**1.01**	20.12
ST_SNGAN@20%	92.5	19.7	2.60	18.96
ST_SNGAN@46%	150	32	3.46	18.22
Re-SNGAN-lr	49.4	10.5	1.57	19.06
Re-SNGAN-RI	33.1	7.3	1.25	19.71
Re-SNGAN	31.5	6.6	1.03	**17.87**

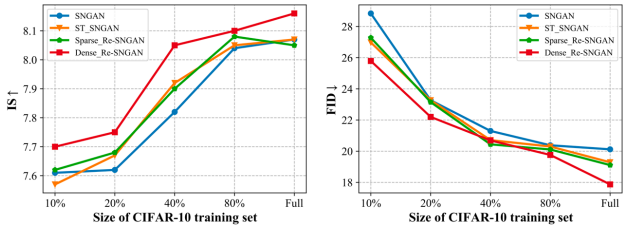

Figure 4: IS (↑) and FID (↓) results of SNGAN with 10%, 20%, 40%, 80%, and full training data of CIFAR-10. Four settings are evaluated: (i) SNGAN (unpruned SNGAN), (ii) ST_SNGAN (applied IMP for finding sparse GANs ticket at 20% pruning ratio), (iii) Sparse_Re-SNGAN (Re-GAN output is sparse), (iv) Dense_Re-SNGAN (Re-GAN output is dense). The main advantage is that it provides both a high-performing dense model and its compressed version at the end of the training.

Table 5: FID comparison on few-shot datasets at 256×256 resolution. FID and IS are calculated using 5k randomly generated samples, with the training data serving as the reference distribution. The best performance is highlighted.

Methods	Obama	Grumpy Cat	Panda	Animal Face		Oxford Flowers
				Cat	Dog	
# of images	100	100	100	389	160	1000
ProGAN	129.52	135.89	235.86	289.73	259.32	224.70
Re-ProGAN	**110.46**	**124.06**	**205.82**	**283.29**	**227.59**	**198.56**
StyleGAN2	86.67	51.3	95.08	81.91	157.63	147.67
Re-StyleGAN2	**72.92**	**38.23**	**78.45**	**73.05**	**133.4**	**138.35**
StyleGAN2 + APA	68.74	33.72	18.24	62.71	81.13	85.88
Re-StyleGAN2 + APA	**64.28**	**31.74**	**16.33**	**60.26**	**77.44**	**82.19**
StyleGAN2 + DiffAug	46.31	28.6	12.89	42.85	58.47	46.21
Re-StyleGAN2 + DiffAug	**45.7**	**27.36**	**12.6**	**42.11**	**57.2**	**45.13**

We provide another set of experiments of SNGAN training on CIFAR-10, with training data ranging from 10% to 100%. Figure 4 provides a summary of the key findings. The following observations can be drawn: First, at the same data availability (from 10% to even 100%), exploring multiple sub-networks during training at only 10% pruning ratio is always preferable to training the dense model and sparse ticket; Second, sparse Re-GAN model is also performing consistently well than GANs tickets.

4.4. Few-shot generation

Collecting a large number of image datasets can be prohibitively costly or perhaps impossible for a certain character or a topic. In this context, a data-efficient model for the image synthesis task will be highly important. In Table 5, we show that our method finds data-efficient GANs that can be trained from scratch with only 100 training image data (without any pre-training) and demonstrates consistently superior performance (See supplementary material for visualizations of few-shot generation and style interpolation). Results also show that proposed Re-GAN improve the generated sample quality of SOTA methods, such as Diffaug [1] and APA [21] for few-shot datasets. We further extend our study for high-resolution experiments (Table 6 and Figure 5) where the Re-StyleGAN2 model gets better FID scores on all 1k, 5k, 10k and 70k FFHQ datasets (see supplementary material for more details). To further demonstrate the effectiveness of our Re-GAN, we apply it to other recent methods, FastGAN [74] for several few-shot datasets. Results are shown in supplementary material.

Table 6: FID comparison on few-shot datasets at 1024×1024 resolution. FID and IS are calculated using 50k randomly generated samples, with the training data (70k) serving as the reference distribution.

Methods	FFHQ			
	70k	10k	5k	1k
StyleGAN2	4.35	13.06	21.76	40.24
Re-StyleGAN2	**4.19**	**11.22**	**19.13**	**36.3**

4.5. Ablation and analysis

Prune G/D: We study the impact of exploring different sub-network structures on G or D only, or on both G and D (represented as B) at 10%, 30%, and 50% pruning ratio (i.e., weight remaining ratio indicates the sparsity levels of explored sub-network structures during training) on CIFAR-10 full training dataset. Figure 6 shows a summary of the results. We notice that employing pruning on D or both G and D, achieves comparable or better performance than the base model. While for the same sparsity

distribution (i.e., pruning ratio), pruning only on G achieves significant performance improvement (i.e., largely reducing the FID). We also notice that update interval (g) 100 works well with only G.

Figure 5: Few-shot generalization results of Re-StyleGAN2 on 10k FFHQ (1024×1024). Samples are randomly drawn without cherry picking.

Robustness to learning rate. Previous research works [75]–[77] have observed that GANs convergence highly depends on the selection of hyperparameters, especially the learning rate, α. As shown in Table 7, we validate our network by training it with five different α values (0.0001, 0.0002, 0.0005, 0.0007, and 0.00001) for the CIFAR-10 dataset. Results show that for all different α values, our models converged and achieved similar inception scores.

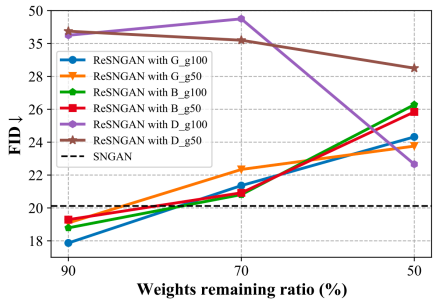

Figure 6: FID with different pruning ratio and update interval (g) of Re-SNGAN (CIFAR-10).

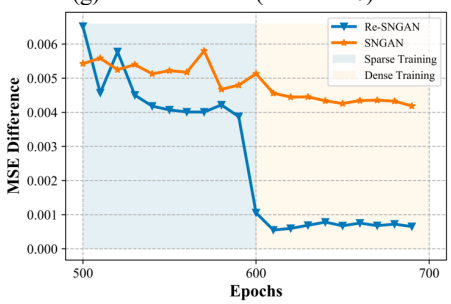

Figure 7: Stability during training on the 10% CIFAR-10.

Stability during training. To compare the stability of Re-SNGAN with SNGAN, we evaluate how the samples produced for the same set of latent points vary during training. As proposed in [30], we use the mean squared error (MSE) between two consecutive generated samples as a quantitative measure of training stability. Figure 7 shows the MSE between generated images from the same latent code in the midst of successive epochs on the 10% CIFAR-10 dataset. Results show that dense Re-SNGAN converges consistently over time, while SNGAN varies dramatically across epochs.

Table 7: Robustness to learning rate.

Methods	Learning rate	Results	
		IS	FID
Real images	-	11.34	-
SNGAN	0.0002	8.07±0.16	20.12
Re-SNGAN	0.0007	7.93±0.30	21.76
	0.0005	8.03±0.23	20.17
	0.0002	**8.16±0.15**	**17.87**
	0.0001	8.03±0.17	19.03
	0.00001	8.01±0.25	19.65

5. Conclusion and Discussion of Broader Impact

We introduce a new perspective on data-efficient GANs by exploring different sub-network structures during training, which is a novel approach compared to existing methods. Extensive experiments on various GANs architectures, objectives, and datasets demonstrate the effectiveness and training efficiency of our proposed GANs training methodology, Re-GAN. In this work, we focus on unstructured sparsity to showcase the algorithm-level innovations of Re-GAN. However, only specific hardware supports sparse tensor cores for weight pruning. Future work could aim for a high pruning ratio and reconfigure network width and depth simultaneously during GANs training, addressing the limitations of the current approach.

This research aims to enhance generalization performance, stabilize GANs models under limited data, and improve training efficiency, which has practical implications for various applications and industries. However, these advancements might also exacerbate existing social risks associated with GANs. We plan to investigate whether the sparse structures affect issues like image generation bias in future work. The ability to generate images with less data could potentially be exploited for undesirable applications, such as DeepFakes.

Acknowledgments

This work is supported by PolyU Internal Start-up Fund (Grant no: P0038876), Research Impact Fund (No. R5034-18), and RGC Collaborative Research Fund (Grant no: C5026-18G).

References

[1] S. Zhao, Z. Liu, J. Lin, J. Y. Zhu, and S. Han, "Differentiable augmentation for data-efficient GAN training," in *Advances in Neural Information Processing Systems*, 2020, pp. 7559–7570.

[2] Z. Si and S.-C. Zhu, "Learning hybrid image templates (HIT) by information projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1354–1367, 2011.

[3] A. Krizhevsky and others, "Learning multiple layers of features from tiny images," 2009.

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, Jan. 2017, pp. 214–223, Accessed: Apr. 05, 2019. [Online]. Available: http://arxiv.org/abs/1701.07875.

[5] A. Brock, J. Donahue, and K. Simonyan, "Large scale GaN training for high fidelity natural image synthesis," in *Proceedings of International Conference on Learning Representations*, 2019.

[6] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, vol. 3, no. January, pp. 2672–2680, doi: 10.3156/jsoft.29.5_177_2.

[7] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2813–2821, doi: 10.1109/ICCV.2017.304.

[8] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[9] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using GAN for improved liver lesion classification," in *International symposium on biomedical imaging (ISBI)*, 2018, pp. 289–293.

[10] J. Hoffman *et al.*, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*, 2018, pp. 1989–1998.

[11] H.-K. Hsu *et al.*, "Progressive domain adaptation for object detection," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2020, pp. 749–757.

[12] Y.-C. Cheng, C. H. Lin, H.-Y. Lee, J. Ren, S. Tulyakov, and M.-H. Yang, "InOut: Diverse Image Outpainting via GAN Inversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11431–11440.

[13] P. Teterwak *et al.*, "Boundless: Generative adversarial networks for image extension," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10521–10530.

[14] X. Huang, M. Y. Liu, S. Belongie, and J. Kautz, "Multimodal Unsupervised Image-to-Image Translation," in *European conference on computer vision (ECCV)*, 2018, pp. 172–189, doi: 10.1007/978-3-030-01219-9_11.

[15] H. Y. Lee *et al.*, "DRIT++: Diverse Image-to-Image Translation via Disentangled Representations," *Int. J. Comput. Vis.*, vol. 128, no. 10–11, pp. 2402–2417, 2020, doi: 10.1007/s11263-019-01284-z.

[16] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, Mar. 2017, vol. 2017-Octob, pp. 2242–2251, doi: 10.1109/ICCV.2017.244.

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

[18] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.

[19] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 12104–12114.

[20] Z. Zhao, Z. Zhang, T. Chen, S. Singh, and H. Zhang, "Image augmentations for gan training," *arXiv Prepr. arXiv2006.02595*, 2020.

[21] L. Jiang, B. Dai, W. Wu, and C. C. Loy, "Deceive D: adaptive pseudo augmentation for GAN training with limited data," in *Advances in Neural Information Processing Systems*, 2021, vol. 34, pp. 21655–21667.

[22] X. Chen, Z. Zhang, Y. Sui, and T. Chen, "GANs Can Play Lottery Tickets Too," *arXiv Prepr. arXiv2106.00134*, 2021, [Online]. Available: http://arxiv.org/abs/2106.00134.

[23] N. M. Kalibhat, Y. Balaji, and S. Feizi, "Winning lottery tickets in deep generative models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, no. 9, pp. 8038–8046.

[24] T. Chen, Y. Chen, Z. Gan, J. Liu, and Z. Wang, "Data-Efficient GAN Training Beyond (Just) Augmentations: A Lottery Ticket Perspective," in *Advances in Neural Information Processing Systems*, 2021, vol. 25, pp. 20941–20955, Accessed: Jul. 06, 2022. [Online]. Available: https://github.

[25] M. R. U. Saputra, P. P. B. De Gusmao, Y. Almalioglu, A. Markham, and N. Trigoni, "Distilling knowledge from a deep pose regressor network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 263–272.

[26] H. Tian, B. Liu, X.-T. Yuan, and Q. Liu, "Meta-learning with network pruning," in *European Conference on Computer Vision*, 2020, pp. 675–700.

[27] S. Han *et al.*, "Dsd: Dense-sparse-dense training for deep neural networks," *arXiv Prepr. arXiv1607.04381*, 2016.

[28] A. Peste, E. Iofinova, A. Vladu, and D. Alistarh, "Ac/dc: Alternating compressed/decompressed training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2021, vol. 34, pp. 8557–8570.

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

[30] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida,

"Spectral normalization for generative adversarial networks," in *Proceedings of International Conference on Learning Representations*, 2018.

[31] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural architecture search for generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-Octob, pp. 3223–3233, doi: 10.1109/ICCV.2019.00332.

[32] "AutoGAN." https://github.com/VITA-Group/AutoGAN.

[33] J. Wu, Q. Zhang, and G. Xu, "Tiny imagenet challenge," *Tech. Rep.*, 2017.

[34] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Dec. 2019, pp. 4401–4410, Accessed: Oct. 02, 2019. [Online]. Available: http://arxiv.org/abs/1812.04948.

[35] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv Prepr. arXiv1511.06434*, 2015.

[36] I. Deshpande, Z. Zhang, and A. G. Schwing, "Generative modeling using the sliced wasserstein distance," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3483–3491.

[37] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary Equilibrium Generative Adversarial Networks," *arXiv Prepr. arXiv1703.10717*, 2017, doi: 10.1109/ACCESS.2018.2804278.

[38] H. Zhang, Z. Zhang, A. Odena, and H. Lee, "Consistency Regularization for Generative Adversarial Networks," *arXiv Prepr. arXiv1910.12027*, 2019, [Online]. Available: http://arxiv.org/abs/1910.12027.

[39] T. D. Nguyen, T. Le, H. Vu, and D. Phung, "Dual discriminator generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2017, vol. 2017-Decem, pp. 2671–2681.

[40] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," Nov. 2016, Accessed: Apr. 04, 2019. [Online]. Available: http://arxiv.org/abs/1511.06434.

[41] H. Zhang *et al.*, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5907–5915.

[42] A. Karnewar and O. Wang, "Msg-gan: Multi-scale gradients for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7799–7808.

[43] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "Cartoongan: Generative adversarial networks for photo cartoonization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9465–9474.

[44] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 11, pp. 3365–3385, 2019.

[45] D. Saxena and J. Cao, "Generative adversarial networks (GANs) challenges, solutions, and future directions," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–42, 2021.

[46] H. Shu *et al.*, "Co-evolutionary compression for unpaired image translation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-Octob, pp. 3234–3243, doi: 10.1109/ICCV.2019.00333.

[47] P. Wang *et al.*, "QGAN: Quantized Generative Adversarial Networks," *arXiv Prepr. arXiv1901.08263*, 2019, [Online]. Available: http://arxiv.org/abs/1901.08263.

[48] M. Li, J. Lin, Y. DIng, Z. Liu, J. Y. Zhu, and S. Han, "GAN Compression: Efficient Architectures for Interactive Conditional GANs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5283–5293, doi: 10.1109/CVPR42600.2020.00533.

[49] H. Wang, S. Gui, H. Yang, J. Liu, and Z. Wang, "Gan slimming: All-in-one gan compression by a unified optimization framework," in *European Conference on Computer Vision*, 2020, pp. 54–73.

[50] C. Yu and J. Pool, "Self-supervised generative adversarial compression," in *Advances in Neural Information Processing Systems*, 2020, vol. 2020-Decem.

[51] X. Song, Y. Chen, Z. H. Feng, G. Hu, D. J. Yu, and X. J. Wu, "SP-GAN: Self-Growing and Pruning Generative Adversarial Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 6, pp. 2458–2469, 2021, doi: 10.1109/TNNLS.2020.3005574.

[52] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proceedings of International Conference on Learning Representations*, 2019.

[53] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proceedings of International Conference on Learning Representations*, 2019.

[54] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv Prepr. arXiv1902.09574*, 2019.

[55] J. Frankle, G. K. Dziugaite, D. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," in *International Conference on Machine Learning*, 2020, pp. 3259–3269.

[56] A. Renda, J. Frankle, and M. Carbin, "Comparing Rewinding and Fine-tuning in Neural Network Pruning," in *Proceedings of International Conference on Learning Representations*, 2020, [Online]. Available: http://arxiv.org/abs/2003.02389.

[57] S. Liu *et al.*, "Deep Ensembling with No Overhead for either Training or Testing: The All-Round Blessings of Dynamic Sparsity," 2021, Accessed: Jun. 27, 2022. [Online]. Available: http://arxiv.org/abs/2106.14568.

[58] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, 2019, doi: 10.1016/j.jpdc.2019.07.007.

[59] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Commun. ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[60] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv Prepr. arXiv1906.02243*, 2019.

[61] A. J. G. D. Holtmaat *et al.*, "Transient and persistent dendritic spines in the neocortex in vivo," *Neuron*, vol. 45, no. 2, pp. 279–291, 2005.

[62] D. D. Stettler, H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert, “Axons and synaptic boutons are highly dynamic in adult visual cortex,” *Neuron*, vol. 49, no. 6, pp. 877–887, 2006.
[63] A. J. Peters, S. X. Chen, and T. Komiyama, “Emergence of reproducible spatiotemporal activity during motor learning,” *Nature*, vol. 510, no. 7504, pp. 263–267, 2014.
[64] J. H. Lim and J. C. Ye, “Geometric gan,” *arXiv Prepr. arXiv1705.02894*, 2017.
[65] Y. LeCun, J. S. Denker, and S. A. Solla, “Advances in Neural Information Processing Systems.” San Francisco, CA: Morgan Kaufmann, 1990.
[66] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv Prepr. arXiv1510.00149*, 2015.
[67] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved training of wasserstein GANs,” in *Advances in Neural Information Processing Systems*, Mar. 2017, vol. 2017-Decem, pp. 5768–5778, Accessed: Apr. 05, 2019. [Online]. Available: http://arxiv.org/abs/1704.00028.
[68] “ProGAN.” https://github.com/BakingBrains/Progressive_GAN-ProGAN-_implementation.
[69] “SNGAN.” https://github.com/w86763777/pytorch-gan-collections.
[70] “StyleGAN-V2.” https://github.com/rosinality/stylegan2-pytorch.
[71] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium,” in *Advances in neural information processing systems*, 2017, pp. 6626--6637.
[72] M.-E. Nilsback and A. Zisserman, “A visual vocabulary for flower classification,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, 2006, vol. 2, pp. 1447–1454.
[73] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.
[74] B. Liu, Y. Zhu, K. Song, and A. Elgammal, “Towards Faster and Stabilized Gan Training for High-Fidelity Few-Shot Image Synthesis,” in *Proceedings of International Conference on Learning Representations*, 2021.
[75] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *Advances in Neural Information Processing Systems*, Jun. 2016, pp. 2234–2242, Accessed: Sep. 06, 2019. [Online]. Available: http://arxiv.org/abs/1606.03498.
[76] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for GANs do actually converge?,” in *International conference on machine learning*, 2018, pp. 3481–3490.
[77] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, “Unrolled Generative Adversarial Networks,” Nov. 2016, Accessed: Sep. 06, 2019. [Online]. Available: http://arxiv.org/abs/1611.02163.