Slimmable Generative Adversarial Networks

Liang Hou,[1,2*] **Zehuan Yuan,**[3†] **Lei Huang,**[4] **Huawei Shen,**[1,2†] **Xueqi Cheng,**[1,2] **Changhu Wang**[3]
[1]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]ByteDance AI Lab
[4]SKLSDE, Institute of Artificial Intelligence, Beihang University
{houliang17z, shenhuawei, cxq}@ict.ac.cn, {yuanzehuan, wangchanghu}@bytedance.com, huanglei@nlsde.buaa.edu.cn

Abstract

Generative adversarial networks (GANs) have achieved remarkable progress in recent years, but the continuously growing scale of models makes them challenging to deploy widely in practical applications. In particular, for real-time generation tasks, different devices require generators of different sizes due to varying computing power. In this paper, we introduce slimmable GANs (SlimGANs), which can flexibly switch the width of the generator to accommodate various quality-efficiency trade-offs at runtime. Specifically, we leverage multiple discriminators that share partial parameters to train the slimmable generator. To facilitate the *consistency* between generators of different widths, we present a stepwise inplace distillation technique that encourages narrow generators to learn from wide ones. As for class-conditional generation, we propose a sliceable conditional batch normalization that incorporates the label information into different widths. Our methods are validated, both quantitatively and qualitatively, by extensive experiments and a detailed ablation study.

Introduction

One of the main reasons for the tremendous success of deep learning in recent years is the increasing scale of models. In the branch of deep generative models, generative adversarial networks (GANs) (Goodfellow et al. 2014) have received widespread attention and evolved from the original simple multi-layer perceptrons to the vast BigGAN framework (Brock, Donahue, and Simonyan 2019) with residual blocks (He et al. 2016) and self-attention layers (Zhang et al. 2019) to synthesize realistic images nowadays. The arms race on increasing the model size is endless, while the computational power and budget of devices are limited, especially for mobile phones. Several GAN applications such as photograph (Kupyn et al. 2018) and autonomous driving (Zhang et al. 2018) require short response time and hopefully run on devices with limited computing power. Recently, researchers began to develop lightweight GAN models. However, different devices usually require customized models of different sizes to meet the given response time budget. Moreover, even a single device needs models of different sizes due to several switchable performance modes, e.g., the high-performance mode and power-saving mode. Consequently, numerous models need to be trained and deployed for a single task, which is also heavy work.

In this work, we are committed to developing a "once-for-all" generator, which we only train and deploy once but can flexibly switch the model size at runtime to address the practical challenges. Inspired by slimmable neural networks (SNNs) (Yu et al. 2019), we focus on developing a generator with configurable widths, where the width refers to the number of channels in layers. In addition to saving inference time, customization on width can reduce memory footprint during the layer-by-layer inference, while reducing depth cannot take this advantage.

Although several discriminative tasks such as image classification and object detection are well studied in SNNs, applying slimmable operators to GANs suffers from three following challenges: First, how to accurately and appropriately estimate the divergence between generators at different widths and the real data through discriminators? Second, how to ensure *consistency* between generators of different widths? Here, the *consistency* means that the generated images should be similar between these generators given the same latent code. Third, how to incorporate the label information into generators at different widths in the class-conditional generation?

In this paper, we propose slimmable generative adversarial networks (SlimGAN) to combat the aforementioned problems. First, we present discriminators with partially shared parameters to serve the generators at different widths. Second, to improve the consistency between generators at different widths, we introduce a novel stepwise inplace distillation technique, which encourages narrow generators to learn from the wide generators. Third, we propose a sliceable conditional batch normalization (scBN) to incorporate the label information into different widths on the basis of switchable batch normalization (sBN) (Yu et al. 2019) for the class-conditional generation. Extensive experiments across several real-world datasets and two neural network backbones demonstrate that SlimGAN can compete with or even outperform the individually trained GANs. Remarkably, our proposed scBN achieves better performance with fewer parameters. A systematic ablation study verifies the effectiveness of our design, including network framework and loss function.

[*] Work done as an intern at ByteDance AI Lab.

[†] Corresponding authors

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Related Work

Generative Adversarial Networks

Generative adversarial networks (GANs) (Goodfellow et al. 2014) were implemented by multi-layer perceptrons at the beginning. To improve the capability of the generator and the discriminator, convolutional layers were introduced in DCGAN (Radford, Metz, and Chintala 2015). Later, WGAN-gp (Gulrajani et al. 2017) not only established flexible Lipschitz constraints but also brought the ResNet (He et al. 2016) backbone into the GAN literature. To further impose the Lipschitz constraint, SNGAN (Miyato et al. 2018) introduced spectral normalization to the discriminator, which is also applied to the generator in SAGAN (Zhang et al. 2019). For class-conditional generation tasks, cGAN-pd (Miyato and Koyama 2018) injected the label information to the generator by employing conditional batch normalization (cBN) (de Vries et al. 2017), and the discriminator with projection technique. Recently, BigGAN (Brock, Donahue, and Simonyan 2019) was capable of generating diverse and realistic high-resolution images, mainly attributed to the massive model.

Model Compression in GANs

The arms race on developing increasingly bloated network architecture hinders the extensive deployment of GANs in practical applications. To reduce the size of the generator, Aguinaldo et al. (2019) compressed GAN models using knowledge distillation techniques. Li et al. (2020) proposed a compression method for conditional GAN models. Meanwhile, Yu and Pool (2020) developed a self-supervised compression method that uses the trained discriminator to supervise the training of a compressed generator. AutoGAN-Distiller (Fu et al. 2020) compressed GAN models using neural architecture search. Recently, Wang et al. (2020a) developed a unified GAN compression framework, including model distillation, channel pruning, and quantization.

Dynamic Neural Networks

Unlike model compression, dynamic neural networks can adaptively choose the computational graph to reduce computation during training and inference. For example, Liu and Deng (2018) presented an additional controller network to decide the computational graph depends on the input. Similarly, Hu et al. (2019) proposed to reduce the test time by introducing an early-exit gating function. Different from adjusting the depth of neural networks, slimmable neural networks (SNNs) (Yu et al. 2019) trained neural networks that can be executable at different widths, allowing immediate and adaptive accuracy-efficiency trade-offs at runtime. Later, US-Net (Yu and Huang 2019b) extended SNN to universally slimmable scenarios and proposed improved training techniques. AutoSlim (Yu and Huang 2019a) utilized model pruning methods to obtain accuracy-latency optimal models but introduced additional storage consumption. RS-Nets (Wang et al. 2020b) proposed an approach to train neural networks which can switch image resolutions during inference.

Nevertheless, the aforementioned approaches are designed for discriminative tasks with a single neural network, while we focus on generative tasks based on GANs. Since GAN consists of two networks, i.e., the generator and discriminator network, modifying the operational mechanism of the generator may destroy the stability of the entire system, which makes the training process of GAN with a slimmable generator challenging.

Preliminaries

Generative Adversarial Networks

Generative adversarial networks (GANs) (Goodfellow et al. 2014) are typically composed of a generator and a discriminator. Specifically, the generator $G : \mathcal{Z} \to \mathcal{X}$ learns to generate fake samples by mapping a random noise vector $z \in \mathcal{Z}$ in the latent space endowed with a predefined prior $\mathcal{P}_Z$ (e.g., multivariate normal distribution) to a sample $x \in \mathcal{X}$ in the high-dimensional complex data space. The discriminator $D : \mathcal{X} \to [0, 1]$ attempts to distinguish the synthetic examples generated by the generator from real data. In contrast, the goal of the generator is to fool the discriminator by mimicking real data. Formally, the objective function of GAN is formulated as follows:

$$\begin{aligned}\min_G \max_D & \mathbb{E}_{x\sim\mathcal{P}_{\text{data}}}[\log(D(x))]+ \\ & \mathbb{E}_{z\sim\mathcal{P}_Z}[\log(1-D(G(z)))],\end{aligned} \tag{1}$$

where $\mathcal{P}_{\text{data}}$ represents the underlying distribution of real data. As proved in (Goodfellow et al. 2014), this minimax game is considered as minimizing the Jansen Shannon (JS) divergence between the real data distribution and the generated one. Ideally, the generator is supposed to converge until $\mathcal{P}_G = \mathcal{P}_{\text{data}}$. The JS divergence estimated by the discriminator can be replaced with other f-divergences (Nowozin, Cseke, and Tomioka 2016) or even true metrics such as Wasserstein distance (Arjovsky, Chintala, and Bottou 2017) by modifying the objective function.

Slimmable Neural Networks

Slimmable neural networks (SNNs) (Yu et al. 2019) can instantly adjust the network width according to the demands of various devices with different capacities. Unlike other training lightweight model methods such as neural architecture search and model compression, SNN is more flexible because it only needs to be trained and deployed once to obtain multiple models at different widths from a pre-specified width list $\mathcal{W}$. In order to avoid the discrepancy of mean and variance between networks at different widths, SNN proposed a switchable batch normalization (sBN), i.e., using independent BN learnable parameters for each width:

$$x'_{w_i} = \gamma_{w_i}\frac{x_{w_i} - \mu(x_{w_i})}{\sigma(x_{w_i})} + \beta_{w_i}, \tag{2}$$

where x_{w_i} represents the data batch at current width $w_i \in \mathcal{W}$. Specifically, $\mu(\cdot)$ and $\sigma(\cdot)$ compute the mean and standard deviation of this batch, γ_{w_i} and β_{w_i} are learnable scale and shift, respectively, of the sBN at width w_i.

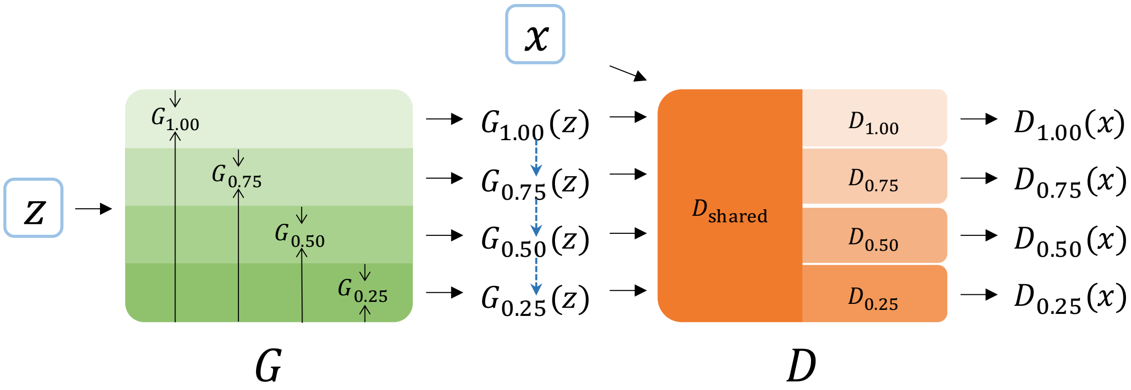


Figure 1: Illustration of SlimGAN with width multiplier list $\mathcal{W} = [0.25, 0.5, 0.75, 1.0]\times$. Wide generators contain the channels of narrow ones. Multiple discriminators share first several layers. Blue dashed lines indicate the stepwise inplace distillation.

Methods

We aim to develop a size-flexible generator that can switch its size to accommodate various computing power. Approximatively, the size-flexible generator implies multiple generators: $G_{\theta_1}, G_{\theta_2}, \cdots, G_{\theta_N}$ with N incremental parameters $\theta_1 \subseteq \theta_2 \subseteq \cdots \subseteq \theta_N$, respectively. In this work, we focus on slimming the width (number of channels) of the generator network instead of depth as reducing width can save memory footprint during the layer-by-layer inference. The width-slimmable generator contains several generators: $G_{w_1}, G_{w_2}, \cdots, G_{w_N}$ at $N = |\mathcal{W}|$ incremental widths $w_1 < w_2 < \ldots < w_N$ ($w_i \in \mathcal{W}$), respectively. Particularly, we train the generator via adversarial training and call our method slimmable GAN (SlimGAN) [1].

Slimmable GAN Framework

We illustrate the overall framework of SlimGAN in Figure 1. Specifically, the SlimGAN consists of a slimmable generator with multi-width configurations and multiple discriminators that share the first several layers. Each discriminator guides the generator at the corresponding width. Here, using multiple shared discriminators, instead of a single discriminator or multiple independent discriminators, is critical for our SlimGAN model. This is also the first major novelty of this model. The idea is motivated by two insights. On one hand, using a single discriminator for all the generators with different widths limits the flexibility and capability of discriminators to discriminate generated data from real data, and finally fails to obtain well-performed generators. On the other hand, although assigning one discriminator for each generator offers high flexibility, it is incapable of leveraging the characteristic of data generated by slimmable generators. Therefore, we borrow the idea of multi-task learning and design multiple parameter-shared discriminators. This design not only offers high flexibility of discriminators but also leverages the similar characteristic of data generated by slimmable generators to improve the training of generators. In addition, sharing parameters with other tasks offers a kind of consistency regularization on discriminators, which potentially improves the generalization of discriminators, and hence promotes the performance of generators (Thanh-Tung, Tran, and Venkatesh 2019).

As for training the generator-discriminator pair at width w_i, we utilize the Hinge version loss (Lim and Ye 2017; Tran, Ranganath, and Blei 2017), which is prevalent and successful in GAN literature.

$$\begin{aligned} &\max_D \mathbb{E}_{x\sim\mathcal{P}_{\text{data}}}[\min(0, -1 + D_{w_i}(x))] + \\ &\qquad \mathbb{E}_{z\sim\mathcal{P}_Z}[\min(0, -1 - D_{w_i}(G_{w_i}(z)))] \\ &\max_G \mathbb{E}_{z\sim\mathcal{P}_Z}[D_{w_i}(G_{w_i}(z))], i = 1, 2, \cdots, N \end{aligned} \tag{3}$$

Stepwise Inplace Distillation

Although a single slimmable generator implies multiple subgenerators, we expect these generators to maintain the *consistency* between them, like an identical generator. Imagine that, a trained slimmable generator is deployed as clients on various devices, and these devices may choose different width configurations according to their diverse energy budgets. We expect these clients to generate consistent samples for the same command (e.g., the latent code z), which is broadcasted by the server. We characterize this requirement as *spatial translation consistency*. In addition, since a single device has different performance modes, e.g., high-power mode or power-saving mode, even the same device may choose generators of different sizes. We also expect this device to generate a consistent sample for the same latent code at any mode, which is considered as *time translation consistency*. However, the adversarial training objective function cannot explicitly guarantee the *consistency* between generators of different widths because the discriminator only distinguishes real from fake but not distinguishes similar from dissimilar.

To achieve consistency, we propose a novel stepwise inplace distillation technique. Different from general-purpose model distillation, we do not utilize knowledge distillation to obtain a smaller model through an already trained one. Instead, we train narrow networks by encouraging them to learn from wide networks during the training process,

[1] Code is available at https://github.com/houliangict/SlimGAN

thereby improving consistency between them. Specifically, the proposed distillation first distills the full generator to the second widest one and then distills the second one to the third one and so on. We employ the pixel mean square error as the objective function in the distillation:

$$\min_{G} \frac{\lambda}{N-1}\mathbb{E}_{z\sim\mathcal{P}_Z}\sum_{i=1}^{N-1}\|G_{w_i}(z)-\mathrm{sg}(G_{w_{i+1}}(z))\|_2^2, \tag{4}$$

where λ is a hyper-parameter that balances the adversarial objectives and the distillation, and $\mathrm{sg}(\cdot)$ means to stop the transfer of gradients in the computational graph. Stop updating the wide generator in distillation prevents it from learning from the narrow one.

Arguably, the distillation can effectively improve the performance of narrow networks. Furthermore, the improvement of narrow networks could also lead to an enhancement of wide networks, because wide generators contain all the channels of narrow generators, which forms a virtuous cycle in SlimGAN. As an alternative, leveraging the full network to teach all narrow generators, however, may be contrary to the assumption of width residuals (Yu and Huang 2019b). In other words, forcing all narrow generators to learn from the widest one would make no difference between them, which may tend to strengthen the expression of parameters they shared but reduce the capability of their specific.

Training Algorithm

Algorithm 1 shows the training procedure of SlimGAN in PyTorch-style pseudo-code. The main difference from training a normal GAN is that we enumerate all the widths in the pre-specified width list at each iteration and switch the computational graph according to the configured width. In the adversarial training part, we sample random noises as the input of each generator. This provides the diversity of fake samples, encouraging models to explore wider optimization space to achieve better results. In the consistency training part, we sample the same latent code to optimize the discrepancy of the outputs between generators at different widths.

Sliceable Conditional Batch Normalization

In the case of class-conditional generation, state-of-the-art class-conditional GANs, e.g., BigGAN (Brock, Donahue, and Simonyan 2019), follow the way of incorporating label information proposed in cGAN-pd (Miyato and Koyama 2018), i.e., conditional batch normalization (cBN) in the generator and projection in the discriminator. In this work, we follow the label projection technique in the discriminator. As for the generator, however, how to introduce the label information under the width-switchable mechanism is the key problem faced by SlimGAN in the class-conditional generation scenario. In other words, how to unify sBN and cBN? A naive way to achieve this goal is to expand each sBN to a cBN:

$$x'_{w_i,c_j} = \gamma_{w_i,c_j}\frac{x_{w_i,c_j}-\mu(x_{w_i,c_j})}{\sigma(x_{w_i,c_j})} + \beta_{w_i,c_j}, \tag{5}$$

where c_j indicates the current label. However, the disadvantages of this design are obvious from two perspectives. First, the number of parameters increased dramatically because of $N \times C$ BN parameters (C is the number of labels), which is contradictory to our motivation, i.e., saving parameters to reduce model size and computation. Second, the information of the same label is separated for generators at different widths.

Algorithm 1 Training SlimGAN

```
Require: dataset $\mathcal{D}$, switchable width multiplier list $\mathcal{W}$
Ensure: generator $G$
for $t = 1, \dots, T$ do
    for $k = 1, \dots, K$ do
        Get mini-batch data, $x = sample(\mathcal{D})$
        for $i = 1, \dots, N$ do
            Generate samples, $\hat{x} = G_{w_i}(z)$ with $z \sim \mathcal{P}_Z$
            Comp. D loss, $loss = lossD(D_{w_i}(x), D_{w_i}(\hat{x}))$
            Compute D gradients, $loss.backward()$
        end for
        Update D weights, $optimizerD.step()$
    end for
    Sample fixed noise $\bar{z} \sim \mathcal{P}_Z$ and initialize $\bar{x} = [\,]$
    for $i = 1, \dots, N$ do
        Generate samples, $\hat{x} = G_{w_i}(z)$ with $z \sim \mathcal{P}_Z$
        Compute G loss, $loss = lossG(D_{w_i}(\hat{x}))$
        Compute G gradients, $loss.backward()$
        Generate fixed samples $\bar{x}.append(G_{w_i}(\bar{z}))$
    end for
    Compute distillation loss, $loss = lossDistill(\bar{x})$
    Compute distillation gradients, $loss.backward()$
    Update G weights, $optimizerG.step()$
end for
return $G$
```

To remedy the above issues, we propose a sliceable conditional batch normalization (scBN) defined as follows:

$$x'_{w_i,c_j} = \gamma_{w_i}\gamma_{c_j}^{:s_i}\frac{x_{w_i,c_j}-\mu(x_{w_i,c_j})}{\sigma(x_{w_i,c_j})} + \beta_{w_i} + \beta_{c_j}^{:s_i}, \tag{6}$$

where γ_{c_j} and β_{c_j} are the learnable parameters of the cBN with label c_j. To incorporate the label embedding into different widths, we slice cBN vectors to sub-vectors with the first $s_i = |\gamma_{w_i}|$ elements (s_i is the number of channels in the layer at current width w_i). Since cBN and sBN are independent, there are $N + C$ BN parameters in our proposed scBN, which not only accordingly reduces the parameters but also explicitly shares the information of the same label.

Experiments

In this section, we first evaluate our proposed SlimGAN across several datasets with two network backbones, compared with the individually trained models. We then conduct class-conditional generation experiments to verify the effectiveness of scBN. Besides, we report the qualitative and quantitative results that indicate the consistency between generators at different widths. We further demonstrate the design of SlimGAN through an extensive ablation study. We finally analyze the parameters complexities of generators.

Backbone	Dataset	Method	FID (↓)				IS (↑)			
			0.25×	0.5×	0.75×	1.0×	0.25×	0.5×	0.75×	1.0×
DCGAN (uncond)	CIFAR-10	Individual	46.9	34.6	30.4	26.7	6.08	6.95	7.39	7.43
		Slimmable	**37.3**	**28.5**	**25.8**	**25.2**	**6.90**	**7.31**	**7.43**	**7.44**
	STL-10	Individual	93.1	69.1	61.8	57.4	6.51	7.82	7.96	8.38
		Slimmable	**68.9**	**60.9**	**56.2**	**55.1**	**7.67**	**8.00**	**8.34**	**8.38**
	CelebA	Individual	24.4	13.2	10.4	9.8	-	-	-	-
		Slimmable	**23.3**	13.3	10.6	**9.4**	-	-	-	-
ResNet (uncond)	CIFAR-10	Individual	41.8	24.1	21.6	20.3	7.36	7.68	7.93	7.91
		Slimmable	**29.9**	**21.6**	**19.6**	**20.0**	7.32	**8.02**	**8.15**	**8.09**
	STL-10	Individual	66.6	58.5	56.3	52.9	7.90	8.52	8.30	8.60
		Slimmable	69.1	59.0	**50.8**	**50.6**	7.60	8.23	**8.83**	**8.81**
	CelebA	Individual	18.0	11.9	9.9	8.9	-	-	-	-
		Slimmable	**13.9**	**10.6**	**9.8**	**8.5**	-	-	-	-
cGAN-pd (cond)	CIFAR-10	Individual	55.1	33.5	16.5	15.5	6.46	7.90	8.22	8.52
		Slimmable (×)	21.7	17.2	16.1	16.2	7.87	8.31	8.49	8.34
		Slimmable (+)	**19.5**	**14.5**	**13.6**	**14.2**	**7.88**	**8.38**	**8.67**	**8.59**
	CIFAR-100	Individual	45.8	23.7	22.5	19.9	7.26	8.49	8.50	9.11
		Slimmable (×)	26.8	19.9	18.9	19.0	8.13	8.90	9.14	9.22
		Slimmable (+)	**23.8**	**18.9**	**18.6**	**17.9**	**8.26**	**9.08**	**9.17**	**9.29**

Table 1: FID and IS on both unconditional (uncond) and class-conditional (cond) generation. We do not calculate IS on CelebA as it is a face dataset that lacking inter-class diversity, which IS measures. For class-conditional generation, (+) means our proposed sliceable conditional batch normalization while (×) means the naive way that extends each sBN to cBN. Bold numbers indicate our slimmable method outperforms the individually trained models.

Datasets

We employ the following datasets for main experiments: **CIFAR-10/100** consists of 50k training images and 10k validation images with resolution of 32×32. CIFAR-10 has 10 classes while CIFAR-100 has 100 classes. **STL-10** is resized into the size of 48×48 as done in (Miyato et al. 2018). There are 100k and 8k unlabeled images in the training set and validation set, respectively. **CelebA** is a face dataset with 202,599 celebrity images with resolution of 178×218 originally. We follow the practice in (Hou, Shen, and Cheng 2020) to center crop them to 178×178 and then resize them to 64×64. We divide the last 19,962 images into the validation set and the remaining 182,637 images as the training set. We use the training set for training the models and the validation set for evaluation when calculating the statistics of the real data.

Evaluation Metrics

For evaluating the performance of all models on generation, we adopt two widely used evaluation metrics: Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance (FID) (Heusel et al. 2017). IS computes the KL divergence between the conditional class distribution and marginal class distribution. FID is the Fréchet distance (the Wasserstein-2 distance between two Gaussian distributions) between two sets of features obtained through the Inception v3 network trained on ImageNet. We randomly generate 50k images to calculate IS on all datasets, and 10k images to compute FID except STL-10, which we sample 8k images.

To measure the consistency between generators at different widths of SlimGAN, we present a metric, called Inception Consistency (IC), which measure the expected feature difference between two generators, G_{w_i} and G_{w_j} at width w_i and w_j, respectively:

$$\mathrm{IC}(G_{w_i}, G_{w_j}) = \mathbb{E}_{z \sim \mathcal{P}_Z}[\|\Phi(G_{w_i}(z)) - \Phi(G_{w_j}(z))\|_2^2],$$

where $\Phi(\cdot)$ outputs the feature of the last hidden layer of Inception v3 network trained on ImageNet.

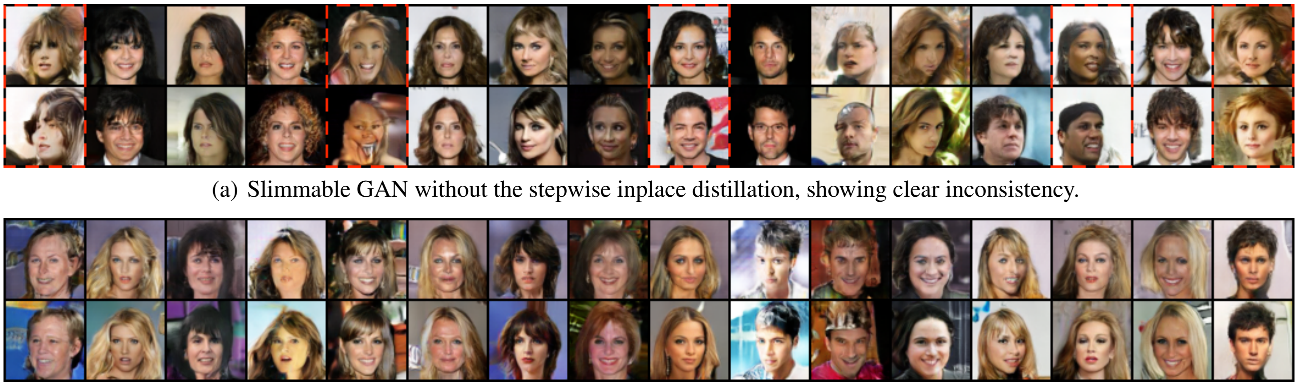
Given the width multiplier list $\mathcal{W}$, we average IC between all generator pairs as mean IC (mIC):

$$\mathrm{mIC}(G, \mathcal{W}) = \frac{1}{N \cdot (N-1)} \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} \mathrm{IC}(G_{w_i}, G_{w_j}).$$

We randomly sample 10k images to estimate the mIC score.

Experimental Settings

We implement all models based on Mimicry (Lee and Town 2020) using PyTorch framework. The optimizer is Adam with betas $(\beta_1, \beta_2) = (0.5, 0.999)$ for DCGAN and $(\beta_1, \beta_2) = (0.0, 0.9)$ for ResNet based SNGAN. The learning rate is $\alpha = 2 \times 10^{-4}$, except CelebA on DCGAN, which is $\alpha = 10^{-4}$. The iterations of updating the generator are $T = 100\text{k}$ for all methods. The discriminator update steps per generator update step are $K = 5$ for ResNet and



(a) Slimmable GAN without the stepwise inplace distillation, showing clear inconsistency.

(b) Slimmable GAN with the stepwise inplace distillation, showing improved consistency.

Figure 2: Qualitative consistency on CelebA.

$K = 1$ for DCGAN. As for the detailed network architecture, we exactly follow that in SNGAN (Miyato et al. 2018) and cGAN-pd (Miyato and Koyama 2018). The width multiplier list is set to $\mathcal{W} = [0.25, 0.5, 0.75, 1.0]\times$.

Experimental Results

Unconditional generation For unconditional generation, we experiment with three datasets, CIFAR-10, STL-10, and CelebA, on two backbones, DCGAN and ResNet. The hyper-parameter is set as $\lambda = 20$ for both backbones on CIFAR-10 and CelebA datasets, $\lambda = 10$ and $\lambda = 30$ for DCGAN and ResNet, respectively, on STL-10. We report the FID and IS results in Table 1. Individual represents individually trained GANs of each width. Our proposed SlimGAN surpasses in most cases or competes with the individually trained GANs in terms of both FID and IS scores, consistently demonstrating the effectiveness of SlimGAN across various datasets and network backbones. Surprisingly, SlimGAN outperforms the individual model at the widest width. We argue that the reasons are twofold. First, training narrow networks could provide extra informative signals for shared parameters with wide networks. Second, the parameter-shared discriminators have a certain regularization, which may improve the generalization of each discriminator. We believe this is a promising advanced training technique for GANs, and leave it for future work. Additionally, some generators at width $0.75\times$ reach or surpass the widest generators, which are trained with only adversarial objectives, reflecting the benefit of the combination of distillation and adversarial training.

Class-conditional generation For class-conditional generation experiments, we adopt cGAN-pd as the backbone on both CIFAR-10 and CIFAR-100, and report both FID and IS in the bottom of Table 1. The hyper-parameter is set as $\lambda = 10$ for CIFAR-10 and $\lambda = 20$ for CIFAR-100. The symbols in the parentheses after our slimmable methods represent different implementations of BN, i.e., $(\times)$ represents the naive BN, and $(+)$ represents our proposed scBN. Overall, the slimmable generators with different BNs outperform the baseline heavily. Particularly, our proposed scBN gains further improvement compared with the naive BN due to sharing the label information across different widths.

Methods	IS (↑)		FID (↓)	
	$0.5\times$	$1.0\times$	$0.5\times$	$1.0\times$
Individual	18.8	29.9	48.1	33.9
Slimmable	**32.7**	**36.1**	**32.8**	**30.8**

Table 2: BigGANs on ImageNet after 50k iterations.

BigGANs on ImageNet We train our slimmable method with BigGAN (Brock, Donahue, and Simonyan 2019) on ImageNet (128 × 128) for 50k iterations. The width multiplier list is set as $\mathcal{W} = [0.5, 1.0]\times$. The IS and FID are reported in Table 2. In a word, our slimmable method surpasses the individually trained BigGANs, showing a strong capability on large-scale dataset of high-resolution images.

SlimDCGAN	CIFAR-10	STL-10	CelebA
+ w/o distillation	282.7	277.4	110.2
+ w/ distillation	**231.3**	**243.2**	**96.1**
SlimResGAN	CIFAR-10	STL-10	CelebA
+ w/o distillation	285.7	342.4	116.9
+ w/ distillation	**241.4**	**248.7**	**97.9**

Table 3: mIC (↓) on CIFAR-10, STL-10, and CelebA.

Consistency We first report the quantitative consistency (mIC) in Table 3, which verifies that distillation can improve the consistency. We also show the qualitative consistency results on CelebA in Figure 2. For each method, the top row represents the narrowest generator and the bottom row indicates the widest generator. The same column in each method shows the images generated through the same latent code. Compared with the method without distillation, our distillation improves the consistency. For example, the method without distillation synthesis faces with disparate hairs.

DCGAN on CIFAR-10	FID (↓)					mIC (↓)
	0.25×	0.5×	0.75×	1.0×	AVG	
Individual	46.9	34.6	30.4	27.4	34.8	-
Individual (full D)	45.6	33.2	29.4	27.4	33.9	-
Slimmable G	40.0	35.2	34.4	33.4	35.8	264.3
+ shared D	40.9	30.2	27.0	25.2	30.8	282.7
+ shared D + distillation (SlimGAN)	37.3	**28.5**	**25.8**	**25.2**	**29.2**	231.3
+ same D	180.4	136.9	141.3	158.6	154.3	376.8
+ slimmable D	43.6	35.8	31.0	33.0	35.9	269.5
+ distillation (w/o GAN loss for narrows)	87.9	56.2	37.8	28.9	52.7	**204.8**
+ shared D + naive distillation	**36.6**	29.8	26.3	25.5	29.6	232.5

Table 4: Ablation Study on CIFAR-10. AVG means the averaged FID across all widths.

Ablation Study

In this section, we conduct an extensive ablation study on CIFAR-10 to verify the effectiveness of the design in SlimGAN, including network framework and objective function. The first two rows in Table 4 are both individually trained GANs. Individual (full D) means the widths of all discriminators in these individual GANs are fixed as the widest width, which is consistent with SlimGAN. Directly applying the slimmable operator to the generator with multiple independent discriminators (Slimmable G), unfortunately, obtains degradation, especially for wide generators. Although this issue is alleviated by sharing partial parameters of these discriminators (shared D), it compromises consistency. Fortunately, with stepwise inplace distillation, our final method (SlimGAN) not only achieves further improvements for narrow generators on generation but also obtains remarkable consistency. When utilizing the same discriminator (same D) for all generators, the awful FID reveals that the one-to-one relationship in the generator-discriminator pair should be obeyed. As an alternative parameter-sharing way, slimming the discriminator (slimmable D) does not gain satisfactory results. This is because those narrow discriminators would lack the capability to estimate the divergences, as they are contained by wide discriminators. Without adversarial training but only distillation for narrow generators, they tend to produce blurry images and get inferior FID. Compared with the stepwise distillation, only the narrowest network is improved when using the naive distillation (all narrow generators learn from the widest one).

Complexity Analysis

Saving parameters is the major advantage of the slimmable generator over the individually trained ones. We investigate the number of parameters of unconditional (uncond) and class-conditional generators in Table 5. Specifically, cond10 and cond100 represent the class-conditional generators (cGAN-pd) that trained with 10 (CIFAR-10) and 100 (CIFAR-100) labels, respectively. Individual (I-) methods require an independent generator on each width, while the slimmable (S-) approach only needs one. Therefore, the slimmable generator reduces parameters greatly compared with the sum of all individuals. As for class-conditional generative models, our proposed scBN (+) only adds negligible parameters on the widest individual generators compared to the naive BN approach. This advantage would become more obvious with the increase of labels or switches.

CIFAR	0.25×	0.5×	0.75×	1.0×	Total
I-uncond	0.35	1.15	2.39	4.08	7.97
I-cond-10	0.36	1.16	2.41	4.10	8.04
I-cond-100	0.42	1.29	2.61	4.37	8.70
S-uncond	-	-	-	-	**4.08**
S-cond-10 (+)	-	-	-	-	**4.11**
S-cond-100 (+)	-	-	-	-	**4.38**
S-cond-10 (×)	-	-	-	-	4.15
S-cond-100 (×)	-	-	-	-	4.81

Table 5: The number of parameters (M) in the generators.

Conclusions

In this paper, we introduce slimmable generative adversarial networks (SlimGAN), which can execute at different widths at runtime according to various energy budgets of different devices. To this end, we utilize multiple discriminators that share partial parameters to train the slimmable generator. In addition to the adversarial objectives, we introduce stepwise inplace distillation to explicitly guarantee the consistency between generators at different widths. In the case of class-conditional generation, we propose a sliceable conditional batch normalization to incorporate the label information under the width-switchable mechanism. Comprehensive experiments demonstrate that SlimGAN reaches or surpasses the individually trained GANs. In the future, we will explore more practical generation tasks, e.g., text-to-image generation and image-to-image translation.

Acknowledgments

This work is funded by the National Key R&D Program of China (2020AAA0105200) and the National Natural Science Foundation of China under grant numbers 91746301 and U1911401. Huawei Shen is also funded by K.C. Wong Education Foundation and Beijing Academy of Artificial Intelligence (BAAI).

References

Aguinaldo, A.; Chiang, P.-Y.; Gain, A.; Patil, A.; Pearson, K.; and Feizi, S. 2019. Compressing GANs using Knowledge Distillation. *arXiv preprint arXiv:1902.00159* .

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 214–223.

Brock, A.; Donahue, J.; and Simonyan, K. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

de Vries, H.; Strub, F.; Mary, J.; Larochelle, H.; Pietquin, O.; and Courville, A. 2017. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems 30*.

Fu, Y.; Chen, W.; Wang, H.; Li, H.; Lin, Y.; and Wang, Z. 2020. Autogan-distiller: Searching to compress generative adversarial networks. *arXiv preprint arXiv:2006.08198* .

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30*.

Hou, L.; Shen, H.; and Cheng, X. 2020. Dual Rejection Sampling for Wasserstein Auto-Encoders. In *24th European Conference on Artificial Intelligence*.

Hu, H.; Dey, D.; Hebert, M.; and Bagnell, J. A. 2019. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3812–3821.

Kupyn, O.; Budzan, V.; Mykhailych, M.; Mishkin, D.; and Matas, J. 2018. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lee, K. S.; and Town, C. 2020. Mimicry: Towards the Reproducibility of GAN Research. In *CVPR Workshop on AI for Content Creation*.

Li, M.; Lin, J.; Ding, Y.; Liu, Z.; Zhu, J.-Y.; and Han, S. 2020. Gan compression: Efficient architectures for interactive conditional gans. *arXiv preprint arXiv:2003.08936* .

Lim, J. H.; and Ye, J. C. 2017. Geometric gan. *arXiv preprint arXiv:1705.02894* .

Liu, L.; and Deng, J. 2018. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 3675–3682.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*.

Miyato, T.; and Koyama, M. 2018. cGANs with Projection Discriminator. In *International Conference on Learning Representations*.

Nowozin, S.; Cseke, B.; and Tomioka, R. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems 29*.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X.; and Chen, X. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29*.

Thanh-Tung, H.; Tran, T.; and Venkatesh, S. 2019. Improving Generalization and Stability of Generative Adversarial Networks. In *International Conference on Learning Representations*.

Tran, D.; Ranganath, R.; and Blei, D. M. 2017. Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896* .

Wang, H.; Gui, S.; Yang, H.; Liu, J.; and Wang, Z. 2020a. GAN Slimming: All-in-One GAN Compression by A Unified Optimization Framework. *arXiv preprint arXiv:2008.11062* .

Wang, Y.; Sun, F.; Li, D.; and Yao, A. 2020b. Resolution switchable networks for runtime efficient image recognition. In *European Conference on Computer Vision*, 533–549.

Yu, C.; and Pool, J. 2020. Self-Supervised GAN Compression. *arXiv preprint arXiv:2007.01491* .

Yu, J.; and Huang, T. 2019a. AutoSlim: Towards One-Shot Architecture Search for Channel Numbers. *arXiv preprint arXiv:1903.11728* .

Yu, J.; and Huang, T. S. 2019b. Universally Slimmable Networks and Improved Training Techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1803–1811.

Yu, J.; Yang, L.; Xu, N.; Yang, J.; and Huang, T. 2019. Slimmable Neural Networks. In *International Conference on Learning Representations*.

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2019. Self-Attention Generative Adversarial Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 7354–7363.

Zhang, M.; Zhang, Y.; Zhang, L.; Liu, C.; and Khurshid, S. 2018. Deeproad: Gan-based metamorphic autonomous driving system testing. *arXiv preprint arXiv:1802.02295* .