This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Co-Evolutionary Compression for Unpaired Image Translation

Han Shu[1], Yunhe Wang[1], Xu Jia[1], Kai Han[1], Hanting Chen[2], Chunjing Xu[1], Qi Tian[1]*, Chang Xu[3]
[1]Huawei Noah's Ark Lab
[2]Key Lab of Machine Perception (MOE), CMIC, School of EECS, Peking University, China
[3]School of Computer Science, Faculty of Engineering, The University of Sydney, Australia
{han.shu, yunhe.wang, jiaxu1, kai.han, xuchunjing, tian.qi1}@huawei.com,
htchen@pku.edu.cn, c.xu@sydney.edu.au

Abstract

Generative adversarial networks (GANs) have been successfully used for considerable computer vision tasks, especially the image-to-image translation. However, generators in these networks are of complicated architectures with large number of parameters and huge computational complexities. Existing methods are mainly designed for compressing and speeding-up deep neural networks in the classification task, and cannot be directly applied on GANs for image translation, due to their different objectives and training procedures. To this end, we develop a novel co-evolutionary approach for reducing their memory usage and FLOPs simultaneously. In practice, generators for two image domains are encoded as two populations and synergistically optimized for investigating the most important convolution filters iteratively. Fitness of each individual is calculated using the number of parameters, a discriminator-aware regularization, and the cycle consistency. Extensive experiments conducted on benchmark datasets demonstrate the effectiveness of the proposed method for obtaining compact and effective generators.

1. Introduction

Generative adversarial networks (GANs [7]) have achieved impressive results in a wide variety of computer vision tasks, such as super-resolution [18] and image editing [35], which are popular applications on mobile devices. Many of these tasks can be considered as an image-to-image translation problem [14, 30], where an image from one domain is mapped to a corresponding paired image in the other domain. This task is further extended to the unsupervised learning setting by [36, 15, 33], where no paired data are required during training. However, launching such image-to-image translation models on mobile devices requires considerable memory and computation cost, which could challenge the hardware performance and would influence users' experience. For instance, it takes about 43*MB* and 1×10^{10} FLOPs (floating-number operations) to process one image of size 224×224 using a generator network in the CycleGAN [36], which requires more resources than some modern CNNs for large-scale image classification (*e.g.* ResNet [10] and MobileNet [12]).

Recently, a number of algorithms have been proposed for compressing and speeding-up deep neural networks. For instance, Han *et al.* [8] proposed to remove subtle weighs in pre-trained neural networks and rely on some encoding techniques to obtain the compressed models. Wang *et al.* [32] further tackled this problem from the perspective of DCT frequency domain to achieve higher compression ratios. Luo *et al.* [22] pruned filters based on statistics information from the next layer. Hu *et al.* [13] iteratively pruned the network by removing less important neurons according to the magnitude of feature maps. In addition, there are also several methods proposed for learning portable deep neural networks with different techniques, *e.g.* matrix/tensor decomposition [5], quantization and binarization [3, 23, 29], knowledge distillation [11, 26, 27] and efficient convolution blocks design [12, 34].

Although the aforementioned methods have made tremendous progress in reducing redundancy in deep neural networks, most of them are designed for recognition tasks such as image classification or object detection. For recognition tasks, neurons with large activation contribute more to the final classification accuracy. Therefore, neurons with weak activation are often eliminated or approximately represented using low-bit data through low-rank decomposition or clustering without obviously degrading the original performance. In contrast, the generative adversarial networks for image translation tasks are usually composed of a generator and a discriminator, which are updated alternatively with a two-player competition strategy. The training of GANs is thus, more difficult than those of conventional neural networks. It is therefore meaningful to investigate the redundancy in

*corresponding author

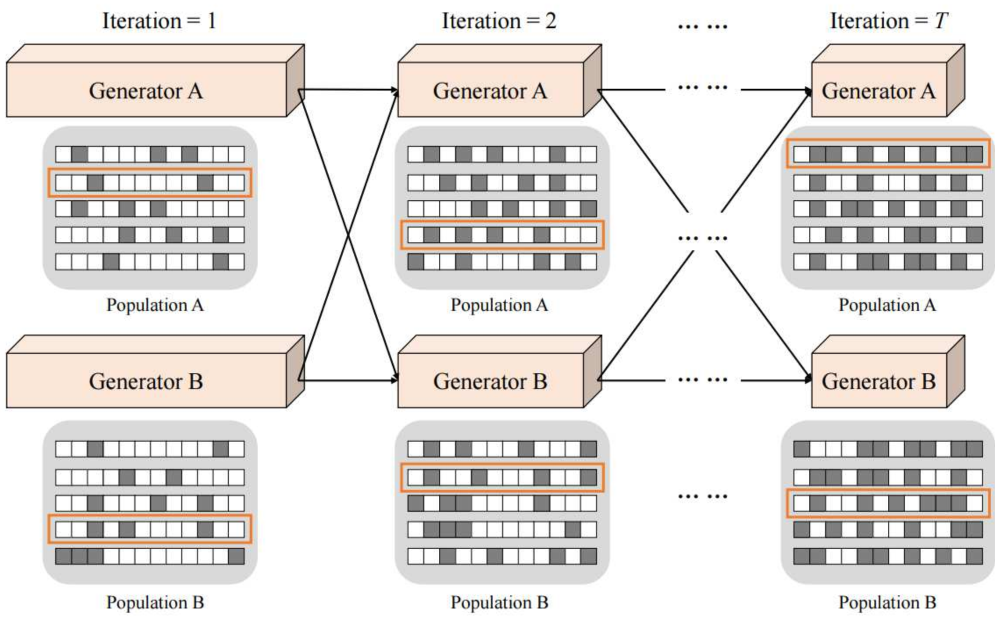

Figure 1. The diagram of the proposed co-evolutionary method for learning efficient generators. Wherein, filters in generators are represented as binary strings, and two populations are maintained for iteratively eliminating redundant convolution filters in each generator. The portable generator will be reconstructed from the best individual (red rectangle) in each population.

GANs and explore an effective approach for learning efficient generators of fewer parameters and calculations.

To this end, we develop a co-evolutionary algorithm to learn efficient architectures for both generators in a synergistic environment as illustrated in Figure 1. Wherein, convolution filters in pre-trained GANs are encoded as a binary string, so that compression and speed-up task can be converted into a binary programming problem. Generators in an image to image translation task could have different redundancies, *e.g*. the generator for converting cityscape images to pixel images would have more redundant than that of the generator for converting pixel images to cityscape images. Two populations are therefore maintained for these two generator networks in the unpaired image translation task, respectively. The fitness of each individual is calculated in terms of the model size and a discriminator-aware loss from GANs. These two populations are updated alternatively by exploiting the best individuals in the previous iteration for obtaining portable architectures of satisfactory performance. Extensive experiments on benchmark datasets show that the proposed co-evolutionary algorithm can effectively compress two generators simultaneously while preserving the quality of transformed images. The compressed generator has less than $1/4$ parameters compared to the original one while maintaining the image translation performance.

The rest of this paper is organized as follows. Section 2 investigates related works on GANs and model compression methods. Section 3 proposes the co-evolutionary approach for removing redundant filters in pre-trained GANs. Section 4 shows the experimental results conducted on benchmark datasets and models, and Section 5 concludes the paper.

2. Related Works

Our goal is to reconstruct compact and efficient generators for image translation based on pretrained generator networks. There are a number of works proposed for image translation with GANs and model compression for compact deep neural networks, which will be reviewed respectively.

2.1. GANs for Image Translation

Generative adversarial networks have achieved impressive performance on the image translation task by adopting a discriminator network to refine the generator. Isola *et al*. [14] introduced generative adversarial networks and L_1 loss to address the issue of paired image translation. Wang *et al*. [30] presented a coarse-to-fine generator and a multi-scale discriminator to generate high-resolution images. Zhu *et al*. [36] implemented the unpaired image-to-image translation by utilizing two opposite domain transfer generators and a well-designed cycle loss. Similarly, Kim *et al*. [15] and Yi *et al*. [33] also adopted the cycle strategy to solve the unpaired image translation problem. Choi *et al*. [2] extended the two-domain translation task to a multi-domain image translation problem.

In fact, those carefully designed generator networks contain tremendous parameters and demand huge computation cost which cannot be efficiently launched on mobile devices, *e.g*. phones and cameras. Thus, we are motivated to explore a compression method to reduce their parameters and computational complexities.

2.2. Model Compression

To learn compact and efficient networks from pretrained models, Denton *et al*. [5] utilized singular value decomposition (SVD) to achieve the low-rank approximation for parameters in fully-connected layers. Chen *et al*. [1] used a hash function and represented weights in the same hash bucket with a single parameter. Han *et al*. [9] removed unimportant parameters in pre-trained neural networks and further [8] utilized quantization, Huffman encoding, and sparse row format for obtaining extremely compact models. Luo *et al*. [22] removed redundant filters and replaced the fully-connected layers by global average pooling (GAP) layers. Vanhouche *et al*. [29] explored deep neural networks with 8-bit integer values to replace original models with 32-bit floating values to achieve the compression and speed-up directly. Courbariaux and Bengio [3] explored neural networks with binary weights and activations. Restgari *et al*. [23] further incorporated binary convolutions into the modern neural architecture to achieve higher performance.

Although these aforementioned methods achieved considerable speed-up and compression ratios on several benchmark models, most of them are developed for recognition tasks such as image classification and object detection, which cannot be straightforwardly applied for generator networks. Actually, GANs consist of a generator and a discriminator, whose outputs are images of high dimension and complex structures. Therefore, we shall develop effective approach for compressing GANs and preserving the visual quality of generated images.

3. Co-Evolution for Efficient Generators

Here we first briefly introduce the CycleGAN [36] for unpaired image-to-image translation, which is the state-of-the-art method for learning the correspondence between two distributions using unpaired data, and then propose a novel co-evolutionary algorithm for simultaneously compressing its two generators.

3.1. Modeling Redundancy in Generators

Formally, given the training dataset from two different domains (*e.g*. Zebra and Horse) $X = \{x_i\}_{i=1}^m$ and $Y = \{y_i\}_{i=1}^n$ with m and n images, respectively. The data distributions of these two domains are denoted as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. The goal of CycleGAN is to learn two mappings simultaneously, *i.e*. $G_1 : X \rightarrow \mathcal{Y}$ and $G_2 : Y \rightarrow X$. For the first mapping G_1 and its discriminator D_1, the corresponding objective function can be mathematically formulated as

$$\begin{aligned}\mathcal{L}_{GAN}(G_1,D_1,X,Y) = \mathbb{E}_{y\sim p_{data}(y)}\left[\log D_1(y)\right] \\ + \mathbb{E}_{x\sim p_{data}(x)}\left[\log(1 - D_1(G(x)))\right],\end{aligned} \tag{1}$$

wherein, the functionality of the generator G_1 is to generate images $G_1(x)$ which looks similar to those from the other domain Y. The discriminator network D_1 is to distinguish between images generated by G_1 and real images in Y. The generator G_1 aims to minimize Eq. 1 while the discriminator tries to maximize it, *i.e*.

$$\min_{G_1}\max_{D_1}\mathcal{L}_{GAN}(G_1, D_1, X, Y), \tag{2}$$

and the entire objective of the CycleGAN is

$$\begin{aligned}\mathcal{L}(G_1, G_2, D_1, D_2) =& \mathcal{L}_{GAN}(G_1, D_1, X, Y) + \\ & \mathcal{L}_{GAN}(G_2,D_2, Y, X) + \lambda\mathcal{L}_{cyc}(G_1, G_2),\end{aligned} \tag{3}$$

where $\mathcal{L}_{cyc}$ is the cycle consistency loss, and λ is the hyper-parameter for seeking the tradeoff between the generation ability and the cycle consistency. It is obvious that, the training of CycleGAN is a more complex procedure than those of recognition tasks, *e.g*. classification [17, 28, 10] and detection [25, 19].

In addition, although GANs perform well on image style transfer, most of generators in these models are well designed with considerable parameters and FLOPs, which are usually unaffordable on mobile devices. In addition, by analyzing Eq. 3, we can find that there are two major differences between the task for compressing image classification or detection models and the task compressing generative networks for image style transfer: 1) discriminator network will be dropped after training the entire generative network, which does not need to be compact; 2) output results of GANs are of high-dimensional, and it is hard to quantitatively evaluate the generated images. We aim to explore effective methods for discovering redundant parameters and compressing original GANs to obtain efficient models.

A straightforward method for reducing complexities of GANs can be directly borrowed from the conventional pruning methods [8, 32] for minimizing the reconstruction error on the output data, which can be formulated as a generator-aware loss function, *i.e*.

$$\mathcal{L}_{GenA} = \frac{1}{m}\sum_{i=1}^{m}||G_1(x_i) - \hat{G}_1(x_i)||_2^2, \tag{4}$$

where $||\cdot||_2$ is the conventional ℓ_2-norm for calculating the difference between generated images using generators before and after compressing, and $\hat{G}_1$ is the compressed generator.

Admittedly, minimizing Eq. 4 can encourage images generated using $\hat{G}_1$ similar to those generated by G_1, but it is not closely related to the style transfer task. In fact, we cannot use an appearance loss to accurately measure the difference between two styles. For instance, a horse with eight or five black and white stripes can be both considered as successful transformations in the image translation task. Therefore, optimizing Eq. 4 would not precisely excavate redundancy in the generator network G_1.

Although the discriminator network D will be abandoned after the training procedure of Eq. 2, it contains important information for distinguishing images from different domains. Thus, we propose to minimize the following discriminator-aware objective function for learning the compressed generator network:

$$\mathcal{L}_{DisA} = \frac{1}{m}\sum_{i=1}^{m} ||D_1(G_1(x_i)) - D_1(\hat{G}_1(x_i))||_2^2, \tag{5}$$

where D_1 is the discriminator in the original network, which captures the style information in the target domain w.r.t. the training dataset $\mathcal{Y}$. Compared with Eq. 4, the above function does not force outputs of original generator and compressed generator to be similar but measures the style discrepancy of generated images through these two generators using the pre-trained discriminator, which is a more appropriate goal for efficient GANs. We will further investigate the difference between Eq. 4 and Eq. 5 on performance of GANs in the experiment part.

In addition, the cycle consistency should also be considered for maintaining the capacity of generators, *i.e.*

$$\mathcal{L}_{cyc} = \frac{1}{m}\sum_{i=1}^{m} ||G_2(\hat{G}_1(x_i)) - x_i||_2^2. \tag{6}$$

Thus, the objective for compressing the first generator G_1 (*e.g.*, horse to zebra) in CycleGAN can be written as

$$\hat{G}_1 = \arg\min_{G_1} \mathcal{N}(G_1) + \gamma\left(\mathcal{L}_{DisA} + \lambda\mathcal{L}_{cyc}\right), \tag{7}$$

where $\mathcal{N}(\cdot)$ counts the number of parameters in neural networks, and γ is the hyper-parameter for balancing the performance of $\hat{G}_1$ and the compression ratio.

Besides the objectives discussed above, there is another important issue should be taken into consideration during the compression procedure of GANs. In general, two generators in the CycleGAN have the same architectures and numbers of convolution filters with the similar capacity for conducting the image-to-image translation task. Only minimizing one generator will make the entire system of CycleGAN unstable, and thus we propose to simultaneously compress these two generators, *i.e.*

$$\begin{aligned}\hat{G}_1, \hat{G}_2 = \arg\min_{G_1, G_2} & \mathcal{N}(G_1) + \mathcal{N}(G_2) \\ & +\gamma\left(\mathcal{L}_{DisA}(G_1, D_1) + \lambda\mathcal{L}_{cyc}(G_1, G_2, X)\right) \\ & +\gamma\left(\mathcal{L}_{DisA}(G_2, D_2) + \lambda\mathcal{L}_{cyc}(G_2, G_1, Y)\right).\end{aligned} \tag{8}$$

which can additionally provide two portable generators at the same time for saving the computing resource.

3.2. Co-Evolutionary Compression

Considering that we cannot accurately estimate the impact of each filter on the final loss according to its output in the given convolutional layer, and functionalities of different filters are interacted, we apply the evolutionary algorithm [24, 31] to encode all convolution filters as binary codes. In addition, there are two variables in Eq. 8, *i.e.* G_1 and G_2, which have their own tasks for learning two different mappings, we thus develop a co-evolutionary approach utilizing Genetic Algorithm (GA [4]) for compressing CycleGAN. Note that, other evolutionary algorithms such as simulated annealing [16] and PSO [6] can also been applied similarly.

Updating G_1: In practice, the filter pruning task will be regarded as a binary programming problem and the generator G_1 will be correspondingly represented as a binary string, *i.e.* individual $\mathbf{p}$. Wherein, each bit is assigned to a convolution filter in the given network, *i.e.*

$$B_l^{(n,:,:,:)} = \begin{cases} 0, & \text{if } \mathbf{p}_l(n) = 0, \\ 1, & \text{otherwise}, \end{cases} \tag{9}$$

where $\mathbf{p}_l$ denotes the state of convolution filters in the l-th layer in G_1. $\mathbf{p}_l(n) = 0$ means discarding the n-th filter in the l-th convolutional layer, otherwise retaining. The number of filters is about tens of thousands in conventional neural networks [10, 36], and the length of $\mathbf{p}$ for l convolutional layers is tolerable. Since convolution filters in different layers are of various sizes, which has different memory usage and computational complexities, we utilize the following function to reformulate $\mathcal{N}(\cdot)$ in Eq. 8:

$$\mathcal{N}(\mathbf{p}) = \frac{\sum_{l=1}^{L}\left(\|\mathbf{p}_{l-1}\|_1 \cdot \|\mathbf{p}_l\|_1 \cdot H_l \cdot W_l\right)}{\sum_{l=1}^{L}(N_l \cdot C_l \cdot H_l \cdot W_l)}, \tag{10}$$

which assigns convolution filters of more weights with higher importance. Wherein, $\|\mathbf{p}_{l-1}\|_1$ is the number of filters in the $l-1$-th layer, *i.e.* the channel number in the l-th layer, N_l, C_l, H_l and W_l are the number of filters, the number of channels and height and width of filters in the l-th convolutional layer in G_1 respectively. Besides memory usage, Eq. 10 also takes FLOPs into consideration since a convolution filter with more weights, *i.e.* $C_l \times H_l \times W_l$ usually involves more multiplications in GANs.

Then, the fitness of an individual for compressing the generator G_1 is defined as

$$\mathcal{F}(\mathbf{p}) = \Big[\mathcal{N}(\mathbf{p}) + \gamma\Big(\mathcal{L}_{DisA}(\hat{G}_1, D_1) + \lambda\mathcal{L}_{cyc}(\hat{G}_1, G_2, X)\Big)\Big]^{-1}, \tag{11}$$

where $\hat{G}_1$ is the compressed generator corresponding to the given individual $\mathbf{p}$.

After defining the calculation of fitness, GA is adopted to find the fittest individual through several evolutions. For each individual, the corresponding compressed network is

Algorithm 1 Co-Evolutionary compression for GANs.

Require: Training set $X = \{x_i\}_{i=1}^m$ and $Y = \{y_i\}_{i=1}^n$. The pre-trained GAN with two generators and discriminators, G_1, G_2, D_1, and D_2, parameters: K, T, λ, γ, and learning rates, *etc*.

Initialize two populations P_0 and Q_0 w.r.t. G_1 and G_2 with K individuals, respectively;
Select the best individuals $\hat{\mathbf{p}}^{(0)}$ and $\hat{\mathbf{q}}^{(0)}$;
for $t = 1$ to T **do**
  Calculate the fitness of each individual in P_t:
  $\mathcal{F}(\mathbf{p}^{(t)}) \leftarrow [\mathcal{N}(\mathbf{p}^{(t)}) + \gamma(\mathcal{L}_{DisA}(\mathbf{p}^{(t)}, D_1, X) + \lambda \mathcal{L}_{cyc}(\mathbf{p}^{(t)}, \hat{\mathbf{q}}^{(t-1)}, X))]^{-1}$;
  Calculate the fitness of each individual in Q_t:
  $\mathcal{F}(\mathbf{q}^{(t)}) \leftarrow [\mathcal{N}(\mathbf{q}^{(t)}) + \gamma(\mathcal{L}_{DisA}(\mathbf{q}^{(t)}, D_2, Y) + \lambda \mathcal{L}_{cyc}(\mathbf{q}^{(t)}, \hat{\mathbf{p}}^{(t-1)}, Y))]^{-1}$;
  Obtain selecting probabilities (Eq. 12);
  Preserve the best individuals:
  $P_t^{(1)} \leftarrow \hat{\mathbf{p}}^{(t-1)}$, $Q_t^{(1)} \leftarrow \hat{\mathbf{q}}^{(t-1)}$;
  for $k = 2$ to K **do**
    Generate a random value $s \sim [0, 1]$;
    Conduct selection, crossover, and mutation for generating new individuals according to s;
  end for
end for
Update fitnesses of individuals in P_t and Q_t;
Establish two generator networks $\hat{G}_1$ and $\hat{G}_2$ by exploiting to the best individual $\hat{\mathbf{p}}^{(T)}$ and $\hat{\mathbf{q}}^{(T)}$, respectively;

Ensure: Portable generator $\hat{G}_1$ and $\hat{G}_2$ after fine-tuning using the entire training set.

fine-tuned on a subset of the training data (*e.g*. 10% training images randomly sampled) and take the fitness on validation set as evaluation metric. Then a probability is assigned to each individual by comparing its fitness among the individuals in the current population:

$$\Pr(\mathbf{p}^j) = \mathcal{F}(\mathbf{p}^j) \Bigg/ \sum_{k=1}^{K} \mathcal{F}(\mathbf{p}^k) , \tag{12}$$

where $\mathbf{p}^j$ is the j-th individual in the population and K is the number of individuals in the population. The population in each iteration are regarded as parents, and selected according to Eq. 12. The selected parents breed another population as offspring using the following operations: **selection**, **crossover**, and **mutation** [4, 31].

Updating G_2: Although architectures of two generators in the CycleGAN are usually symmetrical with the same amount of convolution filters, redundancy in G_1 and G_2 can be significantly different. For instance, learning the mapping from semantic maps to streetviews is harder than that of from streetviews to semantic maps. Therefore, we utilize another population to optimize the other generator in CycleGAN, *i.e*. G_2.

Similarly, we encode all convolution filters in G_2 to formulate a population with K individuals, *i.e*. $\mathbf{q}^1, ..., \mathbf{q}^K$, and the corresponding fitness can be defined as

$$\begin{aligned} \mathcal{F}(\mathbf{q}) = \Big[\mathcal{N}(\mathbf{q}) + \gamma \Big(\mathcal{L}_{DisA}(\hat{G}_2, D_2) \\ + \lambda \mathcal{L}_{cyc}(\hat{G}_2, G_1, Y)\Big)\Big]^{-1}, \end{aligned} \tag{13}$$

which can be also optimized during the evolutionary procedure.

Moreover, it can be found in Eq. 11 and Eq. 13 that, the calculation of cycle consistency of each generator involves the other generator in the CycleGAN. Thus, two populations are alternatively updated to simultaneously optimize G_1 and G_2. In specific, for the t-th iteration, we first obtain the best individual $\mathbf{p}^{(t)}$ of G_1 utilizing the best individual of G_2 in the previous iteration $\mathbf{q}^{(t-1)}$, and then utilize it to calculate the fitness of G_2. In addition, the best individual preserving strategy is also adopted to increase the robustness of the evolutionary algorithm. The detailed procedure for learning portable GANs using the proposed method is summaries in Algorithm 1.

4. Experiments

In this section, we qualitatively and quantitatively evaluate the proposed discriminator aware compression method on three benchmark unpaired image translation datasets, *i.e*. horse2zebra, summer2winter, and cityscapes. The architecture of CycleGAN is directly borrowed from their original paper [36]. Each generator in the CycleGAN is sequentially composed of one 7×7 stride-1 convolutional layer, two 3×3 stride-2 convolutional layers, nine residual blocks [10], two 3×3 stride-2 transpose convolutional layers and one 7×7 stride-1 convolutional layer. In addition, each discriminator consists of 5 convolutional layers and one FCN [21] classification layer. We use the default setting in [36] to pretrain and finetune the CycleGAN for having a fair comparison.

Impact of Parameters. Our goal is to learn efficient generative network for unpaired image-to-image style transfer. As discussed in the Algorithm 1, the objective function Eq. 8 for compressing GANs will be converted as the fitness calculation in the framework of the proposed co-evolutionary approach. λ is the parameter for weighting the cycle consistency term, which is set as 10 according to the original CycleGAN [36]. In addition, the identity loss for maintaining the information of each domain is also applied along with the cycle consistency. The number of individuals K is set as 32, and the maximum iteration number T is equal to 100, which refer to those in [31].

Then, we further investigate the trade-off between the compression ratio and performance of compressed generative networks according to different hyper-parameter γ. It

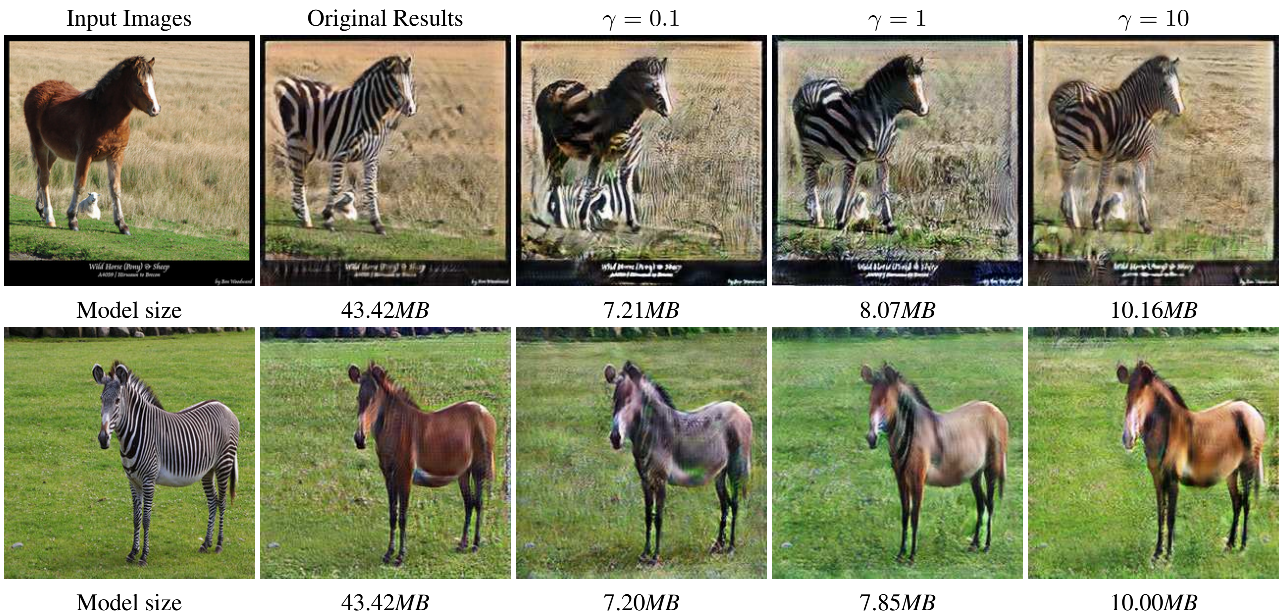

Figure 2. Images generated using the generator compressed by exploiting the proposed method with different hyper-parameters. The top line and the bottom line show the results from horse to zebra, and results from zebra to horse, respectively. Two generators are compressed from an entire CycleGAN. Model sizes of different generators are provided.

can be found in Figure 3.2 that, a larger γ brings a lower compression ratio, *i.e*. the model size is much smaller than that of the original model. However, the visual quality of the resulting images will be better for a larger γ.

As a result, we set $\gamma = 10$ to obtain the compressed model with an acceptable generative ability, images generated using the compressed network are similar to those using the original model. The compression ratios of two generators are $4.27\times$ and $4.34\times$, respectively. In addition, compression ratios of two generators are similar since the difficulty for transferring horses to zebras is also similar to that of transferring zebras to horses.

Ablation Study. After investigating the trade-off between the generative ability and model size of GANs, we further conduct extensive ablation experiments to evaluate the functionality of different components in the proposed scheme.

A co-evolutionary approach for iteratively compressing two generators in the CycleGAN was developed in Section 3, which involves two populations for obtaining generative models with higher performance. Thus, we first compare the results using the evolutionary algorithm to compress two generators separately and those from the proposed co-evolutionary algorithm, as shown in Figure 3(b) and Figure 3(d), respectively.

In order to have a fair comparison, we tune the hyper-parameter to obtain compressed network with the similar model size, *e.g*. the generator using the proposed method is 10.16*MB* on the horse2zebra task. It is clear that, the proposed co-evolutionary approach obtained images with higher visual quality, *e.g*. clear zebra pattern and more white mountains, since the proposed method can simultaneously investigate the redundancy in both two generators. In addition, we can obtain two efficient and effective generators at the same time, which is much more flexible than the scheme for compressing them separately.

We then compared the performance of the proposed two loss functions for evaluating the capacity of compressed GANs, *i.e*. the generator-aware loss and the discriminator-aware loss. The results of compressed models under the generator-aware constraint are shown in Figure 3(c). It is obvious that, the generated images using the generator-aware loss $\mathcal{L}_{GenA}$ are worse than those using the discriminator-aware loss, since the style information cannot be easily captured by the reconstruction error. For example, the difference between horses and zebras are only exist on the body of horses, the overall difference between input images and Figure 3(c) are not significant.

Comparison with Conventional Pruning Method. In contrast to conventional method for pruning redundant convolution filters in pre-trained deep neural networks, the proposed method introduces a discriminator aware loss, *i.e*. Eq. 5 to recognize useless filters for conducting the image style transfer task. Therefore, we then compare the proposed method with the state-of-the-art filter pruning method, namely, ThiNet [22], which minimizes the reconstruction error of output features. Similarly, we also tuned the hyper-parameters in ThiNet to ensure that the model size (10.88*MB*) of resulting generator of ThiNet is similar to that of using the proposed method.

It can be found in Figure 3(a), images generated through a generator compressed by ThiNet for a similar amount of parameters cannot capture the style information in the target domain, *e.g*. the generated zebra images are fundamentally

Figure 3. The generated images on horse2zebra and summer2winter datasets using different methods and strategies. The first two columns illustrate the input images and images generated by the original CycleGAN. Images in (a) are generated by by compressed generators using the conventional ThiNet for filter pruning, while (b) by evolutionary approach for compressing two generators separately using $\mathcal{L}_{DisA}$, (c) by co-evolutionary method using $\mathcal{L}_{GenA}$, (d) by co-evolutionary method using $\mathcal{L}_{DisA}$.

different to those of original model and compressed model using the proposed method, as shown in Figure 3(d). In fact, the conventional filter pruning method has the similar assumption to that of the generator-aware loss, which obtained similar but worse results as those in Figure 3(c), which is not suitable to conduct the compression task for the unpaired image translation. In addition, we also compare the proposed method with other filter pruning methods, namely, network trimming [13] and slimming [20], which obtained similar results to those of ThiNet and can be found in the supplementary materials.

Filter Visualization. Since the evolutionary algorithm can globally discover the most beneficial filters for given task, it is necessary to see what filters are recognized as redundant and what filters are essential for generator. Thus, we visualize the first several filters in the first convolutional layer of CycleGAN on the horse2zebra dataset as shown in Figure 4. Interestingly, the discarded filters by our method are not only with small norms but may also have big values, which is significantly different from the results of the conventional filter pruning method, *i.e*. ThiNet [22]. Actually, the weights in filters for extracting color and texture information can be very small.

It can be found in Figure 4, the proposed method retains filters with more distinct structures, which are beneficial for maintaining an acceptable performance of the generator network. Furthermore, filters after fine-tuning do not have significant changes, which demonstrates importance and functionality of these convolution filters for conducting the subsequent image-to-image translation task.

Table 1. Statistics of compressed generators.

Task	Memory	r_c	FLOPs	r_s
horse2zebra	10.16MB	4.27×	$13,448M$	4.23×
zebra2horse	10.00MB	4.34×	$13,060M$	4.35×
summer2winter	7.98MB	5.44×	$11,064M$	5.14×
winter2summer	7.61MB	5.70×	$10,994M$	5.17×
cityscapes-A2B	8.98MB	4.84×	$12,977M$	4.38×
cityscapes-B2A	12.26MB	3.54×	$16,445M$	3.46×

Detailed Compression results. Moreover, detailed results of the six generators trained on three datasets, *i.e*. horse2zebra, summer2winter, and cityscapes, are illustrated in Table 1, r_c and r_s are compression rates for model size and FLOPs respectively. It is obvious that, the proposed co-evolutionary method can effectively remove redundant filters in pre-trained GANs and obtain efficient generators.

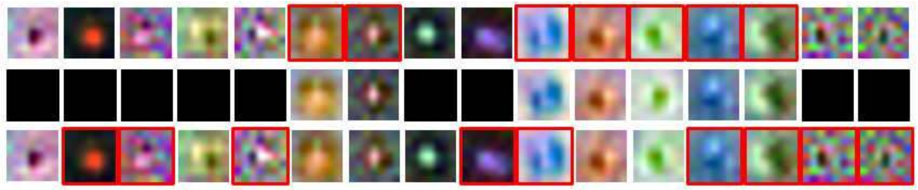

Figure 4. Filter visualization results. From top to bottom: the original filters with red rectangles selecting the remained filters by the proposed method, filters after fine-tuning, and the original filters with red rectangles selecting the filters remained by ThiNet.

Table 2. FCN scores of different generators calculated on the cityscapes dataset.

Method	Memory	Mean Pixel Acc	Mean Class Acc.	Mean class IoU
Original [36]	43.42*MB*	0.538	0.172	0.121
ThiNet [22]	10.88*MB*	0.218	0.089	0.054
Ours	12.26*MB*	0.542	0.212	0.131

Additionally, we can obtain two efficient generators from CycleGAN [36] for conducting the unpaired image-to-image translation task by simultaneously explore their redundancy. Furthermore, there are some interesting phenomenons in Table 1, *i.e.* the generator for more difficult transformation task tend to have less redundancy. For instance, since the task for transferring semantic map to streetview is more difficult than that of transferring streetview to semantic map. The model size (12.26*MB*) of the second compressed generator (*i.e.* cityscapes-B2A) is much larger than that (8.98*MB*) of the first generator (*i.e.* cityscapes-A2B) for transferring streetview to semantic map, which demonstrates the superiority of the proposed co-evolutionary approach for compressing GANs and provides some guidance for designing GANs for various tasks. In addition, detailed statics of two generators in the compressed CycleGAN and more visualization results on three benchmark datasets generated of compressed models using the proposed method can be found in the supplementary materials.

Runtime. The proposed method directly removes redundant filters and produces efficient GANs. Thus, the compressed model does not require other additional support (*e.g.* sparse matrices and Huffman encoding [8]) for realizing the network speed-up. We then compared runtimes for processing images using original and compressed models. In practice, the averaged runtime of the original model for processing one image is about $2,260$ms using an Intel Xeon E5-2690 CPU. In contrast, the runtime of the compressed model with a 10.16*MB* model size (*i.e.* the first line in Table 1) is about 730ms, which achieved an about $3.1\times$ speed-up, which is lower than that of the theoretical speed-up ratio ($4.23\times$) due to the costs of incurred by data transmission, ReLU, *etc*. The demo code for verifying the proposed method can be found in our supplementary materials.

Quantitative Evaluation. Besides the above experiments, we also conduct the quantitative evaluation of the proposed method. In order to evaluate the quality of compressed generators the "FCN-score" [14] is utilized on images generated from semantic maps to cityscape images. In practice, a pre-trained FCN-8s network [21] on the cityscapes dataset is exploited for conducting the semantic segmentation experiments and detailed results are shown in Table 2. Measurements for the segmentation experiment are per-pixel accuracy, per-class accuracy and mean class IOU. It is obvious that, the proposed method obtained better results compared with the conventional ThiNet [22] for pruning convolution filters, which are slightly higher results than those of using the original generator, since we can effectively remove useless filters to establish generative models perform well on discriminator networks. In addition, the segmentation results are shown in our supplementary materials.

On datasets of horse2zebra and summer2winter, Fréchet Inception Distance (FID) is adopted to evaluate the results of the proposed method as shown in Table 3. The results of proposed method are close to the original CycleGAN[36], obviously better than the weight based pruning method.

Table 3. Comparision of FID scores.

FID	Original[36]	ThiNet[22]	Ours
horse2zebra	74.04	189.28	96.15
zebra2horse	148.81	184.88	157.90
summer2winter	79.12	81.06	78.58
winter2summer	73.31	80.17	79.16

5. Conclusion

This paper studies the model compression and speed-up problem of generative networks for unpaired image-to-image style translation. A novel co-evolutionary scheme is developed for simultaneously pruning redundant filters in both two generators. Two portable generator networks will be effectively obtained during the procedure of the genetic algorithm. Experiments conducted on benchmark datasets and generative models demonstrate that the proposed co-evolutionary compression algorithm can fully excavate redundancy in GANs and achieve considerable compression and speed-up ratios. In addition, images generated using the compressed generator also maintain the style information with high visual quality, which can be directly applied on any off-the-shelf platforms.

Acknowledgments: We thank Dr. Gang Niu for the insightful discussion. Chang Xu was supported by the Australian Research Council under Project DE180101438.

References

[1] Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449*, 2015. 3

[2] 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*, pages 8789–8797, 2018. 2

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 1, 3

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. 4, 5

[5] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 1, 3

[6] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43, 1995. 4

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 1

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 1, 3, 8

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3, 4, 5

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[12] 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

[13] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016. 1, 7

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. 1, 2, 8

[15] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, pages 1857–1865. JMLR. org, 2017. 1, 2

[16] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983. 4

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3

[18] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017. 1

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 3

[20] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2755–2763, 2017. 7

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 5, 8

[22] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5058–5066, 2017. 1, 3, 6, 7, 8

[23] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016. 1, 3

[24] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *ICML*, pages 2902–2911. JMLR. org, 2017. 4

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[26] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 1

[27] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. *arXiv preprint arXiv:1811.02796*, 2018. 1

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 3

[29] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2011. 1, 3

[30] 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*, pages 8798–8807, 2018. 1, 2

[31] Yunhe Wang, Chang Xu, Jiayan Qiu, Chao Xu, and Dacheng Tao. Towards evolutional compression. In *SIGKDD*, 2018. 4, 5

[32] Yunhe Wang, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. Cnnpack: Packing convolutional neural networks in the frequency domain. In *NIPS*, 2016. 1, 3

[33] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, pages 2868–2876. IEEE, 2017. 1, 2

[34] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 1

[35] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, pages 597–613, 2016. 1

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017. 1, 2, 3, 4, 5, 8