GAN Slimming: All-in-One GAN Compression by A Unified Optimization Framework

Haotao Wang[1], Shupeng Gui[2], Haichuan Yang[2]
Ji Liu[3], and Zhangyang Wang[1]

[1] University of Texas at Austin, Austin TX 78712, USA
{htwang, atlaswang}@utexas.edu
[2] University of Rochester, Rochester NY 14627, USA
{sgui2, hyang36}@ur.rochester.edu
[3] AI Platform, Ytech Seattle AI Lab, FeDA Lab, Kwai Inc., Seattle WA 98004, USA
ji.liu.uwisc@gmail.com

Abstract. Generative adversarial networks (GANs) have gained increasing popularity in various computer vision applications, and recently start to be deployed to resource-constrained mobile devices. Similar to other deep models, state-of-the-art GANs suffer from high parameter complexities. That has recently motivated the exploration of compressing GANs (usually generators). Compared to the vast literature and prevailing success in compressing deep classifiers, the study of GAN compression remains in its infancy, so far leveraging individual compression techniques instead of more sophisticated combinations. We observe that due to the notorious instability of training GANs, heuristically stacking different compression techniques will result in unsatisfactory results. To this end, we propose the first unified optimization framework combining multiple compression means for GAN compression, dubbed **GAN Slimming** (GS). GS seamlessly integrates three mainstream compression techniques: model distillation, channel pruning and quantization, together with the GAN minimax objective, into one unified optimization form, that can be efficiently optimized from end to end. Without bells and whistles, GS largely outperforms existing options in compressing image-to-image translation GANs. Specifically, we apply GS to compress CartoonGAN, a state-of-the-art style transfer network, by up to **47**× times, with minimal visual quality degradation. Codes and pre-trained models can be found at https://github.com/TAMU-VITA/GAN-Slimming.

1 Introduction

Generative adversarial networks (GANs) [12], especially, image-to-image translation GANs, have been successfully applied to image synthesis [30], style transfer [7], image editing and enhancement [35,34,26], to name just a few. Due to the growing usage, there has been an increasing demand to deploy them on resource-constrained devices [38]. For example, many filter-based image editing

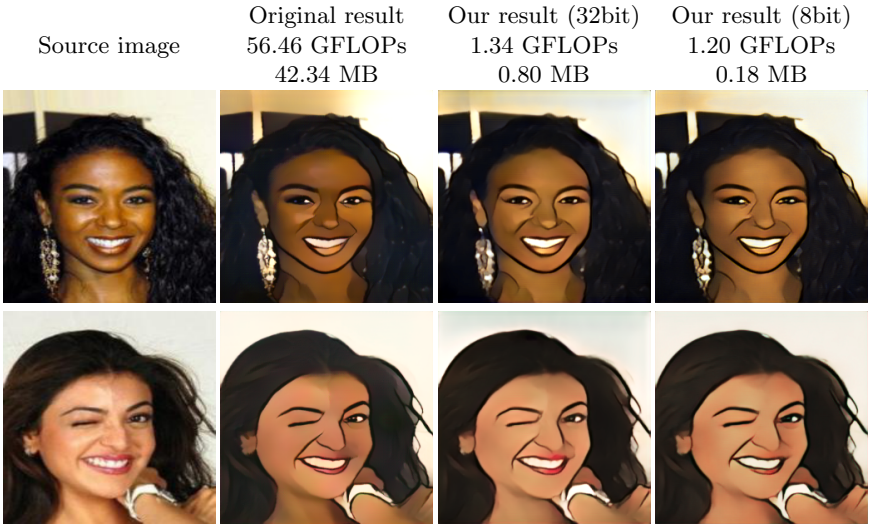


Fig. 1: Representative visual examples by GAN Slimming on CartoonGAN [7].

applications now desire to run image-to-image translation GANs locally. However, GANs, just like most other deep learning models, bear explosive parameter amounts and computational complexities. For example, in order to process a 256×256 image, a state-of-the-art style transfer network, CartoonGAN [7], would cost over 56 GFLOPs. Launching such models on mobile devices requires considerable memory and computation costs, which would be infeasible for most devices, or at least degrades user experience due to the significant latency.

Existing deep model compression methods mainly focus on image classification or segmentation tasks, and were not directly applicable on GAN compression tasks due to notorious instability of GAN minimax training. For example, [52] shows that generators compressed by state-of-the-art classifier compression methods [23,39,42] all suffer great performance decay compared with the original generator. Combining (either heuristically cascading or jointly training) multiple different compression techniques, such as channel pruning, model distillation, quantization and weight sharing, has been shown to outperform separately using single compression techniques alone in traditional classification tasks [57,43,47]. In comparison, current methods [52,3] have so far only tried to apply one single technique to compressing GANs. [52] proposed the first dedicated GAN compression algorithm: an evolutionary method based channel pruning algorithm. However, the method is specifically designed for CycleGAN and non-straightforward to extend to GANs without cycle consistency structure (*e.g.*, encoder-decoder GANs [7,50] that are also popular). A latest work [3] proposed to training an efficient generator by model distillation. By removing the dependency on cycle consistency structure, [3] achieves more general-purpose GAN compression

than [52]. However, the student network in [3] is still hand-crafted and relies on significant architecture engineering for good performance.

As discussed in [52,3], applying a single compression technique to GANs is already challenging due to their notorious training instability. As one may imagine, integrating multiple compression techniques together for GAN compression will only further amplify the instability, putting an open question:

Can we gain more from combining multiple compression means for GANs? If yes, how to overcome the arising challenge of GAN instability?

Our answer is by presenting the first end-to-end optimization framework combining multiple compression means for general GAN compression, named **GAN Slimming (GS)**. The core contribution of GS is a unified optimization form, that seamlessly integrates three popular model compression techniques (channel pruning, quantization and model distillation), to be jointly optimized in a minimax optimization framework. GS pioneers to advance GAN compression into jointly leveraging multiple model compression methods, and demonstrate the feasibility and promise of doing so, despite the GAN instability.

Experiments demonstrate that GS overwhelms state-of-the-art GAN compression options that rely on single compression means. For example, we compress the heavily-parameterized CartoonGAN by up to 47×, achieving nearly real-time cartoon style transfer on mobile devices, with minimal visual quality loss. Moreover, we have included a detailed ablation study for a deeper understanding of GS. Specifically, we demonstrate that *naively stacking different compression techniques cannot achieve satisfactory GAN compression, sometimes even hurting catastrophically*, therefore testifying the necessity of our unified optimization. We also verify the effectiveness of incorporating the minimax objective into this specific problem.

2 Related Works

2.1 Deep Model Compression

Many model compression methods, including knowledge distillation [47], pruning [18,20] and quantization [18], have been investigated to compress large deep learning models, primarily classifiers [2,4,6,17,51,53,58,60,62]. Structured pruning [61], such as channel pruning [24,39,45,61,63], result in hardware-friendly compressed models and thus are widely adopted in real-world applications. The authors of [61] enforced structured sparsity constraint on each layer's kernel weights, aided by group Lasso [31] to solve the optimization. [39] added ℓ_1 constraint on the learnable scale parameters of batch normalization layers in order to encourage channel sparsity, and used subgradient descent to optimize the ℓ_1 loss. Similarly, [24] also utilized sparsity constraint on channel-wise scale parameters, solved with an accelerated proximal gradient method [46].

Quantization, as another popular compression means, reduces the bit width of the element-level numerical representations of weights and activations. Earlier

works [18,37,66] presented to quantize all layer-wise weights and activations to the same low bit width, *e.g.*, from 32 bits to 8 bits or less. The model could even consist of only binary weights in the extreme case [8,48]. Note that, introducing quantization into network weights or activations will result in notable difficulty for propagating gradients [25]. Straight-through estimator (STE) [8] is a successful tool to solve this problem by a proxy gradient for back propagation.

Knowledge distillation was first developed in [22] to transfer the knowledge in an ensemble of models to a single model, using a soft target distribution produced by the former models. It was later on widely used to obtain a smaller network (student model), by fitting the "soft labels" (probabilistic outputs) generated from a trained larger network (teacher model). [1] used distillation to train a more efficient and accurate predictor. [41] unified distillation and privileged information into one generalized distillation framework to learn better representations. [59,5] used generative adversarial training for model distillation.

Combination of multiple compression techniques For compressing a deep classifier, [55,64] proposed to jointly train (unstructured) pruning and quantization together. [57,54] adopted knowledge distillation to fine-tune a pruned student network, by utilizing the original dense network as teacher, which essentially followed a two-step cascade pipeline. Similarly, [43,47] used full-precision networks as teachers to distill low-precision student networks. [14] showed jointly training pruning and quantization can obtain compact classifiers with state-of-the-art trade-off between model efficiency and adversarial robustness.

Up to our best knowledge, all above methods cascade or unify two compression techniques, besides that they investigate compressing deep classifiers only. In comparison, our proposed framework jointly optimize three methods in one unified form[4], that is innovative even for general model compression. It is further adapted for the special GAN scenario, by incorporating the minimax loss.

2.2 GAN Compression

GANs have been successful on many image generation and translation tasks [12,13,16,29,44], yet their training remains notoriously unstable. Numerous techniques were developed to stabilize the GAN training, *e.g.*, spectral normalization [44], gradient penalty [15] and progressive training [29]. As discussed in [52,3], the training difficulty causes extra challenges for compressing GANs, and failed many traditional pruning methods for classifiers such as [42,23,39].

The authors of [52] proposed the first dedicated GAN compression method: a co-evolution algorithm based channel pruning method for CycleGAN. Albeit successfully demonstrated on the style transfer application, their method faces several limitations. First, their co-evolution algorithm relies on the cycle consistency loss to simultaneously compress generators of both directions. It is hence non-straightforward to extend to image-to-image GANs without cycle consistent

[4] A concurrent work [65] jointly optimized pruning, decomposition, and quantization, into one unified framework for reducing the memory storage/access.

loss (*e.g.*, encoder-decoder GANs [7,50]). Second, in order to avoid the instability in GAN training, the authors model GAN compression as a "dense prediction" process by fixing the original discriminator instead of jointly updating it with the generator in a minimax optimization framework. This surrogate leads to degraded performance of the compressed generator, since the fixed discriminator may not suit the changed (compressed) generator capacity. These limitations hurdle both its broader application scope and performance.

The latest concurrent work [3] explored model distillation: to guide the student to effectively inherit knowledge from the teacher, the authors proposed to jointly distill generator and discriminator in a minimax two-player game. [3] improved over [52] by removing the above two mentioned hurdles. However, as we observe from experiments (and also confirmed with their authors), the success of [3] hinges notably on the appropriate design of student network architectures. Our method could be considered as another important step over [3], that "learns" the student architecture jointly with the distillation, via pruning and quantization, as to be explained by the end of Section 3.1.

3 The GAN Slimming Framework

Considering a dense full-precision generator G_0 which converts the images from one domain $\mathcal{X}$ to another $\mathcal{Y}$, our aim is to obtain a more efficient generator G from G_0, such that their generated images $\{G_0(x), x \in \mathcal{X}\}$ and $\{G(x), x \in \mathcal{X}\}$ have similar style transfer qualities. In this section, we first outline the unified optimization form of our GS framework combining model distillation, channel pruning and quantization (Section 3.1). We then show how to solve each part of the optimization problem respectively (Section 3.2), and eventually present the overall algorithm (Section 3.3).

3.1 The Unified Optimization Form

We start formulating our GS objective from the traditional minimax optimization problem in GAN:

$$\min_G \max_D L_{GAN}, \text{ where } L_{GAN} = \mathbb{E}_{y\in\mathcal{Y}}[\log(D(y))] + \mathbb{E}_{x\in\mathcal{X}}[\log(1 - D(G(x))], \quad (1)$$

where D is the discriminator jointly trained with efficient generator G by minimax optimization. Since G is the functional part to be deployed on mobile devices and D can be discarded after training, we do not need to compress D. Inspired by the success of model distillation in previous works [22,3], we add a model distillation loss term L_{dist} to enforce the small generator G to mimic the behaviour of original large generator G_0, where $\mathrm{d}(\cdot, \cdot)$ is some distance metric:

$$L_{dist} = \mathbb{E}_{x\in\mathcal{X}}[\mathrm{d}(G(x), G_0(x))], \quad (2)$$

The remaining key question is: how to properly define the architecture of G? Previous methods [22,3] first hand-crafted the smaller student model's architecture and then performed distillation. However, it is well known that the choice

of the student network structure will affect the final performance notably too, in addition to the teacher model's strength.

Unlike existing distillation methods [3], we propose to *jointly infer* the G architecture together with the distillation process. Specifically, we assume that G can be "slimmed" from G_0, through two popular compression operations: channel pruning and quantization. For channel pruning, we follow [39] to apply L_1 norm on the trainable scale parameters γ in the normalization layers to encourage channel sparsity: $L_{cp} = \|\gamma\|_1$. Denoting all other trainable weights in G as W, we could incorporate the channel pruning via such sparsity constraint into the distillation loss in Eq. (2) as below:

$$L_{dist}(W, \gamma) + \rho L_{cp}(\gamma) = \mathbb{E}_{x \in \mathcal{X}}[\mathrm{d}(G(x; W, \gamma), G_0(x))] + \rho \|\gamma\|_1, \tag{3}$$

where ρ is the trade-off parameter controlling the network sparsity level. Further, to integrate quantization, we propose to quantize both activations and weights,[5] using two quantizers $q_a(\cdot)$ and $q_w(\cdot)$, respectively, to enable the potential flexibility for hybrid quantization [56]. While it is completely feasible to adopt learnable quantization intervals [28], we adopt uniform quantizers with pre-defined bit-width for $q_a(\cdot)$ and $q_w(\cdot)$, respectively, for the sake of simplicity (including hardware implementation ease). The quantized weights can be expressed as $q_w(W)$, while we use G_q to denote generators equipped with activation quantization $q_a(\cdot)$ for notation compactness. Eventually, the final objective combining model distillation, channel pruning and quantization has the following form:

$$\begin{aligned} L(W, \gamma, \theta) &= L_{GAN}(W, \gamma, \theta) + \beta L_{dist}(W, \gamma) + \rho L_{cp}(\gamma) \\ &= \mathbb{E}_{y \in \mathcal{Y}}\ [\log(D(y; \theta))] + \mathbb{E}_{x \in \mathcal{X}}\ [\log(1 - D(G_q(x; q_w(W), \gamma); \theta))] \\ &+ \mathbb{E}_{x \in \mathcal{X}}\ [\beta \mathrm{d}(G_q(x; q_w(W), \gamma), G_0(x))] \\ &+ \rho \|\gamma\|_1, \end{aligned} \tag{4}$$

where θ represents the parameters in D. The blue parts represent the distillation component, green represents channel pruning red represents quantization. The above Eq. (4) is the target objective of GS, which is to be solved in a minimax optimization framework:

$$\min_{W, \gamma} \max_{\theta}\ L(W, \gamma, \theta) \tag{5}$$

Connection to AutoML Compression. Our framework could be alternatively interpreted as performing a special neural architecture search (NAS) [19,11] to obtain the student model, where the student's architecture needs be "morphable" from the teacher's through only pruning and quantization operations.

Interestingly, two concurrent works [9,36] have successfully applied NAS to search efficient generator architectures, and both achieved very promising performance too. We notice that a notable portion of the performance gains shall be attributed to the carefully designed search spaces, as well as computationally

[5] We only quantize W, while always leaving γ unquantized.

intensive search algorithms. In comparison, our framework is based on an end-to-end optimization formulation, that (1) has explainable and well-understood behaviors; (2) is lighter and more stable to solve; and (3) is also free of the NAS algorithm's typical engineering overhead (such as defining the search space and tuning search algorithms). Since our method directly shrinks the original dense model via pruning and quantization only, it cannot introduce any new operator not existing in the original model. That inspires us to combine the two streams of compression ideas (optimization-based versus NAS-based), as future work.

3.2 End-to-End Optimization

The difficulties of optimizing (5) can be summarized in three-folds. First, the minimax optimization problem itself is unstable. Second, updating W involves non-differentiable quantization operations. Third, updating γ involves a sparse loss term that is also non-differentiable. Below we discuss how to optimize them.

Updating W The sub-problem for updating W in Eq. (5) is:

$$\begin{aligned} &\min_{W} \ L_W(W), \\ &\text{where } L_W(W) = \mathbb{E}_{x\in\mathcal{X}}[\log(1 - D(G_q(x; q_w(W), \gamma); \theta)) \\ &\qquad\qquad + \beta \mathrm{d}(G_q(x; q_w(W), \gamma), G_0(x))], \end{aligned} \tag{6}$$

To solve (6) with gradient-based methods, we need to calculate $\nabla_W L_W$, which is difficult due to the non-differentiable $q_a(\cdot)$ and $q_w(\cdot)$. We now define the concrete form of $q_a(\cdot)$, $q_w(\cdot)$ and then demonstrate how to back propagate through them in order to calculate $\nabla_W L_W$. Since both $q_a(\cdot)$ and $q_w(\cdot)$ are elementwise operations, we only discuss how they work on scalars. We use a and w to denote a scalar element in the activation and convolution kernel tensors respectively.

When quantizing activations, we first clamp activations into range $[0, p]$ to bound the values, and then use $s_a = p/2^m$ as a scale factor to convert the floating point number to m bits integers: $\mathrm{round}(\min(\max(0, a), p)/s_a)$. Thus the activation quantization operator is as follows:

$$q_a(a) = \mathrm{round}(\min(\max(0, a), p)/s_a) \cdot s_a. \tag{7}$$

For weights quantization, we keep the range of the original weights and use symmetric coding for positive and negative ranges to quantize weights to n bits. Specifically, the scale factor $s_w = \|w\|_\infty / 2^{(n-1)}$, leading to the quantization operator for weights:

$$q_w(w) = \mathrm{round}(w/s_w) \cdot s_w. \tag{8}$$

Since both quantization operators are non-differentiable, we use a proxy as the "pseudo" gradient in the backward pass, known as the straight through estimator (STE). For the activation quantization, we use

$$\frac{\partial q_a(a)}{\partial a} = \begin{cases} 1 & \text{if } 0 \le a \le p; \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Similarly for the weight quantization, the pseudo gradient is set to

$$\frac{\partial q_w(w)}{\partial w} = 1. \tag{10}$$

Now that we have defined the derivatives of $q_a(\cdot)$ and $q_w(\cdot)$, we can calculate $\nabla_W L_W$ through back propagation and update W using the Adam optimizer [32].

Updating $\boldsymbol{\gamma}$ The sub-problem for updating γ in Eq. (4) is a sparse optimization problem with a non-conventional fidelity term:

$$\begin{aligned} &\min_{\gamma} L_\gamma(\gamma) + \rho\|\gamma\|_1, \\ &\text{where } L_\gamma(\gamma) = \mathbb{E}_{x\in\mathcal{X}}[\log(1 - D(G_q(x; q_w(W), \gamma); \theta)) \\ &\qquad\qquad + \beta \mathrm{d}(G_q(x; q_w(W), \gamma), G_0(x))], \end{aligned} \tag{11}$$

We use the proximal gradient to update γ as follows:

$$g_\gamma^{(t)} \leftarrow \nabla_\gamma L_\gamma(\gamma)\Big|_{\gamma=\gamma^{(t)}} \tag{12}$$

$$\gamma^{(t+1)} \leftarrow \mathrm{prox}_{\rho\eta^{(t)}}(\gamma^{(t)} - \eta^{(t)} g_\gamma^{(t)}) \tag{13}$$

where $\gamma^{(t)}$ and $\eta^{(t)}$ are the values of γ and learning rate at step t, respectively. The proximal function $\mathrm{prox}_\lambda(\cdot)$ for the ℓ_1 constraint is the soft threshold function:

$$\mathrm{prox}_\lambda(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{x}) \odot \max(|\boldsymbol{x}| - \lambda\mathbf{1}, \mathbf{0}) \tag{14}$$

where $\odot$ is element-wise product, $\mathrm{sgn}(\cdot)$ and $\max(\cdot,\cdot)$ are element-wise sign and maximum functions respectively.

Updating $\boldsymbol{\theta}$ The sub-problem of updating θ is the inner maximization problem in Eq. (5), which we solve by the gradient ascent method:

$$\begin{aligned} &\max_{\theta}\ L_\theta(\theta), \\ &\text{where } L_\theta(\theta) = \mathbb{E}_{y\in\mathcal{Y}}[\log(D(y;\theta))] + \mathbb{E}_{x\in\mathcal{X}}[\log(1 - D(G_q(x);\theta))], \end{aligned} \tag{15}$$

We iteratively update D (parameterized by θ) and G (parameterized by W and γ) following [67].

3.3 Algorithm Implementation

Equipped with the above gradient computation, the last missing piece in solving problem (4) is to choose d. Note that, most previous distillation works are for classification-type models with softmax outputs (soft labels), and therefore adopt KL divergence. For GAN compression, the goal of distillation shall minimize the discrepancy between two sets of generated images. To this end, we adopt the perceptual loss [27] as our choice of d. It has shown to effectively measure not only low-level visual cue, but also high-level semantic differences between images, and has been popularly adopted to regularizing GAN-based image generation.

Finally, Algorithm 1 summarizes our GS algorithm with end-to-end optimization. By default, we quantize both activation and kernel weights uniformly to 8-bit (*i.e.*, $m = n = 8$) and set activation clamping threshold p to 4. We use Adam ($\beta_1 = 0.9$, $\beta_2 = 0.5$, following [67]) to update W and θ, and SGD to update γ. We also use two groups of learning rates $\alpha^{(t)}$ and $\eta^{(t)}$ for updating $\{W, \theta\}$ and γ respectively. $\alpha^{(t)}$ starts to be decayed linearly to zero, from the $T/2$-th iteration, while $\eta^{(t)}$ is decayed using a cosine annealing scheduler.

Algorithm 1: GAN Slimming (GS)

Input: $\mathcal{X}$, $\mathcal{Y}$, β, ρ, T, $\{\alpha^{(t)}\}_{t=1}^{T}, \{\eta^{(t)}\}_{t=1}^{T}$.
Output: W, γ

Random initialization: $W^{(1)}, \gamma^{(1)}, \theta^{(1)}$
for $t \leftarrow 1$ **to** T **do**
  Get a batch of data from $\mathcal{X}$ and $\mathcal{Y}$;
  $W^{(t+1)} \leftarrow W^{(t)} - \alpha^{(t)} \nabla_W L_W$;
  $\gamma^{(t+1)} \leftarrow \text{prox}_{\rho\eta^{(t)}}(\gamma^{(t)} - \eta^{(t)} \nabla_\gamma L_\gamma)$;
  $\theta^{(t+1)} \leftarrow \theta^{(t)} + \alpha^{(t)} \nabla_\theta L_\theta$;
end
$W \leftarrow q_w(W^{T+1})$
$\gamma \leftarrow \gamma^{T+1}$

4 Experiments

4.1 Unpaired Image Translation with CycleGAN

Image translation and stylization is currently an important motivating application to deploy GANs on mobile devices. In this section, we compare GS with the only two published GAN compression methods CEC [52] and GD [3] on horse2zebra [67] and summer2winter [67] datasets. Following [52], we use model size and FLOPs to measure the efficiency of generator and use FID [21] between source style test set transfer results and target style test set to quantitatively measure the effectiveness of style transfer. We used the same implementation of FID as [52] for fair comparison. The metric statistic of original CycleGAN is summarized in Table 1. We denote the original dense model as G_0 and an arbitrary compressed generator as G. Following [52], we further define the following three metrics to evaluate efficiency-quality trade-off of different compression methods:

$$r_c = \frac{\text{ModelSize}_{G_0}}{\text{ModelSize}_G},\ r_s = \frac{\text{FLOPs}_{G_0}}{\text{FLOPs}_G},\ r_f = \frac{\text{FID}_{G_0}}{\text{FID}_G}.$$

Larger r_c and r_s indicate more model compactness and efficiency and larger r_f indicates better style transfer quality.

Quantitative comparison results on four different tasks are shown in Table 2. GS-32 outperforms both CEC and GD on all four tasks, in terms that it achieves better FID (larger r_f) with less computational budgets (larger r_c and r_s). For example, on horse-to-zebra task, GS-32 has much better FID than both CEC and GD, while achieving more model compactness. Combined with quantization, our method can further boost the model efficiency (much larger r_c) with minimal loss of performance (similar r_f). For example, on horse-to-zebra task,

Table 1: Statistics of the original CycleGAN model: FLOPs, model size and FID on different tasks.

GFLOPs	Memory (MB)	FID			
		horse-to-zebra	zebra-to-horse	summer-to-winter	winter-to-summer
52.90	43.51	74.04	148.81	79.12	73.31

Table 2: Compassion with the state-of-the-art GAN compression methods [52] and [3] on CycleGAN compression. The best metric is shown in bold and the second best is underlined.

Task	Metric	Method			
		CEC [52]	GD [3]	GS-32	GS-8
horse-to-zebra	r_s	4.23	3.91	<u>4.66</u>	**4.81**
	r_c	4.27	4.00	<u>5.05</u>	**21.75**
	r_f	0.77	0.76	**0.86**	<u>0.84</u>
zebra-to-horse	r_s	4.35	3.91	<u>4.39</u>	**4.40**
	r_c	4.34	4.00	<u>4.81</u>	**21.00**
	r_f	0.94	0.99	<u>1.24</u>	**1.25**
summer-to-winter	r_s	5.14	3.91	<u>6.21</u>	**7.18**
	r_c	5.44	4.00	<u>6.77</u>	**38.10**
	r_f	1.01	1.08	**1.13**	<u>1.12</u>
winter-to-summer	r_s	5.17	3.91	<u>6.01</u>	**6.36**
	r_c	5.70	4.00	<u>6.17</u>	**31.22**
	r_f	0.93	0.97	<u>0.98</u>	**1.01**

GS-8 achieves 4× larger r_c compared with GS-32 with negligible FID drop. On winter-to-summer task, GS-8 compress CycleGAN by 31× and achieve even slightly better FID. The visual comparison results are collectively displayed in Fig. 2. We compare the transfer results of four images reported in [52] for fair comparison. As we can see, the visual quality of GS is better than or at least comparable to those of CEC and GD.

4.2 Ablation study

In order to show the superiority of our unified optimization framework over single compression methods and their naive combinations, we conduct thorough ablation studies by comparing the following methods:

- Distillation (*i.e.*, GD [3]): Use model distillation alone to train a slim student generator.[6]
- Channel pruning (CP): Directly use channel pruning during GAN minimax training process. This is implemented by adding L_{cp} to L_{GAN}. After channel pruning, we finetune the sub-network by minimax optimizing Eq. (1).

[6] Following [3], we use student networks with 1/2 channels of the original generator.

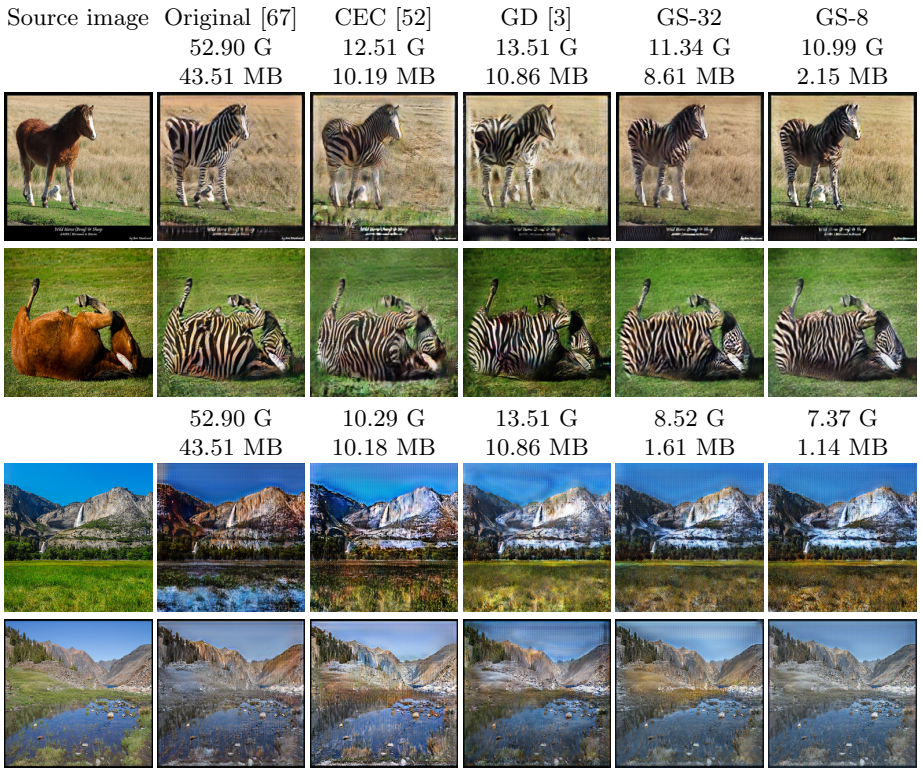


Fig. 2: CycleGAN compression results. Top two rows: horse-to-zebra task. Bottom two rows: summer-to-winter task. Six columns from left to right: source image, style transfer results by original CycleGAN, CEC, GD, GS-32 and GS-8 respectively. FLOPs (in G) and model size (in MB) of each method on each task are annotated above the images.

- Cascade: Distillation + CP (D+CP): Use channel pruning to further compress on the student network obtained by model distillation. Then finetune the sub-network by minimax optimizing Eq. (1).
- Cascade: CP + Distillation (CP+D): First do channel pruning on the original network, then use distillation to finetune the pruned network. This method is shown to outperform using channel pruning alone on classification tasks [57].
- GS-32: Jointly optimizing channel pruning and distillation.
- Cascade: GS-32 + quantization (postQ): First use GS-32 to compress the original network, then use 8 bit quantization as post processing and also do quantization-aware finetune on the quantized model by solving problem (1).
- GS-8: Jointly optimizing channel pruning, distillation and quantization.
- GS-8 (MSE): Replace the perceptual loss in GS-8 by MSE loss.
- GAN compression with fixed discriminator (*i.e.*, CEC [52]): Co-evolution based channel pruning. Modeling GAN compression as dense prediction pro-

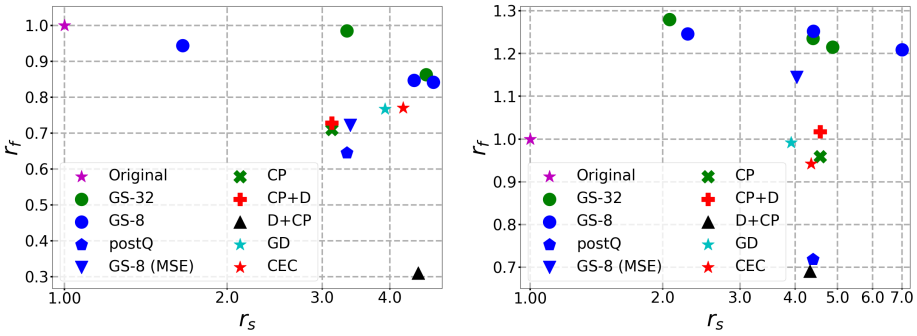
cess instead of minimax problem by fixing the discriminator (both network structure and parameter values) during compression process.

Numerical and visualization results on horse2zebra dataset are shown in Fig. 3 and Fig. 4 respectively. As we can see, our unified optimization method achieves superior trade-off between style transfer quality and model efficiency compared with single compression techniques used separately (*e.g.*, CP, GD) and their naive combinations (*e.g.*, CP+D, D+CP, postQ), showing the effectiveness of our unified optimization framework. For example, directly injecting channel sparsity in GAN minimax optimization (CP) greatly increases the training instability and achieves degraded image generation quality as shown in Fig. 4. This aligns with the conclusions in [52] that model compression methods developed for classifiers are not directly applicable on GAN compression tasks. Using model distillation to finetune channel pruned models (CP+D) can indeed improve image generation quality compared with CP, however the generation quality is still much more inferior to our methods at similar compression ratio, as shown in Fig. 3 and Fig. 4. Compared with GD, which uses a hand-crafted student network, GS-32 achieves much better FID with even considerably larger compression ratio, showing the effectiveness of jointly searching slim student network structures by channel pruning and training the student network with model distillation. In contrast, directly using channel pruning to further compress the student generator trained by GD (D+CP) will catastrophically hurt the image translation performance. Doing post quantization and quantization-aware finetune (postQ) on GS-32 models also suffers great degradation in image translation quality compared with GS-8, showing the necessity to jointly train quantization with channel pruning and model distillation in our unified optimization framework. Replacing perceptual loss with MSE loss as d in Eq. (2) fails to generate satisfying target images, since MSE loss cannot effectively capture the high-level semantic differences between images. Last but not least, GS-32 largely outperforms CEC, verifying the effectiveness of incorporating minimax objective into GAN compression problem.

4.3 Real-world Application: CartoonGAN

Finally, we apply GS to a recently proposed style transfer network CartoonGAN, which transforms photos to cartoon images, in order to deploy the model on mobile devices. CartoonGAN has its heavily parameterized generator (56.46 GFLOPs on 256×256 images) publicly available.[7] Since CartoonGAN has a feed-forward encoder-decoder structure, without using cycle consistent loss, CEC [52] is not directly applicable to compress it. So we only compare GS with the other published state-of-the-art method GD [3] on this task. Experiments are conducted on the CelebA dataset [40]. Following [3], we use a student generator with 1/6 channels of the teacher generator for GD, which achieves similar (but less) compression ratio compared with GS.

[7] Available at https://github.com/maciej3031/comixify.



(a) horse-to-zebra task (b) zebra-to-horse task

Fig. 3: Numerical results of ablation studies on horse2zebra dataset.

The visual results of cartoon style transfer, together with model statistics (FLOPs and model sizes), are shown in Fig. 5.[8] All FLOPs are calculated for input images with shape 256×256. At large compression ratio, the style transfer results of GD have obvious visual artifacts (*e.g.*, abnormal white spots). In contrast, GS-32 can remarkably compress the original generator by around $42\times$ (in terms of FLOPs) with minimal degradation in the visual quality. GS-8 can further improve the FLOPs compression ratio to $47\times$ with almost identical visual quality. These results again show the superiority of our student generator jointly learned by channel pruning, quantization and distillation, over the hand-crafted student generator used in GD. Part of the proposed GS framework is integrated into some style transfer products in Kwai Inc.'s Apps.

5 Conclusion

In this paper, we propose the first end-to-end optimization framework combining multiple compression techniques for GAN compression. Our method integrates model distillation, channel pruning and quantization, within one unified minimax optimization framework. Experimental results show that our method largely outperforms existing GAN compression options which utilize single compression techniques. Detailed ablation studies show that naively stacking different compression methods fails to achieve satisfying GAN compression results, sometimes even hurting the performance catastrophically, therefore testifying the necessity of our unified optimization framework.

[8] Following [10], we use color matching as the post-processing on all compared methods, for better visual display quality.

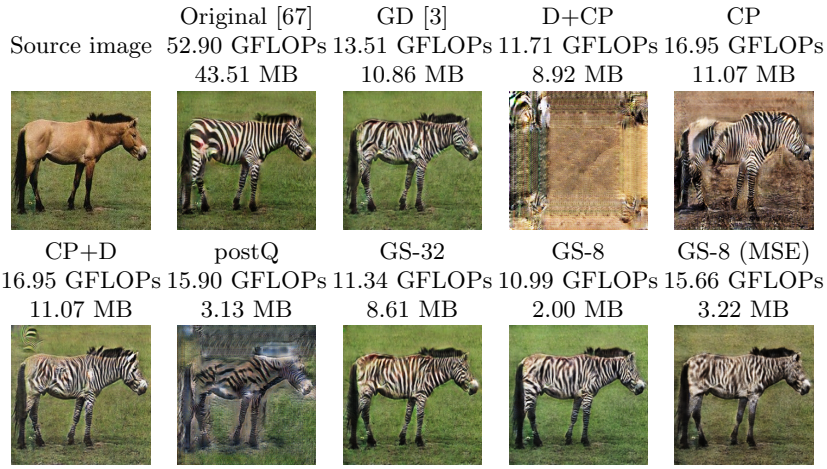


Fig. 4: Visualization results of ablation studies on horse2zebra dataset. FLOPs and model size of each method are annotated above the images.

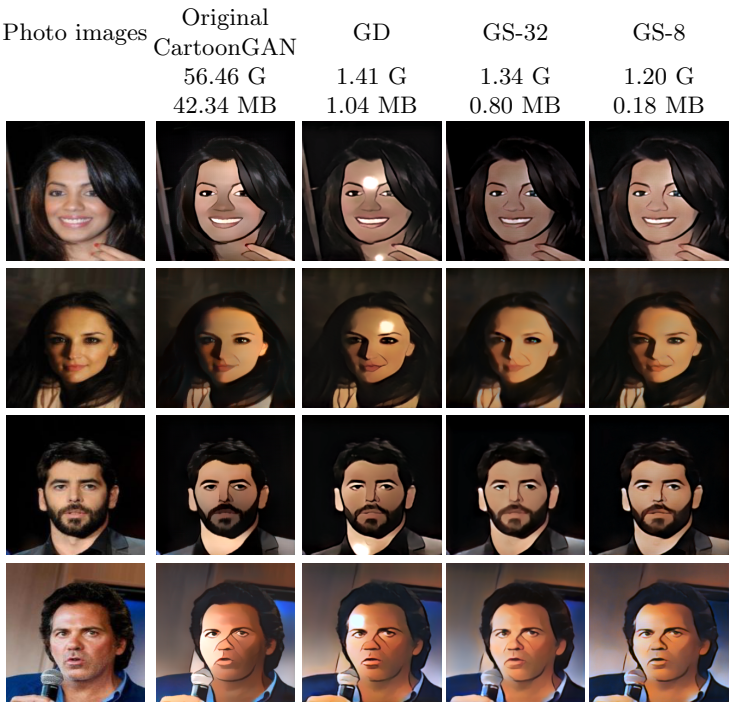


Fig. 5: CartoonGAN compression results. From left to right columns: original photo images, cartoon images generated by original CartoonGAN, GD [3], GS-32 and GS-8 compressed models, respectively. Corresponding FLOPs (in G) and model size (in MB) are annotated on top of each column.

References

1. Bulò, S.R., Porzi, L., Kontschieder, P.: Dropout distillation. In: International Conference on Machine Learning. pp. 99–107 (2016)
2. Chen, H., Wang, Y., Shu, H., Tang, Y., Xu, C., Shi, B., Xu, C., Tian, Q., Xu, C.: Frequency domain compact 3D convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1641–1650 (2020)
3. Chen, H., Wang, Y., Shu, H., Wen, C., Xu, C., Shi, B., Xu, C., Xu, C.: Distilling portable generative adversarial networks for image translation. In: AAAI Conference on Artificial Intelligence (2020)
4. Chen, H., Wang, Y., Xu, C., Xu, C., Tao, D.: Learning student networks via feature embedding. IEEE Transactions on Neural Networks and Learning Systems (2020)
5. Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., Tian, Q.: Data-free learning of student networks. In: IEEE International Conference on Computer Vision. pp. 3514–3522 (2019)
6. Chen, H., Wang, Y., Xu, C., Shi, B., Xu, C., Tian, Q., Xu, C.: AdderNet: Do we really need multiplications in deep learning? In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1468–1477 (2020)
7. Chen, Y., Lai, Y.K., Liu, Y.J.: CartoonGAN: Generative adversarial networks for photo cartoonization. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 9465–9474 (2018)
8. Courbariaux, M., Bengio, Y., David, J.P.: BinaryConnect: Training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems. pp. 3123–3131 (2015)
9. 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 (2020)
10. Gatys, L.A., Bethge, M., Hertzmann, A., Shechtman, E.: Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897 (2016)
11. Gong, X., Chang, S., Jiang, Y., Wang, Z.: AutoGAN: Neural architecture search for generative adversarial networks. In: IEEE International Conference on Computer Vision. pp. 3224–3234 (2019)
12. 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)
13. Gui, J., Sun, Z., Wen, Y., Tao, D., Ye, J.: A review on generative adversarial networks: Algorithms, theory, and applications. arXiv preprint arXiv:2001.06937 (2020)
14. Gui, S., Wang, H., Yang, H., Yu, C., Wang, Z., Liu, J.: Model compression with adversarial robustness: A unified optimization framework. In: Advances in Neural Information Processing Systems. pp. 1283–1294 (2019)
15. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems. pp. 5767–5777 (2017)
16. Guo, T., Xu, C., Huang, J., Wang, Y., Shi, B., Xu, C., Tao, D.: On positive-unlabeled classification in GAN. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 8385–8393 (2020)
17. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: GhostNet: More features from cheap operations. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1580–1589 (2020)

18. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149 (2015)
19. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: AMC: AutoML for model compression and acceleration on mobile devices. In: European Conference on Computer Vision. pp. 784–800 (2018)
20. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: IEEE International Conference on Computer Vision. pp. 1389–1397 (2017)
21. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in Neural Information Processing Systems. pp. 6626–6637 (2017)
22. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
23. Hu, H., Peng, R., Tai, Y.W., Tang, C.K.: Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint arXiv:1607.03250 (2016)
24. Huang, Z., Wang, N.: Data-driven sparse structure selection for deep neural networks. In: European Conference on Computer Vision. pp. 304–320 (2018)
25. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: Training neural networks with low precision weights and activations. Journal of Machine Learning Research **18**(1), 6869–6898 (2017)
26. Jiang, Y., Gong, X., Liu, D., Cheng, Y., Fang, C., Shen, X., Yang, J., Zhou, P., Wang, Z.: EnlightenGAN: Deep light enhancement without paired supervision. arXiv preprint arXiv:1906.06972 (2019)
27. Johnson, J., Alahi, A., Li, F.F.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. pp. 694–711 (2016)
28. Jung, S., Son, C., Lee, S., Son, J., Han, J.J., Kwak, Y., Hwang, S.J., Choi, C.: Learning to quantize deep networks by optimizing quantization intervals with task loss. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4350–4359 (2019)
29. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)
30. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
31. Kim, S., P Xing, E.: Tree-guided group lasso for multi-task regression with structured sparsity. The Annals of Applied Statistics **6**(3), 1095–1117 (2012)
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
33. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto (2009)
34. Kupyn, O., Martyniuk, T., Wu, J., Wang, Z.: DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better. In: IEEE International Conference on Computer Vision. pp. 8878–8887 (2019)
35. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4681–4690 (2017)

36. Li, M., Lin, J., Ding, Y., Liu, Z., Zhu, J.Y., Han, S.: GAN compression: Efficient architectures for interactive conditional GANs. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5284–5294 (2020)
37. Lin, J., Rao, Y., Lu, J., Zhou, J.: Runtime neural pruning. In: Advances in Neural Information Processing Systems. pp. 2181–2191 (2017)
38. Liu, S., Du, J., Nan, K., Wang, A., Lin, Y., et al.: Adadeep: A usage-driven, automated deep model compression framework for enabling ubiquitous intelligent mobiles. arXiv preprint arXiv:2006.04432 (2020)
39. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: IEEE International Conference on Computer Vision. pp. 2736–2744 (2017)
40. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: IEEE International Conference on Computer Vision. pp. 3730–3738 (2015)
41. Lopez-Paz, D., Bottou, L., Schölkopf, B., Vapnik, V.: Unifying distillation and privileged information. arXiv preprint arXiv:1511.03643 (2015)
42. Luo, J.H., Wu, J., Lin, W.: ThiNet: A filter level pruning method for deep neural network compression. In: IEEE International Conference on Computer Vision. pp. 5058–5066 (2017)
43. Mishra, A., Marr, D.: Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. arXiv preprint arXiv:1711.05852 (2017)
44. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018)
45. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 11264–11272 (2019)
46. Parikh, N., Boyd, S., et al.: Proximal algorithms. Foundations and Trends in Optimization **1**(3), 127–239 (2014)
47. Polino, A., Pascanu, R., Alistarh, D.: Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668 (2018)
48. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet classification using binary convolutional neural networks. In: European Conference on Computer Vision. pp. 525–542 (2016)
49. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: Advances in Neural Information Processing Systems. pp. 2234–2242 (2016)
50. Sanakoyeu, A., Kotovenko, D., Lang, S., Ommer, B.: A style-aware content loss for real-time HD style transfer. In: European Conference on Computer Vision. pp. 698–714 (2018)
51. Shen, M., Han, K., Xu, C., Wang, Y.: Searching for accurate binary neural architectures. In: IEEE International Conference on Computer Vision Workshops (2019)
52. Shu, H., Wang, Y., Jia, X., Han, K., Chen, H., Xu, C., Tian, Q., Xu, C.: Co-Evolutionary compression for unpaired image translation. In: IEEE International Conference on Computer Vision. pp. 3235–3244 (2019)
53. Singh, P., Verma, V.K., Rai, P., Namboodiri, V.: Leveraging filter correlations for deep model compression. In: IEEE Winter Conference on Applications of Computer Vision. pp. 835–844 (2020)
54. Theis, L., Korshunova, I., Tejani, A., Huszár, F.: Faster gaze prediction with dense networks and fisher pruning. arXiv preprint arXiv:1801.05787 (2018)

55. Tung, F., Mori, G.: CLIP-Q: Deep network compression learning by in-parallel pruning-quantization. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 7873–7882 (2018)
56. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: HAQ: Hardware-Aware automated quantization with mixed precision. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 8612–8620 (2019)
57. Wang, M., Zhang, Q., Yang, J., Cui, X., Lin, W.: Graph-adaptive pruning for efficient inference of convolutional neural networks. arXiv preprint arXiv:1811.08589 (2018)
58. Wang, Y., Xu, C., Chunjing, X., Xu, C., Tao, D.: Learning versatile filters for efficient convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1608–1618 (2018)
59. Wang, Y., Xu, C., Xu, C., Tao, D.: Adversarial learning of portable student networks. In: AAAI Conference on Artificial Intelligence. pp. 4260–4267 (2018)
60. Wang, Y., Xu, C., Xu, C., Tao, D.: Packing convolutional neural networks in the frequency domain. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(10), 2495–2510 (2018)
61. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems. pp. 2074–2082 (2016)
62. Wu, J., Wang, Y., Wu, Z., Wang, Z., Veeraraghavan, A., Lin, Y.: Deep-k-Means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In: International Conference on Machine Learning. pp. 5363–5372 (2018)
63. Yang, H., Zhu, Y., Liu, J.: ECC: Platform-independent energy-constrained deep neural network compression via a bilinear regression model. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 11206–11215 (2019)
64. Yang, S., Wang, Z., Wang, Z., Xu, N., Liu, J., Guo, Z.: Controllable artistic text style transfer via shape-matching GAN. In: IEEE International Conference on Computer Vision. pp. 4442–4451 (2019)
65. Zhao, Y., Chen, X., Wang, Y., Li, C., You, H., Fu, Y., Xie, Y., Wang, Z., Lin, Y.: Smartexchange: Trading higher-cost memory storage/access for lower-cost computation. arXiv preprint arXiv:2005.03403 (2020)
66. Zhu, C., Han, S., Mao, H., Dally, W.J.: Trained ternary quantization. arXiv preprint arXiv:1612.01064 (2016)
67. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision. pp. 2223–2232 (2017)

A Image Generation with SNGAN

We have demonstrated the effectiveness of GS in compressing image-to-image GANs (*e.g.*, CycleGAN [67], StyleGAN [50]) in the main text. Here we show GS is also generally applicable to noise-to-image GANs (*e.g.*, SNGAN [44]). SNGAN with the ResNet backbone is one of the most popular noise-to-image GANs, with state-of-the-art performance on a few datasets such as CIFAR10 [33]. The generator in SNGAN has 7 convolution layers with 1.57 GFLOPs, with 32×32 image outputs. We evaluate SNGAN generator compression on the CIFAR-10 dataset. Inception Score (IS) [49] is used to measure image generation and style transfer quality. We use latency (FLOPs) and model size to evaluate the network efficiency. Quantative and visualization results are shown in Table 3 and Figure 6 respectively. GS is able to compress SNGAN by up to $8\times$ (in terms of model size), with minimum drop in both visual quality and the quantitative IS value of generated images.

Table 3: SNGAN compression results.

Method	MFLOPs	Model Size (MB)	IS
Original	1602.75	16.28	8.27
GS-32	1108.78	12.88	8.01
	509.39	8.32	7.65
GS-8	1115.11	3.24	8.14
	510.33	2.01	7.62

Fig. 6: CIFAR-10 images generation by SNGAN (original and compressed). Leftmost column: images generated by original SNGAN. The rest columns: images generated by GS-8 compressed SNGAN, with different compression ratios. Images are randomly selected instead of cherry-picked.