Sparse Unbalanced GAN Training with In-Time Over-Parameterization

Anonymous authors
Paper under double-blind review

Abstract

Generative adversarial networks (GANs) have received an upsurging interest since being proposed due to the high quality of the generated data. While GANs achieving increasingly impressive results, the resource demands associated with the large model size hinders its usage in resource-limited scenarios. For inference, the existing model compression techniques can reduce the model complexity with comparable performance. However, the training efficiency of GANs has less be explored due to the fragile training process of GANs. In this paper, we for the first time explore the possibility of directly training sparse GAN from scratch without involving any dense or pre-training steps. Even more unconventionally, our proposed method enables training *sparse unbalanced GANs* with an extremely sparse generator in an end-to-end way, chasing high training and inference efficiency gains. Instead of training full GANs, we start by training a sparse subnetwork and periodically explore the sparse connectivity during training, while maintaining a fixed parameter count. Extensive experiments with modern GAN architectures validate the efficiency of our method. Our sparsified GANs, trained from scratch in one single run, outperform the ones learned by expensive iterative pruning and retraining. Perhaps most importantly, we find instead of inheriting parameters from expensive pre-trained GANs, directly training sparse GANs from scratch can be a much more efficient solution. For example, only training with a 80% sparse generator and a 50% sparse discriminator, our method can achieve even better performance than the dense BigGAN.

1 Introduction

The past decade has witnessed impressive results achieved by generative adversarial networks (GANs) (Goodfellow et al., 2014; Zhu et al., 2017; Arjovsky et al., 2017; Miyato et al., 2018; Miyato & Koyama, 2018; Brock et al., 2018; Karras et al., 2017; 2019; 2020). In concert with the improved quality of the generated data, the training and inference costs of the state-of-the-art GANs have also been explored, curbing the application of GANs in edge devices. Reducing computational costs and memory requirements is of importance for many GAN-based applications.

Prior works utilize model compression techniques, such as pruning (Shu et al., 2019), distillation (Li et al., 2020; Chen et al., 2020; Wang et al., 2020b), quantization (Wang et al., 2019), and lottery tickets hypothesis (LTH) (Frankle & Carbin, 2018; Chen et al., 2021b;a) to produce an efficient generator with competitive performance. While increasingly efficient, the existing techniques are not designed to accelerate training as they either operate on fully pre-trained GANs or require the over-parameterized dense GANS to be stored or updated during training. As the resource demands associated with training increases quickly (Strubell et al., 2019), such highly over-parameterized dependency may lead to financial and environmental concerns (Patterson et al., 2021). For example, while the sparse GANs (winning tickets) learned by LTH can match the performance of the dense model, the identification of these winning tickets involves accomplishing the costly train-prune-retrain process many times, resulting in much greater overall FLOPs than training a dense GAN model.

Instead of inheriting knowledge from pre-trained GANs, it is more desirable to train *sparse GANs from scratch* in an end-to-end way (sparse training). What's more, training sparse unbalanced[1]

[1]We refer sparse unbalanced GANs to the scenarios where the sparsities of generators and discriminators are not well-matched.

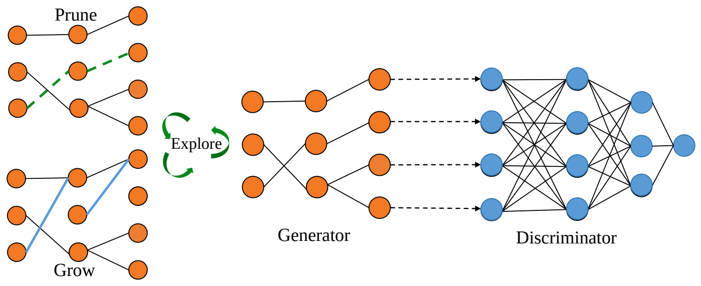

Figure 1: STU-GAN directly trains sparsity-unbalanced GANs from scratch with an extremely sparse generator and a much denser discriminator. The training instability is mitigated by periodically explore parameters for better sparse connectivities with a prune-and-grow scheme during training.

GANs consisting of an extremely sparse generator and a much denser discriminator are arguably more appealing, as such an unbalanced training process will yield extremely sparse generators, very desirable for inference. However, this tantalizing perspective has never been fulfilled due to several daunting challenges. (1) Just like most other deep neural networks (Mocanu et al., 2016; Evci et al., 2019), naively training sparse GANs from scratch without pre-training knowledge typically leads to unsatisfactory performance (Yu & Pool, 2020). Therefore, it remains mysterious whether we can train extremely sparse GANs (even if balanced) to match the performance of the dense equivalents; (2) It is well-known that dense GANs suffer from notorious training instability (Berthelot et al., 2017; Ham et al., 2020). The large sparsity unbalance between generators and discriminators will only make it worse. This naturally raises a question:

How can we train a sparse unbalanced GAN model from scratch without sacrificing performance?

In this paper, we attempt to close this research gap from the perspective of In-Time Over-Parameterization (ITOP) (Liu et al., 2021). ITOP is a concept proposed to understand and improve the expressibility of sparse training (Mocanu et al., 2018; Liu et al., 2020; Evci et al., 2020). Instead of training an over-parameterized dense model, ITOP only requires training a subnet of the dense network from scratch while periodically exploring the parameter space. The variant of over-parameterization – "In-Time Over-Parameterization" (defined as a nearly full exploration of all network parameters during training) along with improved performance is gradually achieved during the course of training in the time series. This finding sheds light on the engaging prospect of ITOP on mitigating the training instability of the sparse unbalanced GANs.

Leveraging the insights from ITOP, we aim to be the first pilot study on training a sparse GAN with unbalanced sparsity between generators and discriminators, without involving any dense or pre-training steps. Our main contributions are:

- We first study the sensitivity of the two most common sparsity-inducing techniques, i.e., pruning and fine-tuning and sparse training from scratch, to the scenario with sparsity unbalance. We empirically find they all severely suffer from the unbalanced sparsity allocation with extremely sparse generators, indicating challenges of sparse unbalanced GAN Training.
- To improve the trainability of sparse unbalanced GANs, we propose an approach termed Sparse Training Unbalanced GAN (STU-GAN). STU-GAN directly trains a sparse unbalanced GAN with an extremely sparse generator and a much denser discriminator from scratch without involving any expensive dense or pre-training steps. By thoroughly exploring the parameter space spanned over the sparse generator, STU-GAN improves its expressibility, and hence stabilizes the atypical training procedure of sparse unbalanced GANs while sticking to a fixed small parameter budget.
- Extensive experiments are conducted with BigGAN (Brock et al., 2018) on CIFAR-10 and ImageNET, SNGAN (Miyato et al., 2018) on CIFAR-10, The consistent performance improvement over the existing techniques verifies the effectiveness of our proposal. Specifically, STU-GAN outperforms dense BigGAN on CIFAR-10 only with a 80% sparse generator and a 70% sparse discriminator, while being end-to-end trainable.

2 Related Work

GAN Compression. While enjoying success in image generation and translation tasks (Karras et al., 2017; Chen et al., 2018; Jing et al., 2019; Gui et al., 2020), generative adversarial networks (GANs), like other deep neural networks, also inherit the undesirable property – high computational complexity and memory requirements. To compress GANs, Shu et al. (2019) proposed a co-evolutionary pruning algorithm to simultaneously pruning redundant filters in both generators. QGAN was proposed by Wang et al. (2019), which can quantize GANs to 2-bit or 1-bit still preserving comparable quality. Distillation was also used by Li et al. (2020) to enhance the compressed discriminator with a pre-trained GAN model. Wang et al. (2020b) moved one step further and combined the above-mentioned three techniques into a unified framework. A trained discriminator is used by (Yu & Pool, 2020) to supervise the training of a compressed generator, achieving compelling results. Very recently, Chen et al. (2021b) extended LTH into GANs, verifying the existence of winning tickets in deep GANs. The existing works on GAN compression all require training an over-parameterized GAN model in advance, not designed for training efficiency. In contrast, our method directly trains a sparse GAN in an end-to-end way, bringing efficiency gains to both training and inference.

In-Time Over-Parameterization. In-Time Over-Parameterization (ITOP) (Liu et al., 2021) refers to a recently emerged topic of (dynamic) sparse training (DST). Mocanu et al. (2018) first proposed sparse evolutionary training (SET) that uses a simple prune-and-regrow scheme to update the sparse connectivity, demonstrating better performance than training with static connectivity (Mocanu et al., 2016; Gale et al., 2019). Following this, weights redistribution are introduced to search for better layer-wise sparsity ratios (Mostafa & Wang, 2019; Dettmers & Zettlemoyer, 2019). While most DST methods use magnitude pruning to remove parameters, there is a large discrepancy between their redistribution criteria. Gradient-based regrowth e.g., momentum (Dettmers & Zettlemoyer, 2019) and gradient (Evci et al., 2020) shows strong results in convolutional neural networks, whereas random regrowth outperforms the former in language modeling (Dietrich et al., 2021). Very recently, Liu et al. (2021) pointed out that the off-the-shell DST methods, intentionally or unknowingly, all perform an "over-parameterization" in the time series by gradually exploring the parameter space spanned over the model. They conjecture that the expressibility of sparse training is highly correlated with the overall number of explored parameters during training. We leverage insights from ITOP to address the severe disequilibrium problem in sparse unbalanced GANs, for the appealing efficiency from training to inference.

3 The Difficulty of Sparse Unbalanced GAN Training

Preliminary and Setups. Generative Adversarial Networks (GANs) consist of a Generator $G(z, \boldsymbol{\theta}_G)$ and a Discriminator $D(x, \boldsymbol{\theta}_D)$. The goal of $G(z, \boldsymbol{\theta}_G)$ is to map a sample z from a random distribution $p(\boldsymbol{z})$ to the data distribution $q_{data}(\boldsymbol{x})$ whereas the goal of $D(x, \boldsymbol{\theta}_D)$ is to determine whether a sample x belongs to the data distribution. Formally, the original dense GANs objective from Goodfellow et al. (2014) is given as follows:

$$\min_{\boldsymbol{\theta}_G} \max_{\boldsymbol{\theta}_D} \mathbb{E}_{x \sim q_{\text{data}}(\boldsymbol{x})}[\log D(x, \boldsymbol{\theta}_D)] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(z, \boldsymbol{\theta}_G)))] \tag{1}$$

where $\boldsymbol{z} \in \mathbb{R}^{d_z}$ is a latent variable drawn from a random distribution $p(\boldsymbol{z})$. Consequently, the objective of sparse GANs can be formaltated as:

$$\min_{\boldsymbol{\theta}_{s_G}} \max_{\boldsymbol{\theta}_{s_D}} \mathbb{E}_{x \sim q_{\text{data}}(\boldsymbol{x})}[\log D(x, \boldsymbol{\theta}_{s_D})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(z, \boldsymbol{\theta}_{s_G})))] \tag{2}$$

where the sparse generator $G(z, \boldsymbol{\theta}_{s_G})$ and sparse discriminator $D(x, \boldsymbol{\theta}_{s_D})$ is parameterized by a fraction of parameters (subnetworks) $\boldsymbol{\theta}_{s_G}$ and $\boldsymbol{\theta}_{s_D}$, respectively. We define sparsity (i.e., fraction of zeros) of $\boldsymbol{\theta}_{s_G}$ and $\boldsymbol{\theta}_{s_D}$ as $s_G = 1 - \frac{\|\boldsymbol{\theta}_{s_G}\|_0}{\|\boldsymbol{\theta}_G\|_0}$ and $s_D = 1 - \frac{\|\boldsymbol{\theta}_{s_D}\|_0}{\|\boldsymbol{\theta}_D\|_0}$, individually, where $\|\cdot\|_0$ is the ℓ_0-norm. We consider unstructured sparsity (individual weights are removed from a network) in this paper, not only due to its promising ability to preserve performance even at extreme sparsities (Frankle & Carbin, 2018; Evci et al., 2020) but also the increasing support for sparse operations on the practical hardware (Gale et al., 2020; Liu et al., 2020; Nvidia, 2020; Zhou et al., 2021).

We first study the effect of sparsity unbalance on two sparsity-inducing techniques applied to GANs, (i) pruning and fine-tuning, and (ii) sparse training from scratch. More specifically, we report Fréchet

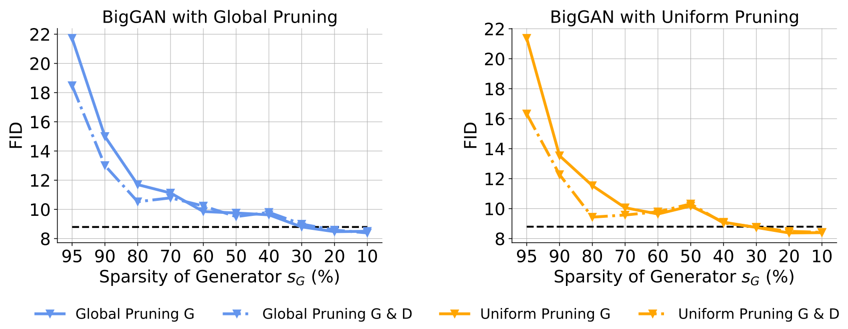

Figure 2: Effect of sparsity unbalance on pruning and fine-tuning. Experiments are conducted with BigGAN on CIFAR-10. Higher s_G refers to fewer parameters remaining in generators. Global pruning refers to pruning weights across layers and uniform pruning refers to pruning layer-wisely.

Inception Distance (FID) achieved by these two methods under two scenarios: (1) only $G(z, \boldsymbol{\theta}_G)$ is sparsified; (2) both $G(z, \boldsymbol{\theta}_G)$ and $D(x, \boldsymbol{\theta}_D)$ are sparsified. The latter has barely been studied due to that existing methods mainly focus on accelerating inference, no need to prune $D(x, \boldsymbol{\theta}_D)$.

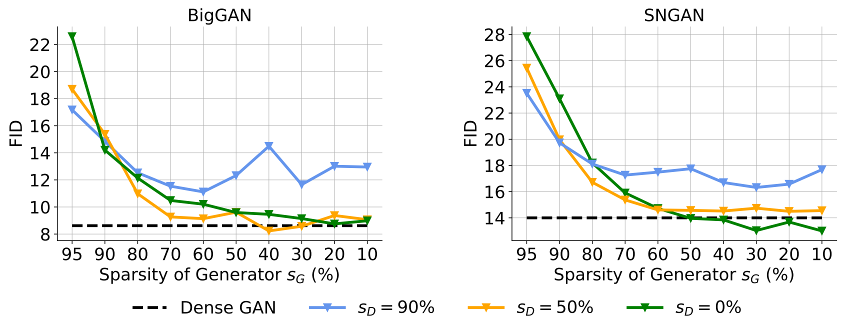

Figure 3: Effect of sparsity unbalance on GAN training. Higher s_G and s_D refer to fewer parameters remaining in the corresponding networks. Global pruning refers to pruning weights across layers and uniform pruning refers to pruning layer-wisely.

Pruning and Fine-tuning. Pruning and fine-tuning, as the most common pruning technique, prunes the pre-trained model to the target sparsity first and then trains it for further t epochs. In this paper, we choose one-shot magnitude pruning (Han et al., 2015b; Frankle & Carbin, 2018), which removes the weights with the smallest magnitude in one iteration of pruning. Doing so makes a fair comparison to sparse training, which does not involve any iterative retraining. Magnitude pruning is performed both uniformly (i.e., weights are removed layer by layer) and globally (i.e., weights are removed across layers). After pruning, the pruned models are further fine-tuned for the same number of epochs as the dense GANs. Intuitively, we should prune and fine-tune the generator together with the discriminator due to the instability of GAN training and hence fine-tuning. Figure 2 shows that pruning and fine-tuning is indeed sensitive to sparsity unbalance. As expected, pruning and fine-tuning $G(z, \boldsymbol{\theta}_G)$ and $D(x, \boldsymbol{\theta}_D)$ together achieve lower FID than only pruning $G(z, \boldsymbol{\theta}_G)$, indicating the vital role of the balanced sparsity in pruning and fine-tuning.

Sparse Training from Scratch. To largely trim down both the training and inference complexity, we prefer training an unbalanced sparse GAN in which $s_G > s_D$. However, balancing the training of dense GANs is already a challenge (Berthelot et al., 2017). Training a sparse unbalanced GAN is even more daunting. To comprehensively understand the effect of sparsity unbalance on GAN training, we directly train sparse GANs without any parameter exploration (static sparse GAN) at various combinations of s_D and s_G. Concretely, we fix $s_D \in [0\%, 50\%, 90\%]$ and vary s_G from 10% to 95%. As shown in Figure 3, the balance of sparsity is essential for GAN training. The low-sparsity discriminator (blue lines) is too strong for the high-sparsity generator whereas the high-sparsity discriminator (green lines) does not have enough capacity to match the results of the dense model. Specifically, GAN training with an extremely sparse generator (i.e., $s_G = 95\%, 90\%$) is a daunting

Algorithm 1 Sparse Training Unbalanced GAN (STU-GAN)

Inputs: Discriminator $D(x, \boldsymbol{\theta}_D)$, Generator $G(z, \boldsymbol{\theta}_G)$, Sparsity of generator s_G, Sparsity of discriminator s_D, initial pruning rate p, update interval ΔT.
Output: Sparse Generator $G(\boldsymbol{z}, \boldsymbol{\theta}_{s_G})$, Sparse discriminator $D(\boldsymbol{x}, \boldsymbol{\theta}_{s_D})$

```
1:  $D(x, \boldsymbol{\theta}_{s_D}) \leftarrow \mathrm{ERK}(D(x, \boldsymbol{\theta}), s_D)$    ▷ Sparse initialization of D$(x, \boldsymbol{\theta}_D)$
2:  $G(z, \boldsymbol{\theta}_{s_G}) \leftarrow \mathrm{ERK}(G(z, \boldsymbol{\theta}), s_G)$    ▷ Sparse initialization of G(z, $\boldsymbol{\theta}_{\mathrm{G}}$)
3:  for $t = 1, 2, \cdots$ do
4:      $\boldsymbol{\theta}_{s_D,t+1} = \mathrm{Adam}\big(\nabla_{\boldsymbol{\theta}_{s_D}} \mathcal{L}_{GAN}(\boldsymbol{\theta}_{s_D,t}, \boldsymbol{\theta}_{s_G,t})\big)$    ▷ Weight optimization of $D(\boldsymbol{x}, \boldsymbol{\theta}_{s_D})$
5:      $\boldsymbol{\theta}_{s_G,t+1} = \mathrm{Adam}\big(\nabla_{\boldsymbol{\theta}_{s_G}} \mathcal{L}_{GAN}(\boldsymbol{\theta}_{s_D,t+1}, \boldsymbol{\theta}_{s_G,t})\big)$    ▷ Weight optimization of $G(\boldsymbol{z}, \boldsymbol{\theta}_{s_G})$
6:      if ($t$ mod $\Delta T$) == 0 then
7:          $\boldsymbol{\theta}_{s_G,t+\frac{1}{2}} = \mathrm{TopK}(|\boldsymbol{\theta}_{s_G,t}|,\ 1-p)$    ▷ Weight exploration of $G(\boldsymbol{z}, \boldsymbol{\theta}_{s_G})$
8:          $\boldsymbol{\theta}_{s_G,t+1} = \boldsymbol{\theta}_{s_G,t+\frac{1}{2}} + \Phi(\boldsymbol{\theta}_{i \notin \boldsymbol{\theta}_{s_G,t+\frac{1}{2}}},\ p)$
9:      end if
10: end for
```

task whose FID raises fastest due to the poor expressibility of the sparse generator. In the rest of paper, we mainly focus on improving the trainability of this daunting but tantalizing setting.

4 Sparse Training Unbalanced GAN (STU-GAN)

We have already known the challenge of training sparse GANs from scratch with the unbalanced sparsity distribution between the generator and discriminator. Inspired by the mechanism behind In-Time Over-Parameterization (Liu et al., 2021), we propose Sparse Training Unbalanced GAN (**STU-GAN**) to close this research gap. The pseudocode of STU-GAN is detailed in Algorithm 1. Instead of training the generator with a static sparse connectivity, STU-GAN dynamically explores the parameter space spanned over the generator throughout training, enhancing the capacity of the highly sparse generator progressively. With the upgraded generator, STU-GAN mitigates the training instability of unbalanced GANs. The parameter exploration is achieved by a prune-and-regrow scheme, which enables STU-GAN to increase the effective weight space of the generator without increasing its parameter count.

Compared with the existing GAN compression techniques, the novelty of STU-GAN is located in: (1) STU-GAN directly trains a sparse GAN from scratch, and thus doesn't require any expensive pre-training steps; (2) STU-GAN starts from a sparse model and maintains the sparsity throughout training, making the approach more suited for edge devices. Specifically, the training process of STU-GAN comprises three main components: sparse initialization, sparse connectivity exploration, and sparse weight optimization, as explained below.

4.1 Sparse Initialization

The choice of the layer-wise sparsity ratios (sparsity of each layer) plays an crucial role for sparse training (Evci et al., 2020). Given that the most the state-of-the-art GANs are constructed based on convolutional neural networks (CNNs), we initialize both $G(z)$ and $D(x)$ with the *Erdős-Rényi-Kernel* (ERK) graph topology (Evci et al., 2020), which automatically allocates higher sparsity to larger layers and lower sparsity to smaller ones. Precisely, the sparsity of each CNN layer l is scaled with $1 - \frac{n^{l-1}+n^l+w^l+h^l}{n^{l-1}\times n^l \times w^l \times h^l}$, where n^l refers to the number of neurons/channels of layer l; w^l and h^l are the width and the height of the convolutional kernel in layer l. For non-CNN layers, the sparsity is allocated with *Erdős-Rényi* (ER) (Mocanu et al., 2018) with $1 - \frac{n^{l-1}+n^l}{n^{l-1}\times n^l}$. ER and ERK typically achieve better performance on CNNs than the naive uniform distribution, i.e., allocating the same sparsity to all layers (Gale et al., 2019).

4.2 Sparse Parameter Exploration

STU-GAN differs from the static sparse GAN training mainly in sparse parameter exploration. Sparse parameter exploration performs prune-and-regrow to searching for better sparse connectivity in generators, which in turn promotes benign competition of the minimax game by improving the quality of counterfeits. Concretely, after every ΔT iteration of training, we eliminate p percentage of parameters with the smallest magnitude from the current sparse subnetwork:

$$\boldsymbol{\theta}_{s,t+\frac{1}{2}} = \mathrm{TopK}(|\boldsymbol{\theta}_{s,t}|,\ (1-p)\cdot N), \tag{3}$$

where $\mathrm{TopK}(v,k)$ returns the weight tensor retaining the top-k elements from v; $\boldsymbol{\theta}_{s,t}$ refers to the sparse subnetwork at t training step; N is the total number of parameters in sparse networks. $\boldsymbol{\theta}_{s,t+\frac{1}{2}}$ is the set of parameters remaining after pruning. While magnitude pruning is simple, we empirically find that it performs better than other more complex pruning criteria such as connectivity sensitivity (Lee et al., 2018), gradient flow (Wang et al., 2020a), and taylor expansion (Molchanov et al., 2016), in the context of sparse training.

To explore new parameters while maintaining the parameter count fixed, we redistribute the same number of pruned parameters back after pruning by:

$$\boldsymbol{\theta}_{s,t+1} = \boldsymbol{\theta}_{s,t+\frac{1}{2}} + \Phi(\boldsymbol{\theta}_{i\notin\boldsymbol{\theta}_{s,t+\frac{1}{2}}},\ p\cdot N) \tag{4}$$

where function $\Phi(v,k)$ refers to growing k weights picked from v based on some certain criterion. $\boldsymbol{\theta}_{i\notin\boldsymbol{\theta}_{s,t+\frac{1}{2}}}$ are the zero elements located in $\boldsymbol{\theta}_{s,t+\frac{1}{2}}$. The redistribution criteria we consider is the gradient redistribution (Evci et al., 2020). Gradient redistribution chooses the parameters with the largest absolute gradient values, indicating the fastest loss reduction in the next iteration. The newly activated weights are initialized as zero to eliminate the historical bias.

This prune-and-regrow scheme performs every ΔT training steps of the generator until a variant of over-parameterization is accomplished at end of the training process, corresponding to the situation where nearly all the parameters of the dense network have been activated. Similar to the synaptic pruning phenomenon (Chechik et al., 1998b;a; Craik & Bialystok, 2006) in biological brains where some connections are strengthened while others are eliminated to learn new experiences, the prune-and-regrow scheme allows the sparse connectivity pattern evolving during training to enhance the sparse generators.

4.3 Sparse Weight Optimization

Sparse Exponential Moving Averages (SEMA) While exponential moving averages (EMA) has shown promising results in prior work (Karras et al., 2017; Yazıcı et al., 2019; Gidel et al., 2019; Mescheder et al., 2018), it becomes less suited for STU-GAN. Since the newly activated weight has no historical information, the original update of EMA $\theta_t^{\mathrm{EMA}} = \beta\theta_{t-1}^{\mathrm{EMA}} + (1-\beta)\theta_t$ ends up with $\theta_t^{\mathrm{EMA}} = (1-\beta)\theta_t$. The large decay factor $\beta = 0.999$ brings the newly activated weights immediately close to zero. To address this problem, we proposed the sparse variant of EMA, SEMA, as following:

$$\theta_{s,t}^{\mathrm{SEMA}} = \begin{cases} 0 & if\ T = 0, \\ \theta_t & if\ T = 1, \\ \beta\theta_{s,t-1}^{\mathrm{SEMA}} + (1-\beta)\theta_t & if\ T > 1. \end{cases} \tag{5}$$

where T refers to the total number of iterations where the weight θ has most recently been activated. In short, SEMA initializes the new activated weights as its original value θ^t instead of $(1-\beta)\theta^t$. Except for this, the activated parameters are optimized with Adam as Goodfellow et al. (2014) in the same way as training dense GANs. The non-activated parameters are forced to be zero before the forward pass and after the backward pass to eliminate their contributions to the loss function.

5 Experimental Results

Experimental Setup. In this section, we conduct experiments to evaluate STU-GAN. Following (Chen et al., 2021b), we choose the widely-used SNGAN (Miyato et al., 2018) on CIFAR-10 for the image generation task. Moreover, to draw a more solid conclusion with large scale GANs, we also evaluate our method with BigGAN (Brock et al., 2018) trained on CIFAR-10 and ImageNet. To enable comparison among different methods, we follow Chen et al. (2021b) and employ two widely-used metrics Fréchet Inception Distance (FID) and Inception Score (IS) as the approximate measure of model performance.

We set the exploration frequency as $\Delta T = 500$ for BigGAN and $\Delta T = 1000$ for SNGAN based on a small random search. The initial pruning rate of weight exploration is $p = 0.5$ for all models, following (Evci et al., 2020; Liu et al., 2021). The original hyperparameters (training epochs, batch size, etc.) and training configurations of GANs are the same as the ones used to train dense GANs[2].

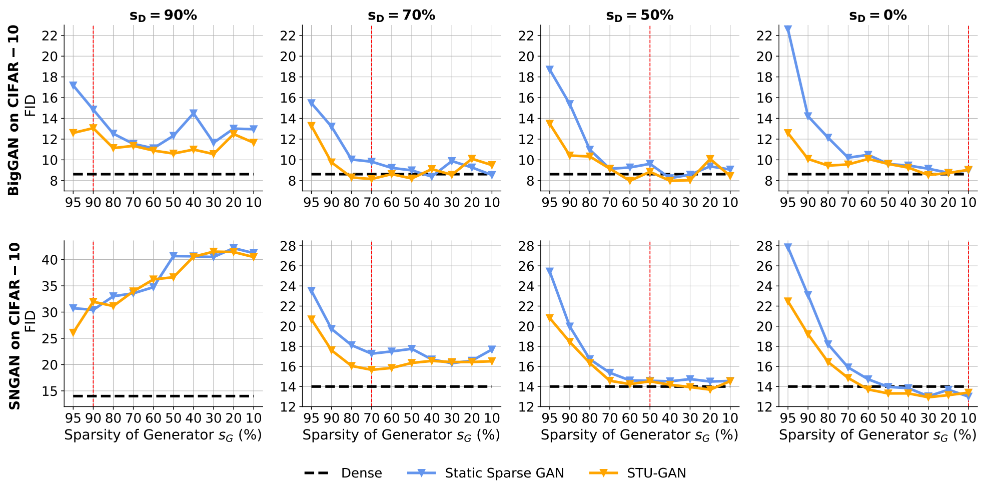

Figure 4: Comparisons between static sparse GAN and STU-GAN with various combinations between s_G and s_D. The sparsity balanced setting with $s_D = s_G$ is indicated with dashed red lines.

5.1 Performance Improvement of STU-GAN over Static Sparse GAN

The most direct baseline of STU-GAN is static sparse GAN, i.e., training a sparse GAN from scratch without any parameter exploration. We implement this baseline in the way that the only difference between STU-GAN and static sparse GAN is the sparse parameter exploration. The comparison is conducted at four settings with different sparsity levels in the discriminator $s_D \in [0.0, 0.5, 0.7, 0.9]$. We do so in the hope of providing insights into GAN training from the perspective of different size of $G(z)$ and $D(x)$ in terms of sparsity. The results are shown in Figure 4. We see that STU-GAN consistently outperforms static sparse GAN with different settings, especially when trained with extremely sparse generators (i.e., higher s_G). Besides this, we further have the following observations:

① **Possibility of improving GAN performance, rather than compromising it.** Applying STU-GAN on BigGAN with a sufficient sparse $D(x, \boldsymbol{\theta}_{s_D})$ (i.e., $s_D = 50\%, 70\%$) can easily yield sparse generators with 20% $\sim$ 90% weights remained while achieving equal or even lower FID compared with the dense GANs (we call such subnetworks matching). Impressively, the combination of a 80% sparse generator and a 70% sparse discriminator is good enough to surpass the performance of the dense GANs. This observation uncovers a very promising finding, that is, instead of training a dense GAN and then compressing it, directly training sparse GANs from scratch can be a better solution when the source budget is not extremely strict, i.e., allowing around 20% $\sim$ 50% weights remaining.

② **STU-GAN substantially stabilizes sparse GAN training, even in the most unbalanced case.** Naively training with an extremely sparse generator and an dense discriminator is nearly impossible due to the notorious training instability of GANs. The generator is too weak to generate high-quality fake data. For instance, under scenarios with extremely sparse generators (i.e., $s_G = 95\%$, first point of every line), performance achieved by static sparse GAN degrades significantly as discriminators get stronger from $s_D = 70\%$ to $s_D = 50\%$, and to $s_D = 0\%$. In contrast, STU-GAN substantially improves the trainability of extremely unbalanced GANs with all s_D. Without using any pre-trained knowledge, STU-GAN decreases FID of BigGAN in the worst case ($s_D = 0\%$ and $s_G = 95\%$) from 22.5 to 12.5, while producing an extremely efficient generator with only 5% parameters.

③ **SNGAN is more sensitive to unbalance sparsity than BigGAN.** SNGAN suffers more from performance loss compared with BigGAN. As discriminators gets sparser, SNGAN has a sharp performance drop. With highly sparse discriminator ($s_D = 90\%, 70\%$), SNGAN can not find any

[2]The training configurations and hyperparameters of BigGAN and SNGAN are obtained from the open-source implementations https://github.com/ajbrock/BigGAN-PyTorch and https://github.com/VITA-Group/GAN-LTH, respectively.

matching subnetworks. This is likely due to the unbalanced architecture design of SNGAN, where the model width (number of filters) of generators is twice the one of discriminators. Over-sparsifying discriminators would further amplify such parameter unbalance, leading to inferior performance. We further confirm this in Appendix A by showing that we can improve the performance under this scenario by applying parameter exploration only to discriminators rather than generators.

5.2 Ablation Study

Which Components Should We Explore? We have learned that parameter exploration in sparse generators improves the expressibility of generators and stabilizes the training of sparse unbalanced GANs. Since parameter exploration is a universal technique that can potentially improve the expressibility of all sparse networks, we expect that its application on discriminators would cause counter-productive results. We evaluate our expectation in Figure 5. The results are on par with our expectations. Simultaneously exploring parameters in $G(z, \boldsymbol{\theta}_{s_G})$ and $D(x, \boldsymbol{\theta}_{s_D})$ achieves worse (i.e., BigGAN) or equal (i.e., SNGAN) performance than solely exploring $G(z, \boldsymbol{\theta}_{s_G})$. However, solely exploring parameters of $D(x, \boldsymbol{\theta}_{s_D})$ leads to much higher FID.

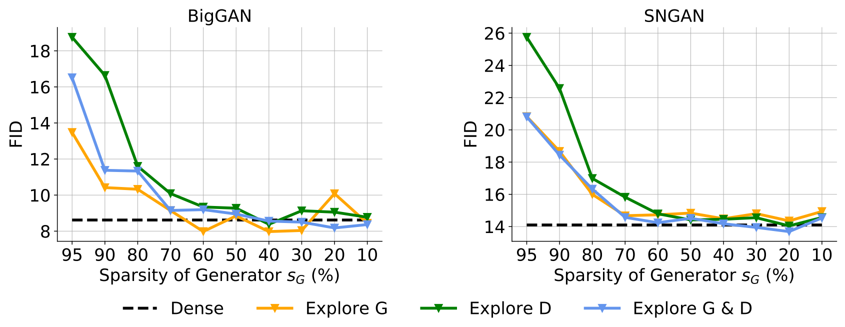

Figure 5: Effect of parameter exploration on different components. “Explore G”, “Explore D”, and “Explore G & D” refers to applying parameter exploration only to generators, discriminators, and both, respectively. Experiments are conducted with 50% sparse discriminators, i.e., $s_D = 50\%$.

Effect of Exploration Frequency. The exploration frequency ΔT (i.e., the number of training steps between two iterations of parameter exploration) directly controls the trade-off between the quality and quantity of the parameter exploration in STU-GAN. Smaller ΔT means more iterations of exploration, which ends up with a larger range of activated parameters. On the contrary, larger ΔT allows the subnetworks between two exploration to be well-trained, improving the correctness of the parameter exploration. We report the trade-off in Figure 6. We see that the best performance is obtained when ΔT is set around 1000. BigGAN seems to be more robust to the exploration frequency compared with SNGAN.

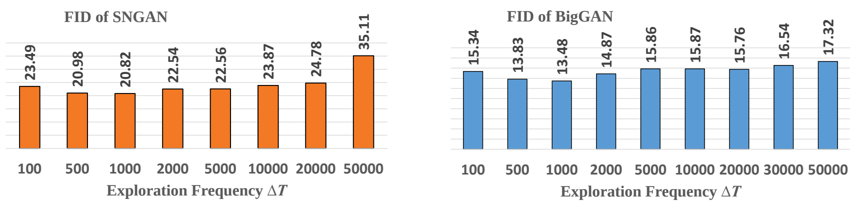

Figure 6: FID of sparse unbalanced BigGAN and SNGAN on CIFAR-10 with various ΔT. The results of BigGAN in trained with 95% sparse $D(x, \boldsymbol{\theta}_{s_D})$ and 95% sparse $G(z, \boldsymbol{\theta}_{s_G})$. The results of SNGAN in trained with 50% sparse $D(x, \boldsymbol{\theta}_{s_D})$ and 95% sparse $G(z, \boldsymbol{\theta}_{s_G})$.

Comparison with Stronger Baselines. We compare STU-GAN with various baselines including static sparse GAN, pruning and fine-tuning (PF (Han et al., 2015a)), and GAN Tickets (Chen et al., 2021b). Static sparse GAN refers to naively training sparse unbalanced GAN with the fixed sparse pattern, whereas PF and GAN Tickets are two post-training pruning methods that operate on a pre-trained model. GAN Tickets is the recently proposed strong baseline, which discovers performing

Table 1: (FID (↓), IS (↑)) of sparse BigGAN and SNGAN on CIFAR-10. We divide sparse methods into two groups, post-training pruning (PF and GAN Tickets) and sparse training (Static Sparse GAN and STU-GAN). "$s_D(\%)$" and "$s_G(\%)$" refers to the sparsity of the discriminator and the generator used to train GANs, respectively. Results of GAN Tickets are obtained from Chen et al. (2021b). The exact sparsity of GAN Tickets at $s_G = 90\%$ is 87%. The best sparse results are bolded.

Methods	SNGAN				BigGAN			
Sparsity	$s_D(\%)$	$s_G = 95\%$	$s_G = 90\%$	$s_G = 80\%$	$s_D(\%)$	$s_G = 95\%$	$s_G = 90\%$	$s_G = 80\%$
Dense GAN	0	(14.01, 8.26)	(14.01, 8.26)	(14.01, 8.26)	0	(8.62, 8.90)	(8.62, 8.90)	(8.62, 8.90)
PF (global)	0	(26.34, 7.31)	(17.97, 7.85)	(16.99, 8.03)	0	(18.48, 8.04)	(13.01, 8.51)	(10.52, 8.66)
PF (uniform)	0	(27.35, 7.27)	(18.23, 7.76)	(16.93, 8.04)	0	(16.32, 8.23)	(12.27, 8.50)	(9.43, 8.73)
GAN Tickets	0	n/a	(19.29, 8.07)	(16.79, 8.16)	0	n/a	(9.87, 8.75)	(9.06, 8.87)
Static Sparse GAN	50	(25.43, 7.58)	(19.97, 7.78)	(16.71, 8.01)	50	(18.70, 7.82)	(15.37, 8.12)	(10.98, 8.47)
Static Sparse GAN	70	(23.52, 7.56)	(19.73, 7.69)	(18.11, 7.93)	70	(15.46, 8.13)	(13.20, 8.32)	(10.03, 8.64)
STU-GAN (ours)	50	(22.34, 7.71)	(18.78, 7.94)	(16.32, 8.01)	50	(13.47, 8.19)	(10.42, 8.57)	(10.34, 8.61)
STU-GAN (ours)	70	**(20.67, 7.77)**	**(17.60, 8.07)**	**(16.04, 8.18)**	70	**(13.27, 8.37)**	**(9.75, 8.80)**	**(8.57, 8.91)**

subnetworks by adopting iterative pruning on a fully trained a dense GAN. We set the number of training epochs of STU-GAN and Static Sparse GAN the same as the dense GANs. As shown in Table 1, STU-GAN consistently achieves the best performance with only 30% parameters remaining in the discriminators. Even though trained from scratch, STU-GAN outperforms those methods that require expensive iterative pruning and retraining, highlighting the superiority of our method in the trade-off between performance and efficiency.

Boosting STU-GAN with Full In-Time Over-Parameterization. Following (Liu et al., 2021), we test if achieving the full "In-Time Over-Parameterization", i.e., extending the training time to explore almost all parameters of the generators, can lead to better performance. We extend training steps by 5 times same as Evci et al. (2020) and report the best FID in Table 2. We see that extending training steps consistently brings performance gains to sparse unbalanced GAN.

Table 2: (FID (↓), IS (↑)) of sparse BigGAN and SNGAN on CIFAR-10 with 5× training steps.

Methods	SNGAN				BigGAN			
Sparsity	$s_D(\%)$	$s_G = 95\%$	$s_G = 90\%$	$s_G = 80\%$	$s_D(\%)$	$s_G = 95\%$	$s_G = 90\%$	$s_G = 80\%$
STU-GAN	50	(22.34, 7.71)	(18.78, 7.94)	(16.32, 8.01)	50	(13.47, 8.19)	(10.42, 8.57)	(10.34, 8.61)
STU-GAN$_{5\times}$	50	**(18.44, 7.97)**	**(16.96, 7.98)**	**(15.68, 8.04)**	50	**(12.92, 8.44)**	**(10.01, 8.62)**	**(9.84, 8.71)**

Performance on ImageNet. To draw more solid conclusions, we evaluate STU-GAN with BigGAN on ImageNet, a complex dataset with high-resolution 128×128 and diverse samples. We compare STU-GAN with densely trained BigGAN with half of filters in the generator and the discriminator (Small Dense). We train sparse BigGAN with STU-GAN at sparsity of $s_G \in [95\%, 80\%, 60\%]$ and $s_D = 50\%$ for comparison. The results are shown in Table 3. We see that our method improves the corresponding dense equivalent by 2.73 FID and 15.24 IS score with a similar parameter count. Very impressively, STU-GAN can still match the performance when only 5% parameters are remained in the generator, suggesting its substantial parameter efficiency.

Table 3: FID (↓) and IS (↑) of sparse BigGAN on ImageNet 128×128 without the truncation trick.

Methods	$s_D(\%)$	$s_G(\%)$	FID (↓)	IS (↑)
Small Dense	50	50	13.56	58.90
STU-GAN	50	95	13.94	60.49
STU-GAN	50	80	11.20	70.28
STU-GAN	50	60	**10.83**	**74.14**

6 Conclusion

In this paper, we study GAN training from the perspective of sparsity. We demonstrate that the well-matched sparsity between generators and discriminators is essential to GAN training, whereas the sparsity unbalance scenarios significantly degrades the trainability of sparse GNA. We further explore the possibility of training sparsity-unbalanced GAN with an extremely sparse generator and a much denser discriminator by proposing Sparse Training Unbalanced GAN (STU-GAN). Training and maintaining only a small fraction of parameters without involving any pre-training, STU-GAN can outperform several strong after-training pruning techniques, shedding light on the appealing prospect of sparsity to stabilize GAN training.

7 Ethics Statement

In this paper, we propose STU-GAN to enable training sparse GANs from scratch. Since our method significantly reduces the number of parameters required to train GANs, we do not see any negative effects on our society. Perhaps one limitation of our method is that our sparsity-oriented algorithm has not been fully supported by the current hardware. Still, our method provides good motivation for future hardware to enable efficient sparse operations. Once this great potential is supported by future hardware, it can provide a significant positive impact on our planet by saving a huge amount of energy and reducing overall total carbon emissions.

8 Reproducibility

We use the BigGAN and SNGAN implementation from the BigGAN-PyTorch repository (`https://github.com/ajbrock/BigGAN-PyTorch`) and GAN-LTH repository (`https://github.com/VITA-Group/GAN-LTH`), respectively. We directly trained our models with the default hyperparameters and configurations in the repository. For hyperparameters induced by our methods, we have reported them at the beginning of Section 5. The FID and IS of CIFAR-10 are calculated with the TensorFlow Inception code. The FID and IS of ImageNet are obtained with the PyTorch.

References

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

David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Neuronal regulation: A mechanism for synaptic pruning during brain maturation. *Neural Computation*, 11:11–8, 1998a.

Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Synaptic pruning in development: A computational account. *Neural Comput*, 10:2418–2427, 1998b.

Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. Distilling portable generative adversarial networks for image translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3585–3592, 2020.

Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Ultra-data-efficient gan training: Drawing a lottery ticket first, then training it toughly. *arXiv preprint arXiv:2103.00397*, 2021a.

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

Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9465–9474, 2018.

Fergus I. M. Craik and Ellen Bialystok. Cognition through the lifespan: Mechanisms of change. In *Trends in Cognitive Sciences*, pp. 131–138, 2006.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Anastasia Dietrich, Frithjof Gressmann, Douglas Orr, Ivan Chelombiev, Daniel Justus, and Carlo Luschi. Towards structured dynamic sparse pre-training of bert. *arXiv preprint arXiv:2108.06277*, 2021.

Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14. IEEE, 2020.

Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1laEnA5Ym.

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

Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *arXiv preprint arXiv:2001.06937*, 2020.

Hyungrok Ham, Tae Joon Jun, and Daeyoung Kim. Unbalanced gans: Pre-training the generator of generative adversarial network using variational autoencoder. *arXiv preprint arXiv:2002.02112*, 2020.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019.

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.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *International Conference on Learning Representations*, 2018.

Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5284–5294, 2020.

Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 2020.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 6989–7000. PMLR, 2021.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

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

Decebal Constantin Mocanu, Elena Mocanu, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. A topological insight into restricted boltzmann machines. *Machine Learning*, 104(2-3): 243–270, 2016.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, 2018.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *International Conference on Learning Representations*, 2016.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, 2019.

Nvidia. Nvidia a100 tensor core gpu architecture. *https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf*, 2020.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3235–3244, 2019.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=SkgsACVKPH.

Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *European Conference on Computer Vision*, pp. 54–73. Springer, 2020b.

Peiqi Wang, Dongsheng Wang, Yu Ji, Xinfeng Xie, Haoxuan Song, XuXin Liu, Yongqiang Lyu, and Yuan Xie. Qgan: Quantized generative adversarial networks. *arXiv preprint arXiv:1901.08263*, 2019.

Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJgw_sRqFQ.

Chong Yu and Jeff Pool. Self-supervised gan compression. *arXiv preprint arXiv:2007.01491*, 2020.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=K9bw7vqp_s.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A Poor Trainability of SNGAN with Extremely Sparse Discriminator

As we showed in Section 5.1, sparse SNGAN with an extremely sparse discriminator severely suffers from poor trainability. Here, we conjecture that this is likely due to the unbalanced architecture design of SNGAN, where the model width (number of filters) of generators is twice of discriminators. Over-sparsifying discriminators would further amplify such parameter unbalance. We evaluate our conjecture by only exploring the parameters of discriminators. Doing so improves the expressibility of the extremely sparse discriminators, and hence improves the performance. As shown in Figure 7, only exploring discriminators leads to lower FID than only exploring generators, in line with our conjecture. Yet, the expressibility improvement caused by parameter exploration can not fully address the over-pruning issue caused by the original unbalanced architecture design.

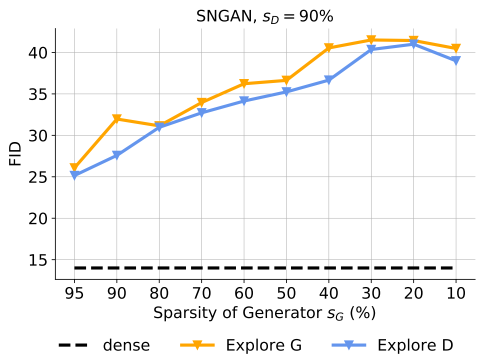

Figure 7: FID of sparse SNGAN trained with extremely sparse discriminator, i.e., $s_D = 90\%$. "Explore G" refers to only applying parameter exploration to the generators. "Explore D" refers to only applying parameter exploration to the discriminators.

B EXAMPLE IMAGES GENERATED BY STU-GAN ON IMAGENET

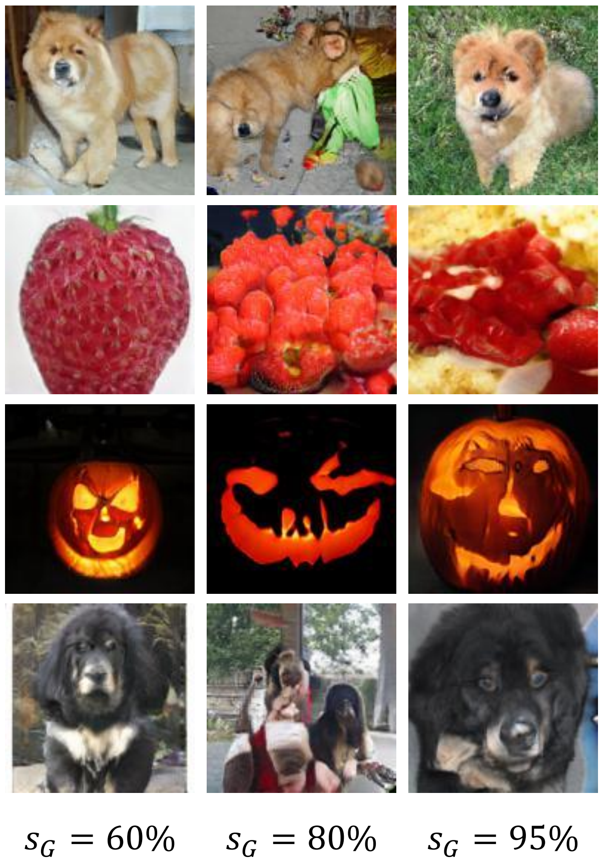

Figure 8: Example Images Generated by STU-GAN on ImageNet 128 × 128. Each row of images are generated via STU-GAN models with various s_G and $s_D = 50\%$.