GAN Compression: Efficient Architectures for Interactive Conditional GANs

Muyang Li[1,3] Ji Lin[1] Yaoyao Ding[1,3] Zhijian Liu[1] Jun-Yan Zhu[2] Song Han[1]
lmxyy@mit.edu jilin@mit.edu yyding@mit.edu zhijian@mit.edu junzhu@adobe.com songhan@mit.edu
[1]Massachusetts Institute of Technology [2]Adobe Research [3]Shanghai Jiao Tong University

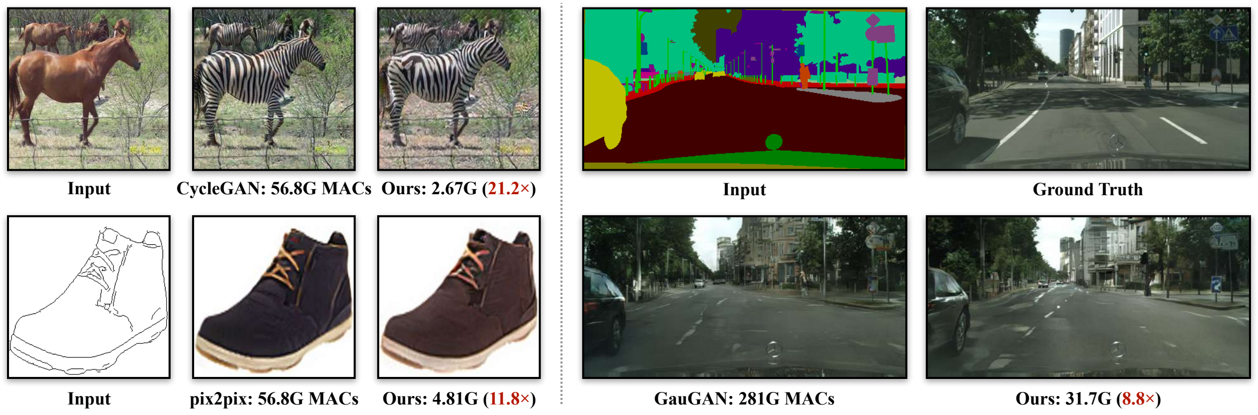


Figure 1: We introduce *GAN Compression*, a general-purpose method for compressing conditional GANs. Our method reduces the computation of widely-used conditional GAN models including pix2pix, CycleGAN, and GauGAN by 9-21× while preserving the visual fidelity. Our method is effective for a wide range of generator architectures, learning objectives, and both paired and unpaired settings.

Abstract

Conditional Generative Adversarial Networks (cGANs) have enabled controllable image synthesis for many computer vision and graphics applications. However, recent cGANs are 1-2 orders of magnitude more computationally-intensive than modern recognition CNNs. For example, GauGAN consumes 281G MACs per image, compared to 0.44G MACs for MobileNet-v3, making it difficult for interactive deployment. In this work, we propose a general-purpose compression framework for reducing the inference time and model size of the generator in cGANs. Directly applying existing CNNs compression methods yields poor performance due to the difficulty of GAN training and the differences in generator architectures. We address these challenges in two ways. First, to stabilize the GAN training, we transfer knowledge of multiple intermediate representations of the original model to its compressed model, and unify unpaired and paired learning. Second, instead of reusing existing CNN designs, our method automatically finds efficient architectures via neural architecture search (NAS). To accelerate the search process, we decouple the model training and architecture search via weight sharing. Experiments demonstrate the effectiveness of our method across different supervision settings (paired and unpaired), model architectures, and learning methods (e.g., pix2pix, GauGAN, CycleGAN). Without losing image quality, we reduce the computation of CycleGAN by more than 20× and GauGAN by 9×, paving the way for interactive image synthesis. The code and demo are publicly available.

1. Introduction

Generative Adversarial Networks (GANs) [14] excel at synthesizing photo-realistic images. Their conditional extension, conditional GANs [44, 27, 69], allows controllable image synthesis and enables many computer vision and graphics applications such as interactively creating an image from a user drawing [45], transferring the motion of a dancing video stream to a different person [57, 8, 1], or creating VR facial animation for remote social interaction [59]. All of these applications require models to interact with humans and therefore demand low-latency on-device performance for better user experience. However, edge devices (mobile phones, tablets, VR headsets) are tightly constrained by hardware resources such as memory and battery. This computational bottleneck prevents conditional GANs from being deployed on edge devices.

Different from image recognition CNNs [31, 53, 19, 25], image-conditional GANs are notoriously computationally intensive. For example, the widely-used CycleGAN model [69] requires more than 50G MACs[*], 100× more

[*]We use the number of Multiply-Accumulate Operations (MAC) to

than MobileNet [25]. A more recent model GauGAN [45], though generating photo-realistic high-resolution images, requires more than 250G MACs, 500× more than MobileNet [25, 49, 24].

In this work, we present *GAN Compression*, a general-purpose compression method for reducing the inference time and computational cost for conditional GANs. We observe that compressing generative models faces two fundamental difficulties: GANs are unstable to train, especially under the unpaired setting; generators also differ from recognition CNNs, making it hard to reuse existing CNN designs. To address these issues, we first transfer the knowledge from the intermediate representations of the original teacher generator to its corresponding layers of its compressed student generator. We also find it beneficial to create pseudo pairs using the teacher model's output for unpaired training. This transforms the unpaired learning to a paired learning. Second, we use neural architecture search (NAS) to automatically find an efficient network with significantly fewer computation costs and parameters. To reduce the training cost, we decouple the model training from architecture search by training a "once-for-all network" that contains all possible channel number configurations. The once-for-all network can generate many sub-networks by weight sharing and enable us to evaluate the performance of each sub-network without retraining. Our method can be applied to various conditional GAN models regardless of model architectures, learning algorithms, and supervision settings (paired or unpaired).

Through extensive experiments, we show that our method can reduce the computation of three widely-used conditional GAN models including pix2pix [27], CycleGAN [69], and GauGAN [45] by 9× to 21× regarding MACs, without loss of the visual fidelity of generated images (see Figure 1 for several examples). Finally, we deploy our compressed pix2pix model on a mobile device (Jetson Nano) and demonstrate an interactive edges2shoes application [demo].

2. Related Work

Conditional GANs. Generative Adversarial Networks (GANs) [14] are excel at synthesizing photo-realistic results [29, 5]. Its conditional form, conditional GANs [44, 27] further enables controllable image synthesis, allowing a user to synthesize images given various conditional inputs such as user sketches [27, 50], class labels [44, 5], or textual descriptions [47, 67]. Subsequent works further increased the resolution and realism of the results [58, 45]. Later, several algorithms were proposed to learn conditional GANs without paired data [55, 51, 69, 30, 62, 38, 11, 26, 32].

The high-resolution, photo-realistic synthesized results come at the cost of intensive computation. As shown in

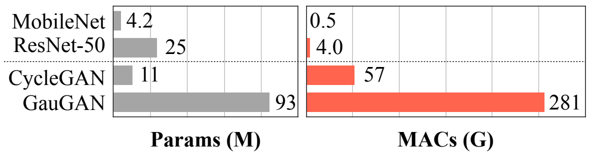


Figure 2: Conditional GANs require two orders of magnitude more computation than image classification CNNs, making it prohibitive to be deployed on edge devices.

Figure 2, although the model size is of the same magnitude as the size of image recognition CNNs [19], conditional GANs require two orders of magnitudes more computations. This makes it challenging to deploy these models on edge devices given limited computational resources. In this work, we focus on efficient image-conditional GANs architectures for interactive applications.

Model acceleration. Extensive attention has been paid to hardware-efficient deep learning for various real-world applications [18, 17, 68, 56, 16]. To reduce redundancy in network weights, researchers proposed to prune the connections between layers [18, 17, 60]. However, the pruned networks require specialized hardware to achieve its full speedup. Several subsequent works proposed to prune entire convolution filters [21, 34, 39] to improve the regularity of computation. AutoML for Model Compression (AMC) [20] leverages reinforcement learning to determine the pruning ratio of each layer automatically. Liu *et al.* [40] later replaced the reinforcement learning by an evolutionary search algorithm. Recently, Shu *et al.* [52] proposed co-evolutionary pruning for CycleGAN by modifying the original CycleGAN algorithm. This method is tailored for a particular algorithm. The compressed model significantly increases FID under a moderate compression ratio (4.2×). In contrast, our model-agnostic method can be applied to conditional GANs with different learning algorithms, architectures, and both paired and unpaired settings. We assume no knowledge of the original cGAN learning algorithm. Experiments show that our general-purpose method achieves 21.1× compression ratio (5× better than CycleGAN-specific method [52]) while retaining the FID of original models.

Knowledge distillation. Hinton *et al.* [23] introduced the knowledge distillation for transferring the knowledge in a larger teacher network to a smaller student network. The student network is trained to mimic the behavior of the teacher network. Several methods leverage knowledge distillation for compressing recognition models [43, 9, 33]. Recently, Aguinaldo *et al.* [2] adopts this method to accelerate unconditional GANs. Different from them, we focus on conditional GANs. We experimented with several distillation methods [2, 63] on conditional GANs and only observed marginal improvement, insufficient for interactive applications. Please refer to our arXiv for more details.

quantify the computation cost. Modern computer architectures use fused multiply–add (FMA) instructions for tensor operations. These instructions compute $a = a + b \times c$ as one operation. 1 MAC=2 FLOPs.

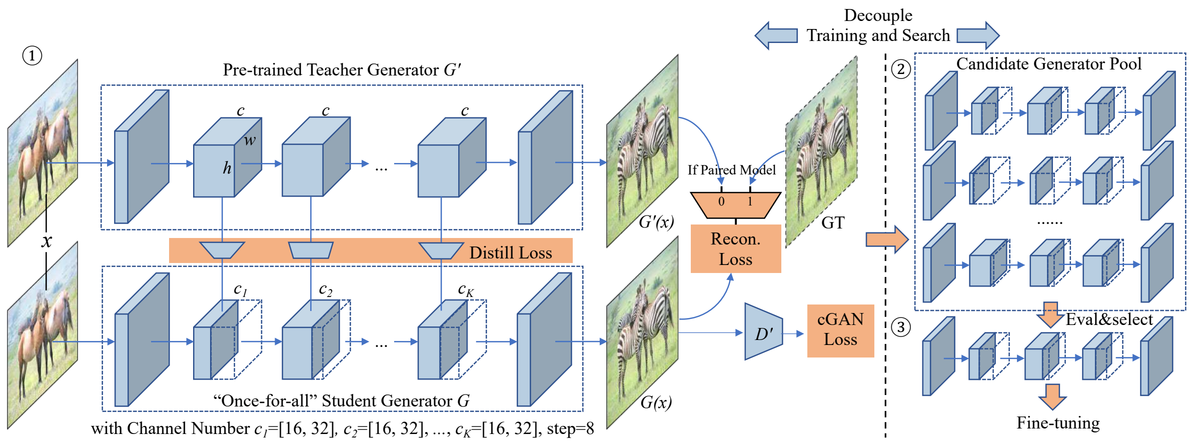

①
Pre-trained Teacher Generator G'
c
w
h
x
Distill Loss
c_1
c_2
c_K
"Once-for-all" Student Generator G
with Channel Number c_1=[16, 32], c_2=[16, 32], ..., c_K=[16, 32], step=8
$G'(x)$
$G(x)$
If Paired Model
0
1
GT
Recon. Loss
D'
cGAN Loss
Decouple Training and Search
②
Candidate Generator Pool
③
Eval&select
Fine-tuning

Figure 3: GAN Compression: ① Given a pre-trained teacher generator G', we distill a smaller "*once-for-all*" [6] student generator G that contains all possible channel numbers through weight sharing. We sample different channel numbers $\{c_k\}_{k=1}^K$ for G at each training step so that *one* generator can support *all* channel numbers. ② We then extract many sub-generators with different channel numbers from the "once-for-all" generator and evaluate their performance. No retraining is needed, which is the advantage of the "once-for-all" generator. ③ Finally, we choose the best sub-generator given the compression ratio target and performance target (FID or mAP), perform fine-tuning, and obtain the final compressed model.

Neural architecture search. Neural Architecture Search (NAS) has successfully designed neural network architectures that outperform hand-crafted ones for image recognition tasks [71, 35, 36]. To reduce the search cost, researchers recently proposed one-shot neural architecture search [37, 7, 61, 15, 24, 4, 6] in which different sub-networks can share the weights. However, little efforts has been paid to search efficient GAN architectures. We study efficient conditional GANs architectures using NAS.

3. Method

Compressing conditional generative models for interactive applications is challenging for two reasons. Firstly, the training dynamic of GANs is highly unstable by nature. Secondly, the architectural differences between recognition and generative models make it hard to apply existing CNN compression algorithms directly. To address these issues, we propose a new training protocol tailored for efficient generative models (Section 3.1) and further increase the compression ratio with neural architecture search (NAS) (Section 3.2). The overall framework is illustrated in Figure 3.

3.1. Training Objective

Unifying unpaired and paired learning. Conditional GANs aim to learn a mapping function G between a source domain X and a target domain Y. They can be trained using either *paired* data ($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ where $\mathbf{x}_i \in X$ and $\mathbf{y}_i \in Y$) or *unpaired* data (source dataset $\{\mathbf{x}_i\}_{i=1}^N$ to target dataset $\{\mathbf{y}_j\}_{j=1}^M$). Here, N and M denote the number of training images. For simplicity, we omit the subscript i and j. Several learning objectives have been proposed to handle both paired and unpaired settings (e.g., [27, 45, 58, 69, 38, 26]). The wide range of training objectives makes it difficult to build a *general-purpose* compression framework. To address this limitation, we unify the unpaired and paired learning in the model compression process, regardless of how the teacher model is originally trained. Given the original teacher generator G', we can transform the unpaired training setting to the paired setting. In particular, for the unpaired setting, we can view the original generator's output as our ground-truth and train our compressed generator G with a paired learning objective. Our learning objective can be summarized as:

$$\mathcal{L}_{\text{recon}} = \begin{cases} \mathbb{E}_{\mathbf{x},\mathbf{y}} \|G(\mathbf{x}) - \mathbf{y}\|_1 & \text{if paired cGANs,} \\ \mathbb{E}_{\mathbf{x}} \|G(\mathbf{x}) - G'(\mathbf{x})\|_1 & \text{if unpaired cGANs.} \end{cases} \quad (1)$$

Here we denote $\mathbb{E}_{\mathbf{x}} \triangleq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}$ and $\mathbb{E}_{\mathbf{x},\mathbf{y}} \triangleq \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}$ for simplicity. $\|\|_1$ denotes L1 norm. With such modifications, we can apply the same compression framework to different types of cGANs. Furthermore, As shown in Section 4.3, learning using the above pseudo pairs makes training more stable and yields much better results, compared to the original unpaired training setting.

As the unpaired training has been transformed into paired training, we will discuss the following sections in the paired training setting unless otherwise specified.

Inheriting the teacher discriminator. Although we aim to compress the generator, a discriminator D stores useful knowledge of a learned GAN as D learns to spot the weakness of the current generator [3]. Therefore, we adopt the same discriminator architecture, use the pre-trained weights from the teacher, and fine-tune the discriminator together with our compressed generator. In our experiments, we observe that a pre-trained discriminator is better than a randomly initialized discriminator which leads to severe training instability and the degradation of image quality. The GAN objective is formalized as:

$$\mathcal{L}_{\text{cGAN}} = \mathbb{E}_{\mathbf{x},\mathbf{y}}[\log D(\mathbf{x},\mathbf{y})] + \mathbb{E}_{\mathbf{x}}[\log(1 - D(\mathbf{x}, G(\mathbf{x})))] \quad (2)$$

where we initialize the student discriminator D using the weights from teacher discriminator D'. G and D are trained using a standard minimax optimization [14].

Intermediate feature distillation. A widely-used method for model compression is knowledge distillation, which matches the distribution of the output layer's logits [23, 43, 9, 63, 33, 46, 10]. However, conditional GANs [27, 69] usually output a deterministic image, rather than a probabilistic distribution. Therefore, it is difficult to distill the dark knowledge from the teacher's output pixels. Especially for paired training setting, output images generated by the teacher model essentially contains no additional information compared to ground-truth target images. Experiments in our arXiv show that for paired training, naively mimicking the teacher model's output brings no improvement.

To address the issue, we match the intermediate representations of the teacher generator instead, as explored in prior work [33, 66, 9]. The intermediate layers contain more channels, provide richer information, and allow the student model to acquire more information in addition to outputs. The distillation objective can be formalized as:

$$\mathcal{L}_{\text{distill}} = \sum_{t=1}^{T} \|G_t(\mathbf{x}) - f_t(G'_t(\mathbf{x}))\|_2, \tag{3}$$

where $G_t(\mathbf{x})$ and $G'_t(\mathbf{x})$ are the intermediate feature activations of the t-th chosen layer in the student and teacher models, and T denotes the number of layers. f_t is a 1×1 learnable convolution that maps the channels from the student model to the teacher model, since they have different channel numbers. We jointly optimize G_t and f_t to minimize the distillation loss $\mathcal{L}_{\text{distill}}$. Our arXiv details which layers we choose in practice.

Full objective. Our final objective is written as follows:

$$\mathcal{L} = \mathcal{L}_{\text{cGAN}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{distill}}\mathcal{L}_{\text{distill}}, \tag{4}$$

where hyper-parameters λ_{recon} and λ_{distill} control the importance of each term. Refer to our arXiv for more details.

3.2. Efficient Generator Design Space

Choosing an efficient student generator architecture is essential for knowledge distillation. We find naively shrinking the channel numbers of the teacher model fails to produce a compact student model: the performance starts to degrade significantly above $4\times$ computation reduction. The reason is that existing generator architectures are adopted from image recognition models [41, 19, 48, 41], which are not optimal for image synthesis. We show how we derive a better architecture design space from an existing cGAN generator and perform neural architecture search (NAS) within the space.

Convolution decomposition and layer sensitivity. Existing generators usually adopt vanilla convolutions to follow the design of classification and segmentation CNNs. Recent efficient CNN designs widely adopt a decomposed version of convolutions (depthwise + pointwise) [25], which proves to have a better performance-computation trade-off. We find that using the decomposed convolution also benefits the generator design in cGANs.

Unfortunately, our early experiments show that naively applying decomposition to all the convolution layers (as in classifiers) will significantly degrade the image quality. Decomposing some of the layers will immediately hurt the performance, while other layers are more robust. Furthermore, this layer sensitivity pattern is not the same as recognition models. For example, in ResNet generator [19, 28], the resBlock layers consume the majority of the model parameters and computation cost while is almost immune to decomposition. On the contrary, the upsampling layers have much fewer parameters, but are fairly sensitive to model compression: moderate compression can lead to a large FID degradation. Therefore, we only decompose the resBlock layers. We conduct a comprehensive study regarding the sensitivity of layers in Section 4.3.

Automated channel reduction with NAS. Existing generators use a hand-crafted (and mostly uniform) channel numbers across all the layers, which contains redundancy and is far from optimal. To further improve the compression ratio, we select the channel width in the generators using automated channel pruning [20, 39, 70, 42] to remove the redundancy, which can reduce the computation quadratically. We support fine-grained choices regarding the numbers of channels. For each convolution layers, the number of channels can be chosen from multiples of 8, which balances MACs and hardware parallelism [20].

Given the possible channel configurations $\{c_1, c_2, ..., c_K\}$, where K is the number of layers to prune, our goal is to find the best channel configuration $\{c_1^*, c_2^*, ..., c_K^*\} = \arg\min_{c_1, c_2, ..., c_K} \mathcal{L}, \quad s.t.\ \text{MACs} < F_t$ using neural architecture search, where F_t is the computation constraint. A straight-forward approach is to traverse all the possible channel configuration, train it to convergence, evaluate, and pick the generator with the best performance. However, as K increases, the number of possible configurations increases exponentially, and each configuration might require different hyper-parameters regarding the learning rates and weights for each term. This trial and error process is far too time-consuming.

3.3. Decouple Training and Search

To address the problem, we decouple model training from architecture search, following recent work in one-shot neural architecture search methods [7, 6, 15]. We first train a "once-for-all" network [6] that supports different channel numbers. Each sub-network with different numbers of channels are equally trained and can operate independently. Sub-networks share the weights with the "once-for-all" network. Figure 3 illustrates the overall framework. We assume that the original teacher generator has $\{c_k^0\}_{k=1}^K$ channels. For a given channel number configuration $\{c_k\}_{k=1}^K, c_k \leq c_k^0$, we obtain the weight of the sub-network by extracting the first $\{c_k\}_{k=1}^K$ channels from the corresponding weight tensors of "once-for-all" network, following Guo et al. [15]. At each training step, we sample a sub-network with a randomly channel number configuration, compute the output and gradients, and update the extracted weights using our learning objective (Equation 4). Since the weights at the first several channels are updated more frequently, they play a more critical role among all the weights.

After the "once-for-all" network is trained, we find the best sub-network by *directly* evaluating the performance of each candidate sub-network on the validation set. Since the "once-for-all" network is thoroughly trained with weight sharing, no fine-tuning is needed. This approximates the model performance when it is trained from scratch. In this manner, we can decouple the training and search of the generator architecture: we only need to train once, but we can evaluate many different channel configurations without further training, and pick the best one as the search result. Optionally, we fine-tune the selected architecture to further improve the performance. We report both variants in Section 4.3.

4. Experiments

4.1. Models, Datasets, Evaluation Metrics

Models. We conduct experiments on three conditional GAN models to demonstrate the generality of our method.

- CycleGAN [69], an unpaired image-to-image translation model, uses a ResNet-based generator [19, 28] to transform an image from a source domain to a target domain.
- Pix2pix [27] is a conditional-GAN based paired image-to-image translation model. For this model, we replace the original U-Net generator [48] by the ResNet-based generator [28] as we observe that the ResNet-based generator achieves better results with less computation cost. See our arXiv version for a detailed U-Net vs. ResNet comparison.
- GauGAN [45] is a state-of-the-art paired image-to-image translation model. It can generate a high-fidelity image given a semantic label map.

Datasets. We use the following four datasets:

- Edges$\rightarrow$shoes. 50,025 images from UT Zappos50K dataset [64]. We evaluate pix2pix model on this dataset.
- Cityscapes. The dataset [12] contains the images of German street scenes. The training set and the validation set consists of 2975 and 500 images, respectively. We evaluate pix2pix and GauGAN model on this dataset.
- Horse$\leftrightarrow$zebra. The dataset consists of 1,187 horse images and 1,474 zebra images originally from ImageNet [13] and used in CycleGAN [69]. We evaluate the CycleGAN model on this dataset.
- Map$\leftrightarrow$aerial photo. The dataset contains 2194 images scraped from Google Maps and used in pix2pix [27]. We evaluate the pix2pix model on this dataset.

Evaluation Metrics.

- Fréchet Inception Distance (FID) [22]. The FID score aims to calculate the distance between the distribution of feature vectors extracted from real and generated images using an InceptionV3 [54] network. A *lower* score indicates a *better* quality of generated images. We use an open-sourced FID evaluation code[†].
- Semantic Segmentation Metrics. Following prior work [27, 69, 45], we adopt a semantic segmentation metric to evaluate the generated images on the Cityscapes dataset. We run a semantic segmentation model on the generated images and compare how well the segmentation model performs. We use the Mean Average Precision (mAP), and use DRN-D-105 [65] as our segmentation model. *Higher* mAPs implies that the generated images look more *realistic* and better reflects the input label map.

4.2. Results

Quantitative Results We report the quantitative results of compressing CycleGAN, pix2pix, and GuaGAN on four datasets in Table 1. By using the best performing sub-network from the "once-for-all" network, *GAN Compression* can compress state-of-the-art conditional GANs by **9-21**$\times$, and reduce the model size by **5-33**$\times$, with only negligible degradation in the model performance. Specifically, our proposed method shows a clear advantage of CycleGAN compression compared to the previous Co-Evolution method [52]. We can reduce the computation of CycleGAN generator by 21.2$\times$, which is 5$\times$ better compared to the previous CycleGAN-specific method [52] while achieving a better FID by more than 30.

Performance *vs*. Computation Trade-off Apart from the large compression ratio, our method consistently improves the performance at different model sizes. Taking the pix2pix model as an example, we plot the performance *vs*. computation trade-off on Cityscapes and Edges$\rightarrow$shoes dataset in Figure 6. In the large model size regime, prune + distill (without NAS) outperforms training from scratch, showing the effectiveness of intermediate layer distillation. Unfortunately, when the channels continue to shrink *uniformly* if

[†]https://github.com/mseitzer/pytorch-fid

Model	Dataset	Method	#Parameters		MACs		Metric			
							FID (↓)		mAP (↑)	
CycleGAN	horse→zebra	Original	11.3M	–	56.8G	–	61.53	–	–	
		Shu *et al.* [52]	–	–	13.4G	(4.2×)	96.15	(34.6 ☹)	–	
		Ours (w/o fine-tuning)	0.34M	(**33.3**×)	2.67G	(**21.2**×)	64.95	(**3.42** ☹)	–	
		Ours	0.34M	(33.3×)	2.67G	(21.2×)	71.81	(10.3 ☹)	–	
Pix2pix	edges→shoes	Original	11.3M	–	56.8G	–	24.18	–	–	
		Ours (w/o fine-tuning)	0.70M	(16.3×)	4.81G	(11.8×)	31.30	(7.12 ☹)	–	
		Ours	0.70M	(**16.3**×)	4.81G	(**11.8**×)	26.60	(**2.42** ☹)	–	
	cityscapes	Original	11.3M	–	56.8G	–	–		35.62	–
		Ours (w/o fine-tuning)	0.71M	(16.0×)	5.66G	(10×)	–		29.27	(6.35 ☹)
		Ours	0.71M	(**16.0**×)	5.66G	(**10.0**×)	–		34.34	(**1.28** ☹)
	map→arial photo	Original	11.3M	–	56.8G	–	47.76	–	–	
		Ours (w/o fine-tuning)	0.75M	(15.1×)	4.68G	(11.4×)	71.82	(24.1 ☹)	–	
		Ours	0.75M	(**15.1**×)	4.68G	(**11.4**×)	48.02	(**0.26** ☹)	–	
GauGAN	cityscapes	Original	93.0M	–	281G	–	–		58.89	–
		Ours (w/o fine-tuning)	20.4M	(4.6×)	31.7G	(8.8×)	–		56.75	(2.14 ☹)
		Ours	20.4M	(**4.6**×)	31.7G	(**8.8**×)	–		58.41	(**0.48** ☹)

Table 1: Quantitative evaluation of GAN Compression: We use the mAP metric (the higher the better) for the Cityscapes dataset and FID (the lower the better) for other datasets. Our method can compress state-of-the-art conditional GANs by **9-21**× in MACs and **5-33**× in model size, with only minor performance degradation. For CycleGAN compression, our systematic approach outperforms previous CycleGAN-specific Co-Evolution method [52] by a large margin.

Model		CycleGAN	Pix2pix	GauGAN
Metric	FID (↓)	61.5→65.0	24.2→26.6	–
	mAP (↑)	–	–	58.9 → 58.4
MAC Reduction		21.2×	11.8×	8.8×
Memory Reduction		2.0×	1.7×	1.8×
Xavier Speedup	CPU	1.65s (18.5×)	3.07s (9.9×)	21.2s (7.9×)
	GPU	0.026s (3.1×)	0.035s (2.4×)	0.10s (3.2×)
Nano Speedup	CPU	6.30s (14.0×)	8.57s (10.3×)	65.3s (8.6×)
	GPU	0.16s (4.0×)	0.26s (2.5×)	0.81s (3.3×)
1080Ti Speedup		0.005s (2.5×)	0.007s (1.8×)	0.034s (1.7×)
Xeon Silver 4114 CPU Speedup		0.11s (3.4×)	0.15s (2.6×)	0.74s (2.8×)

Table 2: Measured memory reduction and latency speedup on NVIDIA Jetson AGX Xavier, NVIDIA Jetson Nano, 1080 Ti GPU and Xeon CPU. CycleGAN, pix2pix, and GauGAN are trained on horse→zebra, edges→shoes and Cityscapes.

Model	ngf	FID	MACs	#Parameters
Original	64	61.75	56.8G	11.38M
Only change downsample	64	68.72	55.5G	11.13M
Only change upsample	64	61.04	48.3G	11.05M
Only change resBlocks	64	62.95	**18.3G**	**1.98M**
Only change downsample	16	74.77	3.6G	0.70M
Only change upsample	16	95.54	3.3G	0.70M
Only change resBlocks	16	79.49	**1.4G**	**0.14M**

Table 3: We report the performance after applying convolution decomposition in each of the three parts (Downsample, ResBlocks, and Upsample) of the ResNet generator respectively on the horse→zebra dataset. `ngf` denotes the **n**umber of the **g**enerator's **f**ilters. Both computation and model size are proportional to $\texttt{ngf}^2$. We evaluate two settings `ngf`=64 and `ngf`=16. We observe that modifying ResBlock blocks shows a significantly better performance *vs.* computation trade-off, compared to modifying other parts of the network.

without NAS, some sensitive channels are pruned too much. As a result, the knowledge from the teacher may be too little for the student, in which case the distillation may even have negative effects on the student model. On the contrary, our training strategy allows us to automatically find the best channel number, leading to a smaller gap between the student and teacher model.

Qualitative Results Figure 4 shows several example results. We provide the input, its ground-truth (except for unpaired setting), the output of the original model, and the output of our compressed model. Our compression method well preserves the visual fidelity of the output image even under a large compression ratio. For CycleGAN, we also provide the output of a baseline model (0.25 CycleGAN: 14.9×). The baseline model 0.25 CycleGAN contains $\frac{1}{4}$ channels and has been trained from scratch. Our advantage is distinct: the baseline model can hardly create a zebra pattern on the output image, given a much smaller compression ratio. There might be some cases where compressed models show a small degradation (*e.g.*, the leg of the second zebra in Figure 4), but compressed models sometimes surpass the original one in other cases (*e.g.*, the first and last shoe images have a better leather texture). Generally, GAN models compressed by our method performs comparatively compared to the original model, as shown by quantitative results.

Accelerate Inference on Hardware For real-world interactive applications, inference acceleration on hardware is more critical than the reduction of computation. To verify the practical effectiveness of our method, we measure the inference speed of our compressed models on several devices with different computuatoinal powers. The results are shown in Table 2. The inference speed of compressed CycleGAN generator on Jetson Xavier GPU can achieve about

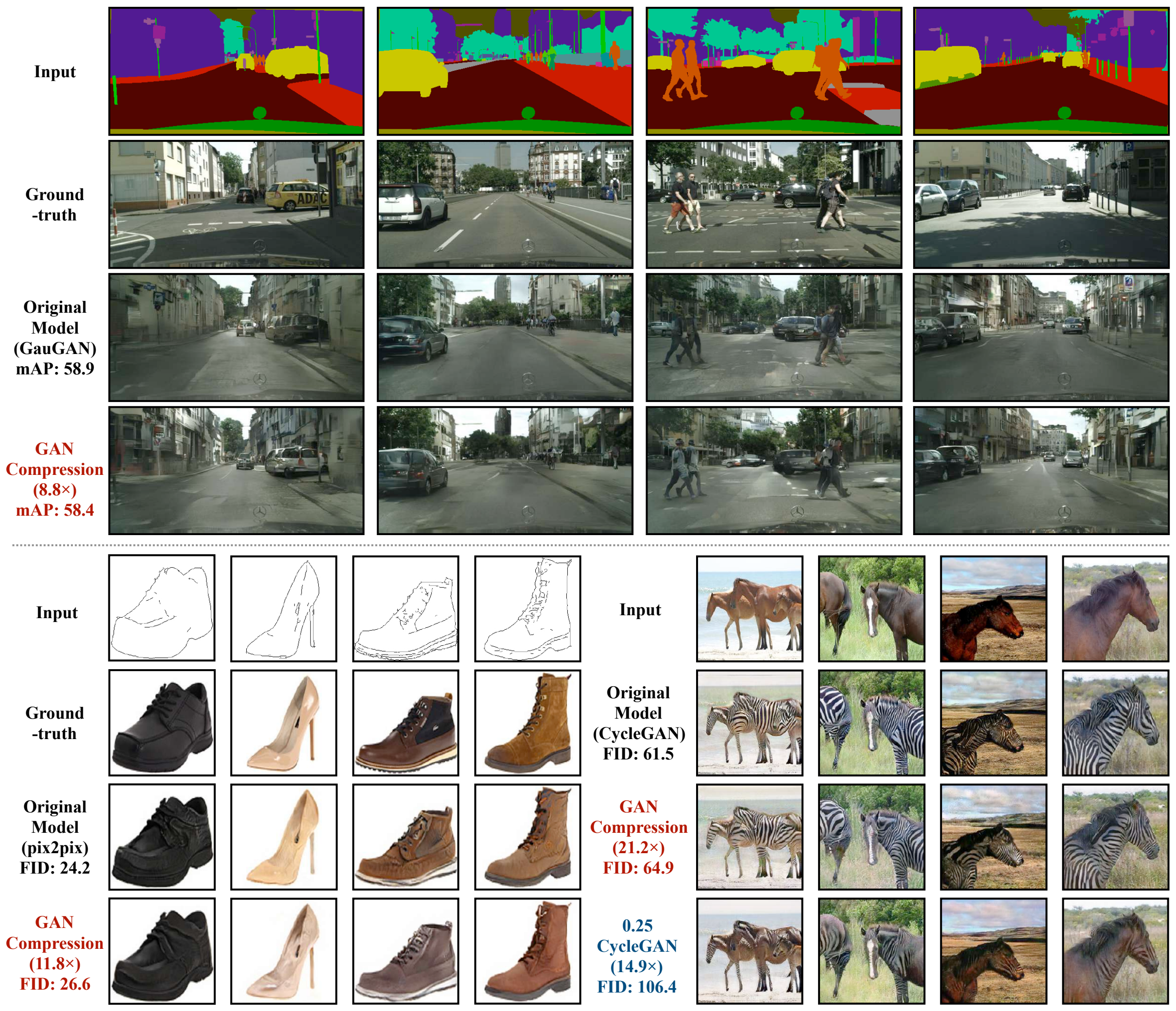

Input
Ground -truth
Original Model (GauGAN) mAP: 58.9
GAN Compression (8.8×) mAP: 58.4
Input
Ground -truth
Original Model (pix2pix) FID: 24.2
GAN Compression (11.8×) FID: 26.6
Input
Original Model (CycleGAN) FID: 61.5
GAN Compression (21.2×) FID: 64.9
0.25 CycleGAN (14.9×) FID: 106.4

Figure 4: Qualitative compression results on Cityscapes, Edges→Shoes and Horse→Zebra. GAN Compression preserves the fidelity while significantly reducing the computation. In contrast, directly training a smaller model (e.g., 0.25 CycleGAN, which linearly scales each layer to 25% channels) yields poor performance.

40 FPS, meeting the demand of interactive applications. The acceleration on GPU is less significant compared to CPU, mainly due to the large degree of parallelism. Nevertheless, we focus on on edge devices where powerful GPUs might not be available.

4.3. Ablation Study

Below we perform several ablation studies regarding our individual system components and design choices.

Advantage of unpaired-to-paired transform. We first analyze the advantage of transforming unpaired conditional GANs into a pseudo paired training setting using the teacher model's output. Figure 7a shows the comparison of performance between the original unpaired training and our pseudo paired training. As our computation budget reduces, the quality of images generated by the unpaired training method degrades dramatically, while our pseudo paired training method remains relatively stable. The unpaired training requires the model to be strong enough to capture the complicated and ambiguous mapping between the source domain and the target domain. Once the mapping is learned, our student model can learn it from the teacher model directly. Additionally, the student model can still learn extra information on the real target images from the inherited discriminator.

Effectiveness of convolution decomposition. We systematically analyze the sensitivity of conditional GANs regarding the convolution decomposition transform. We take the ResNet-based generator from CycleGAN to test its effectiveness. We divide the structure of ResNet generator into three parts according to its network structure: Downsample (3 convolutions), ResBlocks (9 residual blocks), and Upsample (the final two deconvolutions). To validate the sensitivity of each stage, we replace all the conventional convolutions in each stage into separable convolutions [25]. The performance drop is reported in Table. 3. The ResBlock part takes a fair amount of computation cost, so decomposing the con-

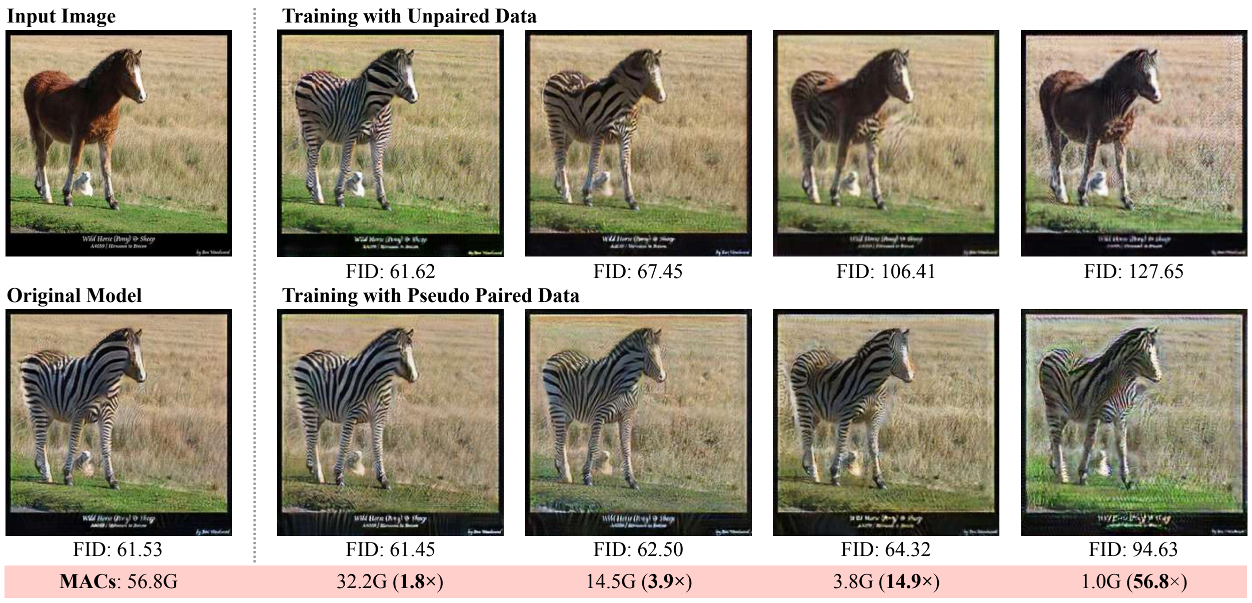
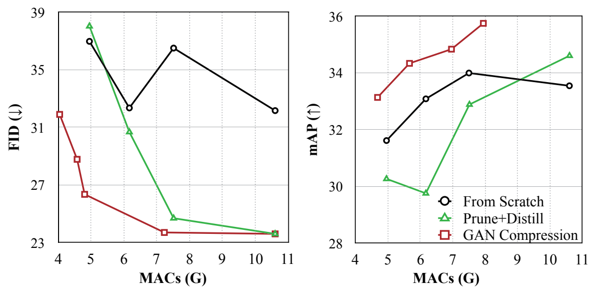


Figure 5: The comparison between training with unpaired data (naive) and training with pseudo paired data (proposed). The latter consistently outperforms the former, especially for small models. The generator's computation can be compressed by 14.9$\times$ without hurting the fidelity using the proposed pseudo pair method. In order to only compare the effectiveness of unpaired *vs*. paired training, both methods do not use automated channel reduction and convolution decomposition.



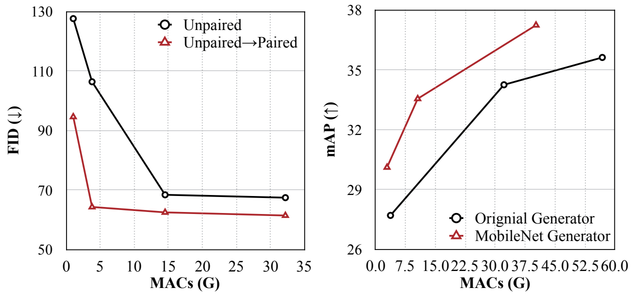
(a) Edge $\rightarrow$ shoes. (b) Cityscapes.

Figure 6: Uniform channel pruning + distillation (without NAS) outperforms training from scratch for larger models, but works poorly when the model is aggressively shrunk. GAN Compression consistently improves the performance *vs*. computation trade-off at various scales.

volutions in the ResBlock can notably reduce computation costs. By testing both the architectures with ngf=64 and ngf=16, the ResBlock-modified architecture shows better computation costs *vs*. performance trade-off. We further explore the computation costs *vs*. performance trade-off of the ResBlock-modified architecture on Cityscapes dataset. Figure. 7b illustrates that such Mobilenet-style architecture is consistently more efficient than the original one, which has already reduced about half of the computation cost.

5. Conclusion

We proposed a general-purpose compression framework for reducing the computational cost and model size of generators in conditional GANs. We first unify unpaired and paired training with our proposed training protocol. Then we use knowledge distillation and neural architecture search to alleviate the training instability and increase the model efficiency. Extensive experiments have shown that our method can compress several conditional GAN models while preserving the visual quality.



(a) Unpaired *vs*. paired training. (b) Normal *vs*. mobile conv.

Figure 7: **(a)** Transforming unpaired training into paired training (using the pseudo pairs generated by the teacher model, without NAS) significantly improves the performance of efficient models. **(b)** Decomposing the convolutions in the original ResNet-based generator into a channel-wise and depth-wise convolutions (MobileNet generator) improves the performance *vs*. computation trade-off.

Acknowledgments We thank NSF Career Award #1943349, MIT-IBM Watson AI Lab, Adobe, Intel, Samsung and AWS machine learning research award for supporting this research. We thank Ning Xu, Zhuang Liu, and Antonio Torralba for helpful comments. We thank NVIDIA for donating the Jetson AGX Xavier that is used in our demo.

References

[1] Kfir Aberman, Rundi Wu, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Learning character-agnostic motion for motion retargeting in 2d. In *SIGGRAPH*, 2019. 1

[2] Angeline Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. *arXiv preprint arXiv:1902.00159*, 2019. 2

[3] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *ICLR*, 2019. 3

[4] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2019. 3

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. 2

[6] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *ICLR*, 2020. 3, 4

[7] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 3, 4

[8] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ICCV*, 2019. 1

[9] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *NeurIPS*, 2017. 2, 4

[10] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4

[11] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018. 2

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 5

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 5

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1, 2, 4

[15] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019. 3, 4, 5

[16] Song Han, Han Cai, Ligeng Zhu, Ji Lin, Kuan Wang, Zhijian Liu, and Yujun Lin. Design automation for efficient deep learning computing. *arXiv preprint arXiv:1904.10616*, 2019. 2

[17] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2015. 2

[18] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015. 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 4, 5

[20] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, 2018. 2, 4

[21] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 2

[22] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017. 5

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2015. 2, 4

[24] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019. 2, 3

[25] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2, 4, 7

[26] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018. 2, 3

[27] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1, 2, 3, 4, 5

[28] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 4, 5

[29] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 2

[30] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017. 2

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[32] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018. 2

[33] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Knowledge distillation from few samples. *arXiv preprint arXiv:1812.01839*, 2018. 2, 4

[34] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *NeurIPS*, 2017. 2

[35] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 3

[36] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018. 3

[37] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019. 3

[38] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017. 2, 3

[39] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 2, 4

[40] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *ICCV*, 2019. 2

[41] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 4

[42] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 4

[43] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, and Xiaoou Tang. Face model compression by distilling knowledge from neurons. In *AAAI*, 2016. 2, 4

[44] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1, 2

[45] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346, 2019. 1, 2, 3, 5

[46] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 4

[47] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 2

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 4, 5

[49] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2

[50] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, pages 5400–5409, 2017. 2

[51] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 2

[52] Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In *ICCV*, 2019. 2, 5, 6

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[54] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[55] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 2

[56] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *CVPR*, 2019. 2

[57] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018. 1

[58] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 2, 3

[59] Shih-En Wei, Jason Saragih, Tomas Simon, Adam W Harley, Stephen Lombardi, Michal Perdoch, Alexander Hypes, Dawei Wang, Hernan Badino, and Yaser Sheikh. Vr facial animation via multiview image translation. *ACM Transactions on Graphics (TOG)*, 38(4):67, 2019. 1

[60] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NeurIPS*, 2016. 2

[61] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 3

[62] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2017. 2

[63] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, pages 4133–4141, 2017. 2, 4

[64] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *CVPR*, 2014. 5

[65] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *CVPR*, 2017. 5

[66] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 4

[67] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *PAMI*, 2018. 2

[68] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017. 2

[69] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 1, 2, 3, 4, 5

[70] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018. 4

[71] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 3