Towards Compressing Efficient Generative Adversarial Networks for Image Translation via Pruning and Distilling

Chapter · September 2021

DOI: 10.1007/978-3-030-86340-1_51

Luqi Gong
Chinese Academy of Sciences
2 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Chao li
Chinese Academy of Sciences
35 PUBLICATIONS **241** CITATIONS
SEE PROFILE

Hui Zhu
Institute of Computing Technology
13 PUBLICATIONS **51** CITATIONS
SEE PROFILE

Tangwen Qian
Chinese Academy of Sciences
30 PUBLICATIONS **488** CITATIONS
SEE PROFILE

Towards Compressing Efficient Generative Adversarial Networks for Image Translation via Pruning and Distilling

Luqi Gong[1], Chao Li[1(✉)], Hailong Hong[1], and Yongjun Xu[1]

Institute of Computing Technology, Chinese Academy of Science, Beijing, China
{gongluqi,lichao,honghailong, xyj}@ict.ac.cn

Abstract. Deploying GANs(Generative Adversarial Networks) for Image Translation tasks on edge devices is plagued with the constraints of storage and computation. Compared to some methods like neural architecture search(NAS), filter pruning is an effective DNN(Deep Neural Network) compressing method. It can compressing DNNs in a short time. The filter importance is measured by the filter norm, the filters with low norm are pruned. As for image classification, the filter with larger norm has larger influence on the final classification scores. However, as illustrated in Figure 4, the filter with large norm don't always have a big impact on the quality of generated images for GANs. Based on the observation that the filter close to the filters' center in the same convolution layer can be represented by others in [8], we develop a distance-based pruning criterion. We prune the filters which are close to the filters' center in a convolution layer. KD(Knowledge distillation) trains the compressed model and improves its performance. The most common KD method ignores the transformation information across the feature maps, which is important for GANs. We take them as additional knowledge and transfer it from the uncompressed GAN to the pruned GAN. Our experiments on CycleGan, Pix2pix, and GauGan achieved excellent performance. Without losing image quality, we obtain **51.68**× and **36.20**× compression on parameters and MACs[1] respectively on CycleGan. Our code[2] will be made available at github.

Keywords: GAN Compression · Pruning · Knowledge Distillation.

1 Introduction

In recent years, GANs(Generative Adversarial Networks)[6] are frequently prescribed for image generation, image translation, text generation and style transfer. With the development of GANs for image translation tasks, their parameters and MACs become very large. However, some applications require interaction with humans and demand low-latency on-device performance for better user experience. Edge devices (VR headsets, mobile phones, tablets, etc) are tightly

[1] Multiply-Accumulate Operations
[2] We will open source within one week after the paper being received

constrained by hardware resources. Deploying the GANs for image translation tasks on edge devices is limited by the device memory and inference speed. As an example, the frequently-used CycleGan[22] has 11.37M parameters and 56.83G MACs, making it difficult to deploy directly on edge devices.

With the observation that DNNs have a large parameter redundancy, model compression methods have been widely studied to reduce the number of parameters and MACs in neural networks. The most frequently used methods include human-designed, neural architecture search(NAS)[4], pruning[7–9, 13], and KD(knowledge distillation)[15, 17, 18], etc. The above methods mainly compress the model for image classification and object detection. However, the network architecture and principle of GANs are different from classical CNN models. As for GAN compression, the human-designed method can not get rid of a large number of attempts. NAS consumes large computational complexity due to the huge search space. However, the pruning method compresses model quickly with a small amount of manual intervention.

Filter norm is mostly taken as the criterion for CNN model pruning methods known as the norm-based criterion. For the classification task, neurons with larger activation contribute more to the final classification scores. This assumption is not suitable for GANs because the GAN's output is image instead of classification score. Figure 4(b) verifies that the filter norm can't represent the filter importance for GANs. Existing CNN model pruning methods fine-tune the model by conventional training to improve the model performance. This only improve part of the compressed model's performance, which is still far from the uncompressed model. With the difficulty in restoring the performance of compressed model to the uncompressed by direct training, KD is proved to be an effective fine-tuning method for GAN in [14, 19]. [14, 19] transfer the uncompressed GAN's knowledge to the compressed GAN. This way is also performance limited as they simply make the compressed GAN to mimic the feature maps of the uncompressed GAN. However, mimicking the generated features of the uncompressed GAN is only a hard constraint in these works. How the output images are generated layer by layer from the input is very important implicit information for GANs. These works ignore transferring this kind of transformation information.

To address the problems mentioned above, we introduce a general GAN compression framework consisting of distance-based filter pruning and KD guided by transformation information across feature maps in different layers. In our GAN pruning method, we calculate the filter center in a convolutional or deconvolutional layer. Then the filters whose distance to the filter center less than the threshold are removed. The rest of the filters can achieve the same feature extraction effect as all the filters because the removed filters are deemed to be represented by other filters. Our pruning method is used for compressing a small GAN architecture. After that, the KD is applied to fine-tuning the pruned GAN. We regard the transformation information across the feature maps as the knowledge and transfer it from the uncompressed GAN to the compressed GAN. This kind of knowledge can guide compressed GAN to learn how to generate feature maps and output images layer by layer.

In summary, the contributions of our paper are summarized as follows:

- **Pruning**: We propose a filter distance-based pruning method for compressing efficient GANs. It can correctly measure the importance of the convolutional and deconolutional filters in GANs.
- **Distillation**: In order to restore the quality of the images generated from the GANs after pruning, the feature maps and their transformation information are transferred from the original GAN to the pruned GAN. We run the Pruning-Distillation process iteratively.
- We evaluate our proposed method on three image translation models trained on four benchmark datasets. We got 51.68× and 36.20× compression on parameters and MACs respectively without performance dropping for CycleGan. We compressed the Pix2pix parameters by 22.31×, MACs by 12.46× at most. The GauGan got 16.49× parameter compression from 93.0M to 5.64M, meanwhile, the MACs were dropped from 281G to 25.63G with a slight performance decrease.

The rest of the paper is organized as follow. In sec 2, we introduce related methods of GAN compressing, including stacking human-designed modules, pruning, neural architecture search, and knowledge distillation. Then our proposed method are detailed in sec 3. We conduct comparative experiments and perform extensive analysis of experimental results in sec 4. Finally, we conclude our paper in sec 5.

2 Related Work

Generally, existing methods compress a meticulous model architecture with high performance by stacking human-designed CNN modules, pruning the over-parameterized network, or neural architecture search. After that, they promote the compressed network's performance by training or KD.

Stacking Human-designed Modules. ShuffleNet[21], SqueezeNet[11], and MobileNet[10] compress the model by using the efficient modules designed manually. They stack the well-designed modules to get an efficient CNN network easily in this way. But they need to design the whole architecture including the number of layer and filter.

Pruning. Pruning methods remove the redundant connections or convolution filters. As for connection pruning, it leads to sparse networks. This needs specific hardware or acceleration library for deployment. Filter pruning methods are widely used in compressing meticulous CNN model. The most common criterion to calculate the filter importance is the filter norm. [20] compresses GANs by pruning filters with low filter norm. This criterion is effective for the classical

CNN model because neurons with larger activation contribute more to the final classification scores. However, GANs have different kind of modules such as the deconvolutional layer. They are used for image generation whose outputs are images instead of classification scores.

Neural Architecture Search. NAS has been applied to design networks that are on par or outperform hand-designed architectures. Methods for NAS can be categorized according to the search space, search strategy, and performance estimation strategy used. [19] compresses GANs by developing a co-evolutionary approach. Generators for two image domains are encoded as two populations and synergistically optimized for investigating the most important convolution filters iteratively, obtaining portable architectures of satisfactory performance. However, this method is designed for specific GANs. GANs compressed by this method generate images with a poor performance. With the increase of compression ratio, the metric(FID) drops severely. [5, 14] introduce neural architecture search(NAS) to GAN compression, and transfer knowledge of multiple intermediate representations from the original model to its compressed model. The large search space leads to big computing resource consumption which is hard to use in industry.

KD(Knowledge Distillation). KD is used to improve the performance of compressed model. It extracts the feature maps or outputs and transfers them from the uncompressed model to the compressed model, aligning them two by loss function. This method is proven to be effective for GAN compression in [1, 2, 5, 20], but the insufficiency of knowledge makes the compressed GAN difficult to restore the performance to the uncompressed model. They simply transfer feature maps from the predesigned student GAN to the teacher GAN. Their success also depends on the appropriate design of student network architectures. [1, 2] require us to design the compressed GAN's architecture including the number of layer and filter manually. However, setting a GAN architecture with an accurate model capacity is difficult in a trial-and-error fashion.

3 Method

Our method consists of convolution kernel pruning and KD illustrated in Figure 1. For one step, we prune the GAN model according to a certain step compression ratio which means the pruning ratio for this step. Then KD is applied to the pruned GAN. We run the Pruning-KD step iteratively until the accumulated pruning ratio reaches the target pruning ratio or KD can't restore the performance of the pruned GAN.

3.1 Notations

Formally, we introduce symbols and notations in this subsection. We assume that a GAN network has L layers. Input and output channels are represented as N_i and N_{i+1} respectively for the i_{th} convolutional or deconvolutional layer. $\boldsymbol{\mathcal{W}_{i,j}} \in \mathbb{R}^{N_i \times K \times K}, i \in [1, L], j \in [1, N_{i+1}]$ represents the j_{th} filter of the i_{th} layer, where K is the kernel size. We regard $\boldsymbol{\mathcal{F}_i} \in \mathbb{R}^{N_{i+1} \times w \times h}$ as the output feature maps of the i_{th} layer, where w and h are the width and height, respectively. f and f_{step} are the target and step compression ratio, respectively. $\mathbb{G}$ and $\mathbb{G}'$ are the generators for uncompressed GAN and compressed GAN. Their outputs are $G(x)$ and $G'(x)$

3.2 Filter Distance-based Pruning Method

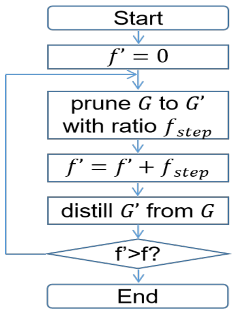


Fig. 1. The whole pipeline of our method. f means the target compression ratio

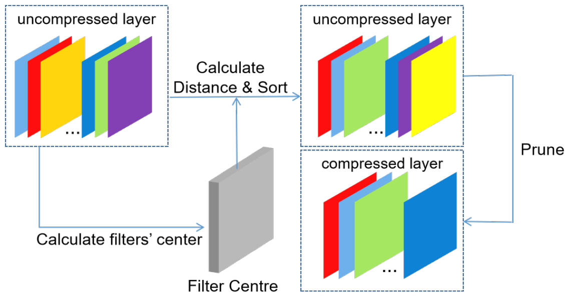


Fig. 2. The filter distance-based pruning method for compressing GANs

As showed in [14, 20], some filters are redundant due to their representation ability can be achieved by other filters. Thus these filters can be pruned. After pruning, the rest filters can play the same role and get the same performance in feature extraction as all filters remained.

As illustrated in Figure 2, We take one pruning step as an example. We calculate the filter center $\boldsymbol{\mathcal{W}_i^*}$ for the filters $\boldsymbol{\mathcal{W}_i} = [\boldsymbol{\mathcal{W}_{i,1}}, \boldsymbol{\mathcal{W}_{i,2}}, ..., \boldsymbol{\mathcal{W}_{i,N_{i+1}}}]$ in the i_{th} layer, i.e.,

$$\boldsymbol{\mathcal{W}_i^*} = \arg\min_{x \in \mathbb{R}^{N_i \times K \times K}} \sum_{j \in [1, N_{i+1}]} ||x - \boldsymbol{\mathcal{W}_{i,j}}||_2. \tag{1}$$

The distances between the filters $\boldsymbol{\mathcal{W}_i}$ and their center $\boldsymbol{\mathcal{W}_i^*}$ are used to measure the importance of filters. They are calculated as $\boldsymbol{d} = [d_1, d_2, ..., d_{N_{i+1}}]$ where

$$d_j = ||\boldsymbol{\mathcal{W}_{i,j}} - \boldsymbol{\mathcal{W}_i^*}||_2^2. \tag{2}$$

$Top(\boldsymbol{d}, N)$ is a function that can get the top N values in $\boldsymbol{d}$. It returns an ordered decreasing list whose length is N. This operation can evaluate and sort the filter importance. We note the last value in d as the threshold th by 3. The threshold's

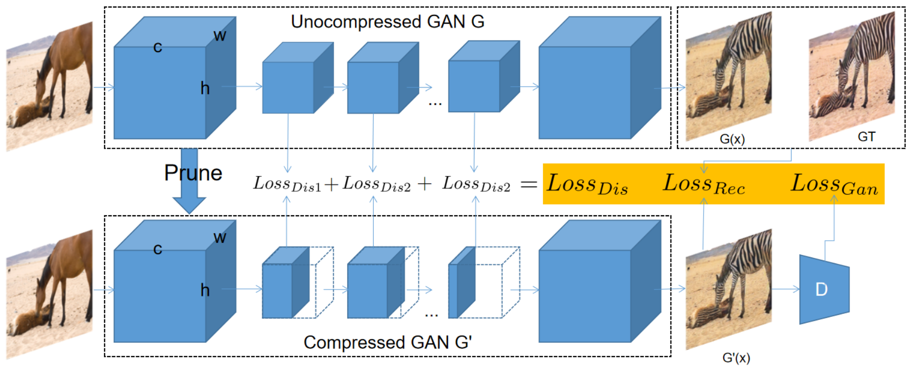


Fig. 3. Our KD loss consists of three parts: regular KD loss, paired learning KD loss, novel KD loss guided by the transformation information

index is $N-1$. For one step, we set $N = (1-f_{step}) \times N_{i+1}$. This achieves pruning the filters in the ratio of f_{step}.

$$th = Top(\boldsymbol{d}, N)[N-1]. \tag{3}$$

Finally, we remove the filters in the i_{th} layer whose distance to the center $\boldsymbol{\mathcal{W}_i^*}$ is less than the threshold th. The pruned generator for this pruning step are parameterized with the remaining filters $\boldsymbol{\mathcal{W}'_i} = [\boldsymbol{\mathcal{W}'_{i,1}}, \boldsymbol{\mathcal{W}'_{i,2}}, ..., \boldsymbol{\mathcal{W}'_{i,f_{step} \times N_{i+1}}}]$. We replace the original convolutional filters $\boldsymbol{\mathcal{W}_i}$ with the pruned convolutional filters $\boldsymbol{\mathcal{W}'_i} \in \mathbb{R}^{f_{step} \cdot N_i \times K \times K}$. The filters and their parameters are pruned to the ratio $(1 - f_{step})$ of the original.

We run the pruning step some times until the cumulative compression ratio reaches the target compression ratio f.

3.3 Fine-tune Compressed GAN via KD

As illustrated in Figure 3, we introduce transformation information into KD loss as the additional supervised information. The transformation information can be defined by the relationship between two intermediate feature maps. The intermediate feature maps are from two different layers in GAN. This kind of relationship can be represented as the inner product of these two vectors' directions. The vectors are flatten from the feature maps of two different layers.

For a GAN framework, assuming $\boldsymbol{\mathcal{F}}_i \in \mathbb{R}^{N_{i+1} \times w \times h}$ and $\boldsymbol{\mathcal{F}}_j \in \mathbb{R}^{N_{j+1} \times w \times h}$ are the feature maps for the i_{th} and j_{th} layer, respectively, where N_{i+1} and N_{j+1} are the number of output channels for the i_{th} and j_{th} layer, and $N_{i+1} = N_{j+1}$. $\boldsymbol{\mathcal{F}_{i,m,n}} \in \mathbb{R}^{N_{i+1}}$ and $\boldsymbol{\mathcal{F}_{j,m,n}} \in \mathbb{R}^{N_{j+1}}$ are the $(\cdot, m, n)$ entries of $\boldsymbol{\mathcal{F}_i}$ and $\boldsymbol{\mathcal{F}_j}$. Then, the transformation information matrix $\boldsymbol{M} \in \mathbb{R}^{N_{i+1} \times N_{j+1}}$ is calculated by

$$\boldsymbol{M} = \sum_{m=1}^{w} \sum_{n=1}^{h} \frac{\boldsymbol{\mathcal{F}_{i,m,n}} \times \boldsymbol{\mathcal{F}^T_{j,m,n}}}{w \times h}. \tag{4}$$

For a GAN compression task, we can assume that there are N transformation information matrices denoted as $\boldsymbol{M}_i^T, i \in [1, N]$, which are generated by the uncompressed GAN, and N transformation information matrices denoted as $\boldsymbol{M}_i^S, i \in [1, N]$, which are generated by the compressed GAN. For each pair of matrices between the teacher and student GANs $(\boldsymbol{M}_i^T, \boldsymbol{M}_i^S), i \in [1, N]$ with the same spatial size, we align them by the $l2$ norm where

$$L_T = \sum_{i=1}^{N} ||\boldsymbol{M}_i^T - \boldsymbol{M}_i^S||_2^2. \tag{5}$$

We also consider the loss of KD proposed in [14] as L_F. In the same way, we transfer the information of the feature maps $\boldsymbol{\mathcal{F}}_j^T, j \in [1, M]$ in the uncompressed GAN to the featuure maps $\boldsymbol{\mathcal{F}}_j^S, j \in [1, M]$ in the compressed GAN by

$$L_F = \sum_{j=1}^{M} ||\boldsymbol{\mathcal{F}}_j^S - \boldsymbol{\mathcal{F}}_j^T||_2^2. \tag{6}$$

Paired image translation task consists of examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists. Unpaired doesn't have this kind of correspondence. For the unpaired image translation task, we can view the uncompressed generator's output $G(x)$ as ground-truth and train our compressed generator $\mathbb{G}'$ with an objective L_{rec}. For the paired setting, we train our compressed generator $\mathbb{G}'$ with ground-truth $\boldsymbol{y}$. This objective is formalized as:

$$L_{rec} = \begin{cases} \mathbb{E}_{x,y}||G(x) - \boldsymbol{y}||_2^2 & \text{if paried GANs,} \\ \mathbb{E}_x||G(x) - G'(x)||_2^2 & \text{if unpaired GANs.} \end{cases} \tag{7}$$

The final loss is a multi-objective loss as showed in Eq. 8 where $\alpha_1, \alpha_2, \alpha_3$ are the coefficients, L_{GAN} is the original loss for adversarial training.

$$L = L_T + \alpha_1 L_F + \alpha_2 L_{rec} + \alpha_3 L_{GAN}. \tag{8}$$

4 Experiments

4.1 Experimental Settings

Models. CycleGan[22] is an unpaired Image-to-Image translation model. It transforms the image from a source domain to a target domain. Pix2Pix[12] is used for supervising Image-to-Image translation. U-Net is the backbone of its generator which can better retain the pixel-level detail at different resolutions. GauGan[16] proposed a spatially-adaptive normalization method which can better protect semantic details.

Datasets. Cityscapes has 5000 images of driving scenes in 50 cities. Horse $\longleftrightarrow$ Zebra collects 1187 horse images and 1474 zebra images from ImageNet. Edges $\longrightarrow$ Shoes consists of 50025 images from UTZappos. Map $\longleftrightarrow$ Aerial has 2194 images downloaded from the Google map.

Experimental Evaluation Metrics Frechet Inception Distance(FID)[3] uses the 2048-dimensional activations from the Inception intermediate layer. Then it models the activations from the real and generated images using the multivariate Gaussian distribution with mean μ and covariance σ. These statistics are then used for calculating the FID. The Lower FID is better.

Implementation Details. We first train a generator from scratch, then we prune it with the step compression ratio 5% and fine-tune it by our KD method. We carry out the pruning-distillation step above iteratively until the performance of the compressed GAN can't restore to the uncompressed GAN or the total compression ratio reaches the pre-set target compression ratio. For the Pix2pix and CycleGan, we use 0.0002 as the learning rate through the training procedure. The batch size is 1 for Cityscapes, Map $\longleftrightarrow$ Aerial, and Horse $\longleftrightarrow$ Zebra as well as 4 for Edges $\longrightarrow$ Shoes, 16 for GauGan. We adopt the Adam optimizer, keeping the learning rate constant before it linearly decays from the initial learning rate to 0. We set constant epoch as 100 while decay epoch is 100, 200, 300, or 400 depending on different datasets. Epoch set for compression is the same as from-scratch training. We use the generator with the best evaluation performance during training. We adjust $\alpha_1, \alpha_2, \alpha_3$ to ensure the three loss items are in the same order of magnitude.

4.2 Detailed Compression Results

Table 1. Experiment Results On Pix2pix,GauGan, CycleGan

Model	Dataset	Method	Parameters	MACs	mAP/FID
Pix2pix	cityscaps	Original	11.38M	56.80G	35.62
		Li *et al.*[14]	0.71M(16.02×)	5.66G(10.04×)	29.27
		Ours	**0.58M(19.62×)**	**3.69G(15.4×)**	**35.03**
	edges→shoes	Original	11.38M	56.8G	24.18
		Li *et al.*[14]	0.70M(16.25×)	4.81G(11.81×)	26.60
		Ours	**0.51M(22.31×)**	**4.56G(12.46×)**	**25.96**
	map→arial photo	Original	11.38M	56.8G	47.76
		Li *et al.*[14]	0.75M(15.17×)	4.68G(12.14×)	48.02
		Ours	**0.51M(22.31×)**	**4.56G(12.46×)**	**47.32**
GauGan	cityscaps	Original	93.00M	281.00G	58.89
		Li *et al.*[14]	20.40M(4.56×)	31.72G(8.86×)	56.75
		Ours	**5.64M(16.49×)**	**25.63G(10.96×)**	**54.40**
CycleGan	horse→zebra	Original	11.37M	56.83G	61.53
		Shu *et al.*[19]	-	13.40G(4.24×)	96.15
		Fu *et al.*[5]	0.98M(11.60×)	6.39G(8.89×)	83.60
		Li *et al.*[14]	0.34M(33.44×)	2.67G(21.28×)	64.95
		Ours	**0.22M(51.68×)**	**1.57G(36.20×)**	**60.49**

As shown in the Table 1, our method obtained better model performance and compression ratio.

For CycleGan compressed on horse $\longrightarrow$ zebra dataset, we achieved 51.68$\times$ compression on parameters and 36.20$\times$ compression on MACs. It is worth mentioning that, different from other methods[14, 19], our method compressed the CycleGan without FID decreases.

For Pix2pix, we conducted experiments on three datasets. The mAP in Cityscapes drops only 0.1 with a compression ratio of 19.62$\times$. For Map $\longleftrightarrow$ Aerial and Edges $\longrightarrow$ Shoes, we compress their model size by 22.31$\times$, MACs by 12.46$\times$.

GauGan is hard to be compressed in [14] which compressed it 4.56$\times$. We compressed it 16.49$\times$ from 93.00M to 5.64M on parameters, 11$\times$ from 281.00G to 25.63G on MACs with small FID decrease.

4.3 Ablation Study

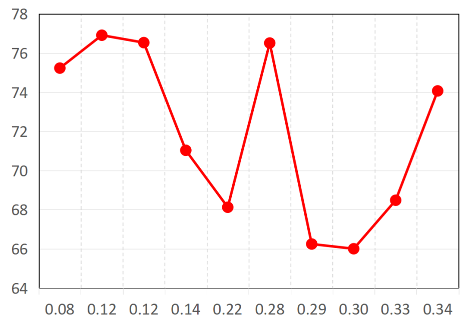
Table 2. Ablation Study For KD: Train, KD, and Ours mean fine-tuning the compressed GAN by normal training, the method in [14], and our KD method, respectively.

Model	Datast	FID		
		Train	KD	Ours
CycleGan	horse→zebra	67.721	63.5	**60.488**
Pix2pix	edges→shoes	27.37	27.46	**25.96**

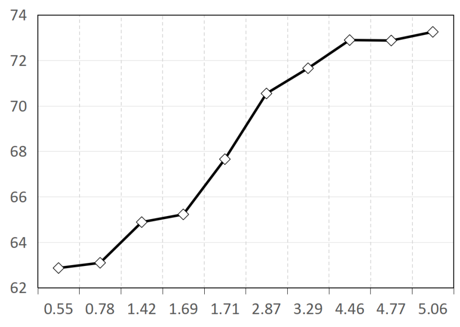
The effectiveness of our pruning method. As illustrated in Figure 4(b), we calculate the distances between some filters in GAN's certain layer and their filter center, we remove each of them. Then the normal training is applied to them as a fine-tuning process. Experiments show if we remove the filter with a large distance to the filter center, the GAN's performance is difficult to restore to the original GAN. This is because the filters far away from the filter center can't be represented by other filters. Such filters should not be removed. The Figure 4(a) shows that the FID after pruning and fine-tuning is not clearly affected by the L1 norm of the filters, which indicates the norm-based pruning method is not suitable for GANs.

The advantage of our KD method. Table 2 shows that if we fine-tune the generator after pruning with the normal training method, the generator can't recovery to the original uncompressed performance. When we apply the KD method in [14], it can get a better generator while our KD method achieves the best results.

Influence of step compression ratio in our experiment setting. We set the setp compression ratio to 3%, 5%, 7%, and 10% showed in Table 3. The FID



(a) L1 norm Pruning Criterion



(b) Our pruning Criterion

Fig. 4. The effectiveness of our pruning method

Table 3. Influence of step compression ratio

step compression ratio	3%	5%	7%	10%
FID	60.20	60.49	62.87	60.45

fluctuation along with the different step compression ratio is less than 3, which means the performance of pruning is not sensitive to this parameter.

5 Conclusion

In our work, we propose a general GAN compression framework. We apply a filter distance-based pruning method to design a small GAN architecture and use the KD method guided by transformation information across the feature maps to improve its image generation ability. Experimental results on different datasets and models showed that our method compresses GANs to a smaller size than other methods with minimal model performance dropping.

Bibliography

[1] Aguinaldo, A., Chiang, P.Y., Gain, A., Patil, A., Pearson, K., Feizi, S.: Compressing gans using knowledge distillation. arXiv preprint arXiv:1902.00159 (2019)

[2] Chen, H., Wang, Y., Shu, H., Wen, C., Xu, C., Shi, B., Xu, C.: Distilling portable generative adversarial networks for image translation. Proceedings of the AAAI Conference on Artificial Intelligence **34**, 3585–3592 (04 2020). https://doi.org/10.1609/aaai.v34i04.5765

[3] Dowson, D., Landau, B.: The fréchet distance between multivariate normal distributions. Journal of multivariate analysis **12**(3), 450–455 (1982)

[4] Enzo, Leiva-Aravena, Eduardo, Leiva, Vasty, Zamorano, Claudia, Rojas, John, and, M.: Neural architecture search with reinforcement learning. ence of the Total Environment (2019)

[5] Fu, Y., Chen, W., Wang, H., Li, H., Lin, Y., Wang, Z.: Autogan-distiller: Searching to compress generative adversarial networks. In: International Conference on Machine Learning. pp. 3292–3303. PMLR (2020)

[6] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)

[7] Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in neural information processing systems. pp. 1135–1143 (2015)

[8] He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4340–4349 (2019)

[9] He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1389–1397 (2017)

[10] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017)

[11] Iandola, F., Han, S., Moskewicz, M., Ashraf, K., Dally, W., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size (02 2016)

[12] Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5967–5976 (2017). https://doi.org/10.1109/CVPR.2017.632

[13] Lee, N., Ajanthan, T., Torr, P.: Snip: single-shot network pruning based on connection sensitivity. International Conference on Learning Representations (2019)

[14] Li, M., Lin, J., Ding, Y., Liu, Z., Zhu, J.Y., Han, S.: 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)

[15] Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5191–5198 (2020)

[16] Park, T., Liu, M., Wang, T., Zhu, J.: Semantic image synthesis with spatially-adaptive normalization. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2332–2341 (2019). https://doi.org/10.1109/CVPR.2019.00244

[17] Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Bengio, Y.: Fitnets: Hints for thin deep nets. In: ICLR (2015)

[18] Sau, B., Balasubramanian, V.: Deep model compression: Distilling knowledge from noisy teachers (10 2016)

[19] Shu, H., Wang, Y., Jia, X., Han, K., Chen, H., Xu, C., Tian, Q., Xu, C.: Co-evolutionary compression for unpaired image translation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3235–3244 (2019)

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

[21] Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018)

[22] Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2242–2251 (2017). https://doi.org/10.1109/ICCV.2017.244