

CoroNetGAN: Controlled Pruning of GANs via Hypernetworks

Aman Kumar¹, Khushboo Anand¹, Shubham Mandloi¹, Ashutosh Mishra¹, Avinash Thakur¹, Neeraj Kasera¹, and Prathosh A P²

¹OPPO Mobiles R & D Center, Hyderabad, India *

²Indian Institute of Science, Bangalore[†]

Abstract

Generative Adversarial Networks (GANs) have proven to exhibit remarkable performance and are widely used across many generative computer vision applications. However, the unprecedented demand for the deployment of GANs on resource-constrained edge devices still poses a challenge due to huge number of parameters involved in the generation process. This has led to focused attention on the area of compressing GANs. Most of the existing works use knowledge distillation with the overhead of teacher dependency. Moreover, there is no ability to control the degree of compression in these methods. Hence, we propose CoroNetGAN for compressing GAN using the combined strength of differentiable pruning method via hypernetworks. The proposed method provides the advantage of performing controllable compression while training along with reducing training time by a substantial factor. Experiments have been done on various conditional GAN architectures (Pix2Pix and CycleGAN) to signify the effectiveness of our approach on multiple benchmark datasets such as Edges \rightarrow Shoes, Horse \leftrightarrow Zebra and Summer \rightarrow Winter. The results obtained illustrate that our approach succeeds to outperform the baselines on Zebra \rightarrow Horse and Summer \rightarrow Winter achieving the best FID score of 32.3 and 72.3 respectively, yielding high-fidelity images across all the datasets. Additionally, our approach also outperforms the state-of-the-art methods in achieving better inference time on various smart-phone chipsets and data-types making it a feasible solution for deployment on edge devices.

1. Introduction

Computer vision applications such as image-to-image translation, image synthesis, image generation, super res-

olution etc. have seen tremendous progress yielding high-fidelity images with the advent of GANs[1]. The development of image-based GAN applications have in-turn accelerated the demand for deployment of such models on edge devices for the usage of the end consumers. However, the complexity of training such parameter heavy models to generate visually pleasing images result in high computational and memory overhead which acts as a bottleneck in deployment of GANs on mobile devices. For instance, the popular CycleGAN requires over 56.8G MACs (Multiply-Accumulate Operations) for generating a single image of resolution 256×256 pixels. On the other hand, Pix2Pix requires 18.6G MACs which is 4X compared to traditional Res-Net50 [2] architecture. This huge number of operations is not desirable for the deployment on edge devices. Hence, there is a need for compressing these networks by removing the redundant parameters and reducing the memory and computational consumption.

Discriminative approaches such as image classification, object detection and semantic segmentation have been at the receiving end of undivided focus since these networks have surpassed human imagination but still, for the learning to saturate, these networks take huge amount of training time. For instance, the popular image classification model, Alexnet [3] has 60 million parameters and requires about 240 MB of memory while VGG16 [4] has 130 million parameters and has takes around 500 MB of memory. The research community has given unmitigated attention on the application of model compression techniques to accelerate deployment of image classification and object detection networks using techniques like weight quantization [5, 6], pruning [7, 8] and knowledge distillation [9, 10]. However, these methods are not directly applicable to generative models such as GANs. A lot of the recently proposed methods have tried to compress generative adversarial networks using the combined techniques of knowledge distillation [11, 12] and channel pruning[13, 14]. However, these approaches don't allow controllable compression to happen neither using a single technique nor through the combina-

*E-mail: {aman.kumar1, khushboo.anand, shubham.mandloi, ashutosh.mishra1, avinash.thakur, neeraj.kasera}@oppo.com

[†]E-mail: {prathoshap@gmail.com}

tion of multiple techniques.

To address the above-mentioned issues, we propose a novel method for compressing the GAN using differentiable pruning method using the concept of hypernetwork. The compression is performed during the training regime. The proposed hypernetwork takes latent vector as an input and dynamically produces weights of a given layer of the generator network. This input latent vector decides the pruning rate for different layers in the network. Sparsification of latent vector is achieved via proximal gradient. Post sparsification, the latent vectors are passed through the hypernetwork that in turn generates the weight of the generator network. Since the latent vector and the weights of the generator network are covariant with each other, the sparsification of latent vectors leads to the pruning of the weights of the concerned network. The proposed method also helps in reducing the training time and inference time as compared to that of conventional GAN training method. Through the experiments on different conditional generative models on various datasets, the potential of the proposed method is revealed. The main contributions of the paper can be summarized as follows:

1. We propose CoroNetGAN, an approach based on differentiable pruning via hypernetworks for GAN compression. To the best of our knowledge, this is the first work that achieves model compression using controllable pruning via hypernetwork for conditional GANs. Our proposed approach compresses the GAN network in a controlled way by providing the compression rate as an input to the algorithm.
2. Compression is achieved simultaneously alongside training unlike the distillation based methods that involve teacher dependency [15]. CoroNetGAN outperforms state-of-the-art compression technique [15] on training time on all the datasets validating the effectiveness of our technique both on training latency and visual appearance of the generated images. This will be of great advantage in reducing the training time while maintaining the accuracy when training GANs on bigger datasets containing billion of images.
3. Our proposed approach, CoroNetGAN outperforms state-of-the-art conditional GAN compression methods on widely used Zebra \rightarrow Horse and Summer \rightarrow Winter datasets. CoroNetGAN obtains reasonable qualitative and quantitative results on other datasets. CoroNetGAN also outperforms state-of-the-art compression techniques [15] on inference time.

2. Related Work

2.1. Generative Adversarial Networks

GANs [1] have proven to generate realistic results on a variety of tasks. For instance, Isola et al. [16] propose Pix2Pix for paired image-to-image translation trained via the combination of adversarial loss and pixel-wise regression loss in order to ensure the visual quality of generated images. Later, [17] is proposed that helps to increase the resolution of translated images with multi-scale neural networks and edge maps. GANs have also been proposed to perform image deblurring [18], style transfer [19, 20], image super resolution [21] along with text-to-image generation [22]. Zhu et al. [23] propose CycleGAN for unpaired image-to-image translation. The algorithm trains generators on different domains of data through a weakly supervised setting using cycle consistency loss. The final objective is to convert the data from one domain to other without using any label information.

2.2. GAN Compression

The tremendous resource consumption by GANs has garnered recent attention towards GAN compression. Wang et al. [24] proposes a novel quantization method and multi-precision quantization algorithm considering different sensitivities of discriminator and generator. Aguinardo et al. [11] introduces the idea of knowledge distillation in GANs between large over-parameterized network and small few parameter networks optimized using joint and mean squared error loss functions. However, the only focus here is to compress the generator keeping the discriminator intact. Most usage of GANs in mobile devices is based on the application of image-to-image translation task. [12] distills the student discriminator to assist training of the student generator and also focused on image translation problem using Pix2Pix framework. Chang et al. in [25] focuses to mimic the functionality of BigGAN with a smaller compressed network and fewer parameters. Different devices with varied computing power require generators of different sizes. In order to accommodate this trade-off, SlimmableGAN [26] proposes flexible switching between the multi-width configurations. Further, Ren et al. [15] overcomes the complex multi-stage compression process and proposes a single-stage GAN online distillation strategy to obtain the compressed model. However, these approaches use images from the teacher directly to distill knowledge. Zhang et al. [27] proposes the idea of investigating GAN compression from frequency perspective and introduces the idea of wavelet analysis. They decompose the image into frequency bands and perform distillation only on bands with higher frequency unlike naive methods that do not prioritize the high frequency. [28] aims to find crucial regions in the image using attention module. Considering the attention value

important to the region, features are distilled from teacher to student. Recent works such as [29] introduce an Inception-based Residual block replacing the original residual blocks in CycleGAN and search for student generator from teacher generator via pruning followed by Similarity based Knowledge Distillation. Further, approaches integrating various compression techniques are also proposed. [13] combines model distillation, channel pruning and quantization and generate a unified optimization form which achieves superior trade-off compared with standalone compression techniques. Liu et al.[14] combines the idea of channel pruning and knowledge distillation and mainly expands the focus on accelerating unconditional GANs.

2.3. HyperNetworks

Hypernetworks are a group of smaller networks that generates the weights for a larger network. These smaller neural networks have been used historically for vision[30], functional representation[31] and bayesian inference tasks[32].

Albeit the word hypernetwork has been coined recently, the concept of using dynamic parameter generation has been used by researchers for a long time [33]. Von der Malsberg et al.[34] indicates a possibility of dynamic modelling between a slow classical weight and a dynamic decaying connection. The technique to model short term memory by computing weight changes of another network was initiated by Schmidhuber et al.[35]. Parameter prediction through co-relation between different parameters of the neural network was extensively studied in [36]. A weight matrix is produced using a learnable lower dimensional matrix using a linear operation.[37] uses weight matrices as a factored representation and feed forward one-shot learners reducing the dimensionality of the hypernetwork.[38] proposes an approach to calculate the parameters for image transformation using a weight generating network. [39] proposes an approach for generating weights for visual question answering task. The parameter prediction network takes input the questions post which the network predicts weights of the main network. In addition, they also use hashing of parameters to reduce the size of the final matrix of parameters. The concept of dynamic filters has been used for image super-resolution [30]. These filters are computed based on input using a similar concept to hypernetwork.

3. Methodology

GAN consist of a generator and a discriminator network employed in a min-max game. The proposed method allows compression of the generator network while training. Compression is achieved using differentiable meta pruning which is based on the idea of hypernetwork. Hypernetwork is responsible for generating the weights of the generator network for each of its layer. The input to the hypernetwork

is a latent vector and the output is a weight matrix of the generator network.

During the forward pass, latent vector is given as an input to hypernetwork to generate the weights of the generator. During back-propagation, the gradient flows in the hypernetwork instead of the main network. It is designed in a way such that its output is covariant with the input latent vector. Proximal Gradient helps in pruning of output channels of the generator network by eliminating the redundant parameters automatically.

3.1. HyperNetwork Design

The hypernetwork consists of three layers. The latent layer takes as input the latent vectors and computes a latent matrix. The embedding layer projects the elements of the latent matrix to an embedding space. The final layer converts the embedded vectors to the final output. The design is taken from [40]. As an example, consider the generator to be an L-layer convolutional neural network. Each layer of the network has its own corresponding latent vector that is responsible for generating the weights of the corresponding layer. The size of the latent vector is equal to the number of output channels in that layer. For instance, consider an l -th convolutional layer having $n * c * w * h$ number of parameters, where n and c are the output and input channels and $w * h$ is the size of kernel respectively. Suppose that the latent vector corresponding to that particular $l - th$ layer is $v^l \in R^c$. Therefore, the previous layer has latent vector $v^{l-1} \in R^n$. The hypernetwork takes latent vector of current layer (v^l) and its previous layer (v^{l-1}) as input and will output the weights matrix of the l -th layer of the generator network. Initially, the first layer of the hypernetwork computes a latent matrix using the two latent vectors:

$$\mathbf{V}^l = \mathbf{v}^l \cdot \mathbf{v}^{l-1T} + \mathbf{B}_0 \quad (1)$$

where,

$$\mathbf{V}^l, \mathbf{B}_0 \in R^{n*c}$$

Here, $[T]$ denotes transpose of the matrix while $[\cdot]$ denotes matrix multiplication.

Subsequently, the second layer of the hypernetwork projects every element of the latent matrix to a m -dimensional embedding space as follows:

$$\mathbf{S}_{ij}^l = \mathbf{V}_{ij}^l \mathbf{w}_1^l + \mathbf{b}_l^1 \quad i = 1..n, j = 1..c \quad (2)$$

where,

$$\mathbf{S}_{ij}^l, \mathbf{w}_1^l, \mathbf{b}_l^1 \in R^m$$

Here, we are considering \mathbf{w}_1^l and \mathbf{b}_l^1 as different for different elements of the matrix. The subscript (i, j) has not been used for easier interpretation of the above mentioned equations. The vectors \mathbf{w}_1^l , \mathbf{b}_l^1 and \mathbf{S}_{ij}^l for all the elements

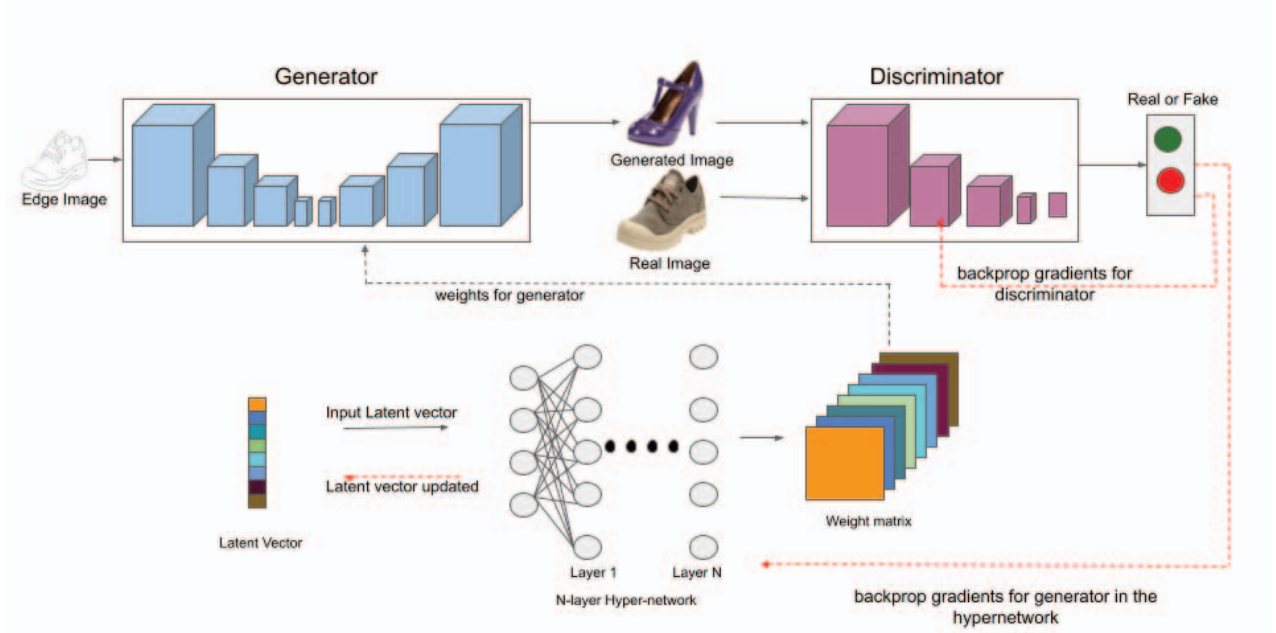


Figure 1. Illustration of the proposed algorithm designed for compressing GAN's using controllable differentiable pruning. A latent vector is attached to each of the convolution layer of the generator. The latent vector generates the weights for the generator via hypernetwork. Sparsification of the latent vector leads to pruning of the corresponding weights of the generator network. The proposed design allows the latent vector and its corresponding weight matrix to be covariant with each other. The generator generates visual results using the computed weight matrix through the hypernetwork (*Best viewed when zoomed*).

of the matrix together forms a 3D tensor, i.e., \mathbf{W}_1^l , \mathbf{B}_1^l and $\mathbf{S}_1^l \in R^{n*c*m}$.

After the second step, the final layer of the hypernetwork is responsible for converting the embedding vectors to the output(\mathbf{F}_{ij}^l) which can be used as weight matrix of the Generator network. This is done by multiplying the embedded vectors \mathbf{S}_{ij}^l by an explicit matrix as follows:

$$\mathbf{F}_{ij}^l = \mathbf{w}_2^l \cdot \mathbf{S}_{ij}^l + \mathbf{b}_2^l \quad i = 1..n, j = 1..c \quad (3)$$

where,

$$\mathbf{F}_{ij}^l, \mathbf{b}_2^l \in R^{wh}$$

and,

$$\mathbf{w}_2^l \in R^{wh*m}$$

\mathbf{w}_2^l and \mathbf{b}_2^l are different and unique for each of the element and subscript (i,j) has not been used for easier interpretations. Vectors \mathbf{w}_2^l , \mathbf{b}_2^l and \mathbf{F}_{ij}^l for all the elements together will be high-dimensional tensors i.e., $\mathbf{W}_2^l \in R^{n*c*wh*m}$ and \mathbf{B}_2^l and $\mathbf{F}^l \in R^{n*c*wh}$.

Combining 1, 2 and 3, the functionality of the proposed approach can be collectively written as:

$$\mathbf{F}^l = h(\mathbf{v}^l, \mathbf{v}^{l-1}; \mathbf{W}^l, \mathbf{B}^l), \quad (4)$$

where, $h(\cdot)$ denotes the functionality of the above architecture. The final output \mathbf{F}^l will be used as the weight parameter of the l -th layer. The hypernetwork is designed in such a way that the weight matrix of the generator is covariant with its corresponding input latent vector as pruning an element in the latent vector automatically leads to removal of corresponding slice in the final weight matrix (\mathbf{F}^l). Figure 1 depicts the overall workflow of the proposed CoroNetGAN.

While designing the hypernetwork, we also execute residual connections in the network. In case of residual or skip connections, we take the input latent vector as the combination of the input latent vector of the previous layer and the corresponding layer from which the skip connection originates. We concatenate both the input latent vectors to create one single input latent vector. The resultant input latent vector along with latent vector of the current layer is used to create the latent matrix. The creation of latent matrix is followed by execution of steps(2),(3) for generating the weights matrix of the convolution layer.

3.2. Vector Sparsity using Proximal Gradient

The differentiable property of the algorithm comes through the use of proximal gradient. Proximal gradient helps in sparsification of the latent vector by searching the potential candidates. Since latent vector is covariant with the weight matrix of the Generator network, it leads to compression of the Generator network. During training time, the parameters of the hypernetwork is updated using

Stochastic Gradient Descent (SGD) optimization algorithm. During back-propagation, gradients flow from the Generator network to the hypernetwork. The latent vectors are updated using the proximal gradient [40] which leads to sparsified input latent vectors as follows:

$$\mathbf{v}[k+1] = \mathbf{prox}_{\lambda\mu R}(v[k] - \lambda\mu\nabla L(v[k])) \quad (5)$$

The proximal gradient algorithm forces the potential elements of the latent vectors to approach zero quicker than the others without any human effort and interference in this process. Due to the fact that proximal operator has closed-form solution and use of SGD, the whole solution is recognized as approximately differentiable.

3.3. Network Pruning

Our proposed method allows the weight matrix to be co-variant with the it's corresponding latent vector. Hence, sparsification of latent vector leads to the pruning of the corresponding weights of the CNN layer in the generator network. Our training regime consists of two stages, namely searching stage and converging stage. During the searching stage, proximal gradients helps in identifying the potential candidates of the latent vector. Therefore, after the searching stage, we get the sparsified latent vector (\hat{v}^l). Proximal gradient help in elements of the latent vector either be zero or approaching towards zero. We use a mask (m^l) on the sparsified latent vector with a predefined threshold (τ). This is followed by masking operation that compares every element of the latent vector with the threshold value. If greater than threshold, the returned value is one else zero. The sparsified latent vector, \hat{v}^l is pruned with the help of the computed mask (m^l).

Once the target compression ratio is achieved, the algorithm shifts from searching to the converging stage. In the converging stage, hypernetwork is discarded, and the training of the generator follows the conventional GAN training procedure. Upon extensive experimentation, it is observed that the number of epochs in searching stage is much smaller than the number of epochs in the converging stage. The pseudo-code of the proposed algorithm is mentioned in Algorithm 1.

4. Experiments

4.1. Experiment Setting

4.1.1 Models and Datasets

We evaluate our approach incorporating the following models to demonstrate the effectiveness of the proposed method:

1. Pix2Pix [16] for paired image-to-image translation with original U-Net generator architecture.

Algorithm 1 CoroNetGAN Pseudo Code

```

total_epochs ← total_number_of_epochs
target_flops ← target_compression_ratio
latent_vectors(v1, v2, ..., vi)
converging ← False
epochs ← 0

```

Compression via Differentiable Pruning

while *converging* ≠ *True* **do**

- Sample $\mathbf{m}\{z_1, z_2, \dots, z_i\}$ images from given dataset
- Sample $\mathbf{m}\{x_1, x_2, \dots, x_i\}$ ground-truths
- Update Hypernetwork using SGD

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z_i)))$$

- Update Discriminator using SGD

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \{\log D(x^i) + \log(1 - D(G(z_i)))\}$$

- Compress latent vector using proximal gradient

$$\mathbf{v}[k+1] = \mathbf{prox}_{\lambda\mu R}(v[k] - \lambda\mu\nabla L(v[k]))$$

- *epochs* ← *epochs* + 1

if *flops* − *target_flops* ≤ *threshold* **then**

converging ← *True*

end if

if *epochs* ≤ *total_epochs* **then**

break

end if

end while

Finetuning

while *epochs* ≤ *total_epochs* **do**

- Sample $\mathbf{m}\{z_1, z_2, \dots, z_i\}$ images from given dataset
- Sample $\mathbf{m}\{x_1, x_2, \dots, x_i\}$ ground-truths
- Update Generator using SGD

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z_i)))$$

- Update Discriminator using SGD

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \{\log D(x^i) + \log(1 - D(G(z_i)))\}$$

- *epochs* ← *epochs* + 1

end while

2. CycleGAN [23] for unpaired image-to-image translation using Res-Net architecture to perform transformation on an image belonging to source domain to desired target domain.
3. Deep Convolutional Generative Adversarial Network (DCGAN) [41] that uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.

For the purpose of quantitative and qualitative evaluation, four datasets are utilised including Edges → Shoes, Horse ↔ Zebra, Summer → Winter and CIFAR10.

1. Edges → Shoes [16] is a paired image-to-image

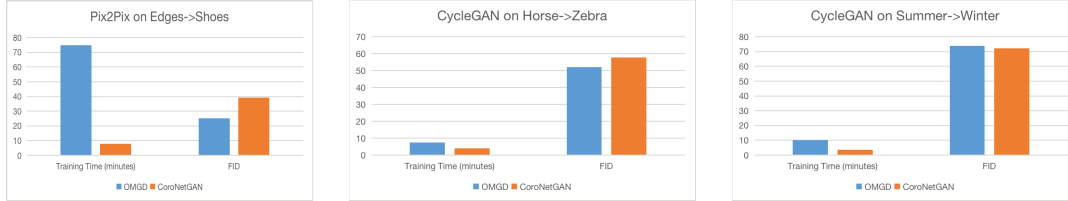


Figure 2. Graphical representation of training time(in minutes) and FID for Pix2Pix(left) on Edges \rightarrow Shoes and CycleGAN(middle,right) on Horse \rightarrow Zebra and Summer \rightarrow Winter datasets respectively. From the graphs, it is evident that total training time for our proposed approach is significantly lesser compared to OMGD [15]. For CycleGAN on Summer \rightarrow Winter dataset, our algorithm outperforms OMGD [15] on both training time and FID (*Best viewed when zoomed*).



Figure 3. Samples generated from our approach. First row contains translated images from Zebra \rightarrow Horse dataset. The second row contains translated images from Horse \rightarrow Zebra dataset (*Best viewed when zoomed*).

translation dataset including images edges of shoes to be mapped to their corresponding complete image of shoes. The dataset consists of 49825 images.

2. Horse \leftrightarrow Zebra [23] dataset contains images originally from ImageNet [42]. It is an unpaired image-to-image translation dataset used for translating horse images to zebra and vice versa. In our experiments, the training set includes 1067 horse images and 1334 zebra images.
3. Summer \rightarrow Winter [23] is also unpaired image-to-image translation dataset which translates summer images to winter. We have used 1231 summer images for training purpose.
4. CIFAR10 dataset [43] consists of 50000 training images and 10000 test images across 10 different classes.

Our approach CoroNetGAN with Pix2Pix architecture is bench-marked on Edges \rightarrow Shoes dataset. On the other hand, CoroNetGAN with CycleGAN has been bench-marked on Horse \leftrightarrow Zebra and Summer \rightarrow Winter datasets.

Although our proposed approach focuses on the compression for conditional GAN, we also made initial attempts to perform compression using our proposed algorithm for unconditional GAN (specifically DCGAN).



Figure 4. Qualitative comparison of CoroNetGAN with CycleGAN architecture on Summer \rightarrow Winter dataset compared with original CycleGAN [23], GAN Compression [44] and OMGD [15] algorithms. Our approach generates visually realistic images and outperforms all the other algorithms on the FID metric (*Best viewed when zoomed*).

4.1.2 Implementation Details

We train our proposed approach using single NVIDIA Tesla V100 GPU on PyTorch deep learning framework. For the algorithm to compress the network, a target compression ratio needs to be selected. When the difference between the actual compression and the target compression ratio falls below 2%, pruning stops and model moves to fine-tuning state from the compression state. The number of parameters in the hypernetwork is proportional to the size of the embedding space. For the experimentation, embedding space is set to 8 across all the experiments. Learning rate is set to 0.0002. Batch size has been set to 4 and 1 for Pix2Pix and CycleGAN respectively on all the experiments across different datasets. The sparsity regularization factor for the proximal gradient is set to 0.5 across all the experiments.

4.1.3 Evaluation Setting

For the quantitative performance comparison, we adopt Frechet Inception Distance (FID) [45] as common evaluation metric. FID is specifically developed for assessing the performance of GANs. It is used to evaluate the quality of the images generated by generative models by comparing the distribution of features corresponding to real and gen-



Figure 5. Qualitative comparison of CoroNetGAN with Pix2Pix architecture on Edges \rightarrow Shoes dataset compared with original Pix2Pix [16], GAN Compression [44] and OMGD [15] algorithms. Our approach generates visually plausible images compared to state-of-the-art methods (*Best viewed when zoomed in*).

erated images using an InceptionV3 [46] network. A lower FID score is an indicator of high similarity between both the distributions and thus better quality of generated images. We have evaluated FID for different architectures with our approach on multiple datasets and compared it against existing methodologies.

4.2. Experimental Results

4.2.1 Quantitative Results

We evaluate our approach on different models and datasets using evaluation setting mentioned in the previous section and report quantitative results compared with the corresponding state-of-the-art methods. The results can be summarized as follows:

Pix2Pix: We incorporate Pix2Pix with its original U-Net architecture in our proposed approach and report the experimental results in Table 1. We observe that our approach is able to achieve second best FID score on Edges \rightarrow Shoes dataset. We are also able to outperform the results of [28, 27, 47] by achieving a better FID. Although, our FID score is higher than [15] but our approach outperforms it in terms of training time. Figure 2 illustrates that our approach significantly improves training time however elevates FID score compared to [15].

CycleGAN: Similar to previous works, we include ResNet style CycleGAN in our method and report the results in Table 1. We observe that the results on Zebra \rightarrow Horse dataset outperform all the state-of-the-art approaches by achieving the best FID score of 32.3 corresponding to 75% compression. Additionally, our approach is also able to improve over all the existing baselines by achieving FID score of 72.3 on Summer \rightarrow Winter dataset. As illustrated in Figure 2 we also outperform [15] on both training time and FID.

Furthermore, we also observe that our approach with CycleGAN is able to beat the results of [28, 27, 47, 44, 48] on Horse \rightarrow Zebra dataset. Even though, we achieve greater FID score than [15, 29], CoroNetGAN outperforms [15] on

training time as illustrated in Figure 2. One thing to note is that we compare CoroNetGAN quantitative results with compression rates of 75% and 85 % using CycleGAN architecture unlike 95% in Pix2Pix since it becomes difficult to compress CycleGAN architecture beyond 85% due to its huge model complexity.

DCGAN: We demonstrate the applicability of our approach on unconditional GAN. For the experimentation and evaluating our approach, we adopt DCGAN [50] on CIFAR10 dataset [43] with original FID score of 45.8. As per the results, we were able to achieve FID score of 56.3 with 50% compression ratio which outperforms the results obtained with random pruning which achieves FID score of 68.8.

Inference Time Comparisons: Additionally, a comparative analysis of the inference time is conducted between the 95% compressed model obtained from our approach and OMGD [15]. Both the models are trained on Edges \rightarrow Shoes and evaluated using diverse smartphone chipsets and data types. The inference results, as presented in Table 3, demonstrate that our proposed model achieves superior inference time performance compared to OMGD.

4.2.2 Qualitative Results

We further show visualization results of our proposed method in comparison with state-of-the-art methodologies in Figure 3, 4 and 5 demonstrating the effectiveness of our approach. As illustrated, our method can generate high-fidelity images comparable to other state-of-the-art approaches across multiple datasets. The reason we believe our approach generates realistic images is that the compression state of our algorithm forces the generator to generate visually plausible images while competing in the min-max game.

4.2.3 Ablation Studies

Our proposed method for GAN compression shows promising results and outperform state-of-the-art methods on some conditional GANs. We perform extensive ablation studies to further demonstrate the effectiveness of hypernetworks on GAN compression on U-Net based architecture for Pix2Pix.

We tried exhaustive hyperparameter search and fine-tuning the learning rate. All the modifications result in a very negligible change of overall FID score. We also enabled a learning rate scheduler to check the improvement in model performance but quantitatively, no major change was observed. We also tried increasing the layer structure of the hypernetwork employed by increasing the dimension of the embedding space. However, this led to an increase in the training time with a small change in the overall FID score.

Model	Dataset	Paper	Params(M)	FLOPs(G)	MACs(G)	FID		
Pix2Pix	Edges → Shoes	Original [16]	54.4	–	18.6	34.31		
		Region-Aware [28]	13.61 (4.00×)	1.56	–	77.69±3.14		
		Wavelet KD [27]	13.61 (4.00×)	1.56	–	80.13±2.18		
		DMAD [47]	2.13 (25.5×)	–	2.99 (6.2×)	46.95		
		OMGD [15]	3.404 (16.0×)	–	1.219 (15.3×)	25		
		CoroNetGAN(75%)	13.225	4.8879	–	39.1		
		CoroNetGAN(95%)	4.721	1.2551	–	54.3		
		Original [23]	11.3	–	56.8	61.53		
		Region-Aware [28]	1.61 (7.08×)	7.29	–	60.01±5.22		
		Wavelet KD [27]	1.61 (7.08×)	7.29	–	61.65±4.73		
CycleGAN	Horse → Zebra	DMAD [47]	0.42 (26.9×)	–	3.97 (14.3×)	62.41		
		Teachers Do More Than Teach [29]	–	–	2.56	53.48		
		GAN Compression [44]	0.34 (33.3×)	–	2.67 (21.2×)	64.95		
		Revisiting Discriminator in GAN Compression [48]	–	–	2.40	59.31		
		OMGD [15]	0.137 (82.5×)	–	1.408 (40.3×)	51.92		
		CoroNetGAN(75%)	2.685	0.217	–	57.7		
		CoroNetGAN(85%)	1.670	0.1347	–	60.9		
		Original [23]	11.3	49.64	–	138.07±4.01		
		Region-Aware [28]	1.61 (7.08×)	7.29 (6.80×)	–	137.03±3.03		
		Wavelet KD [27]	1.61 (7.08×)	7.29 (6.80×)	–	138.84±1.47		
CycleGAN	Zebra → Horse	DMAD [47]	0.30 (37.7×)	–	3.50	139.3		
		CoroNetGAN (75%)	2.685	0.217	–	32.3		
		Original [23]	11.3	–	56.8	79.12		
		DMAD [47]	0.24 (47.1×)	–	3.18 (17.9×)	78.24		
		OMGD [15]	0.137 (82.5×)	–	1.408 (40.3×)	73.79		
		Auto-GAN [49]	–	4.34	–	78.33		
		CoroNetGAN (75%)	2.685	0.217	–	72.3		
		CoroNetGAN (85%)	1.670	0.1347	–	74.7		
		CycleGAN	Summer → Winter	Original [23]	11.3	–	56.8	79.12
				DMAD [47]	0.24 (47.1×)	–	3.18 (17.9×)	78.24
OMGD [15]	0.137 (82.5×)			–	1.408 (40.3×)	73.79		
Auto-GAN [49]	–			4.34	–	78.33		
CoroNetGAN (75%)	2.685			0.217	–	72.3		
CoroNetGAN (85%)	1.670			0.1347	–	74.7		

Table 1. Performance comparison of CoroNetGAN with state-of-the-art algorithms on Pix2Pix and CycleGAN architectures. It can be observed that our approach achieves best FID with CycleGAN on Zebra → Horse and Summer → Winter datasets. Our results also outperform [28, 27, 47, 11, 48] and achieve competitive FID on Horse → Zebra dataset. We also achieve second best FID score on Edges → Shoes dataset beating the results of [28, 27, 47].

Model	Dataset	Method	Params(M)	FLOPs(G)	FID
Pix2Pix	Edges → Shoes	Original [16]	54.41	–	34.31
		CoroNetGAN(75%)	13.225 (24.31%)	4.8879 (26.94%)	39.1
		CoroNetGAN(G + D)(75%)	Generator 13.321 (24.48%) Discriminator 0.725 (26.23%)	Generator 4.8993 (27%) Discriminator 0.4767 (26.74%)	38.6

Table 2. Generator and Discriminator compression in CoroNetGAN in Pix2Pix architecture on Edges → Shoes dataset. It is evident that compressing both generator and discriminator helps in improving the FID score.

Chipset	d-type	Model	GPU Inference Time(CL)(ms)
Qualcomm Snapdragon SM8450	32-bit	Ours	12.5419
		OMGD [15]	15.3378
	16-bit	Ours	11.794
		OMGD [15]	15.283
	8-bit	Ours	12.244
		OMGD [15]	16.0191
Dimensity 1200-Max Octa	32-bit	Ours	20.7268
		OMGD [15]	21.1919
	16-bit	Ours	20.1961
		OMGD [15]	20.9635
	8-bit	Ours	20.9972
		OMGD [15]	21.541

Table 3. Shows inference time comparison between the model compressed by our methodology to 95% and the model compressed from OMGD on different processors. Both the models are trained on Edges → Shoes dataset. The inference time is computed for the input resolution of 256×256. The 8-bit quantization of the compressed model results in increased processing time due to the presence of a higher number of quantization and de-quantization blocks compared to other data types.

Compression of both generator and discriminator:

To evaluate the significance of our approach, we design a variant CoroNetGAN(G+D compression) which compresses both the generator and discriminator till 75% during training. As mentioned in Table 2, this ablation results in improving the FID score from 39.1 to 38.6 as the generator is able to generate better results while compression of discriminator.

Generator finetuning through HyperNetwork: We also tried to finetune the weights of the generator generated from

the compression state through the HyperNetwork itself, but we did not find any significant improvements in the evaluation metric.

5. Conclusion

In this work, we propose a novel method CoroNetGAN for GAN compression based on differentiable pruning via hypernetworks. Unlike other approaches having teacher dependency overhead or post-hoc compression, the compression in our approach is done during the training time itself

giving us the benefit of reducing overall time. Experiments conducted on conditional GANs (Pix2Pix, CycleGAN) substantiate the effectiveness of our proposed method where we have been able to outperform various state-of-the-art techniques on multiple datasets without compromising the visual quality of generated images. Our approach offers significant improvement in training time and inference time as compared to the existing methods. Additionally, we also demonstrate the ability of our approach to be applicable for unconditional GANs (specifically DCGAN). The applicability on other unconditional GANs are open avenues for future work.

References

- [1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 1, 2
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1
- [3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84–90, 2012. 1
- [4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015. 1
- [5] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. 1
- [6] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2017. 1
- [7] Zihao Xie, Li Zhu, Lin Zhao, Bo Tao, Liman Liu, and Wenbing Tao. Localization-aware channel pruning for object detection. *Neurocomputing*, 403:400–408, 2020. 1
- [8] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018. 1
- [9] 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
- [10] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017. 1
- [11] Angeline M. Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya D. Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. *ArXiv*, abs/1902.00159, 2019. 1, 2, 8
- [12] Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. Distilling portable generative adversarial networks for image translation. In *AAAI*, 2020. 1, 2
- [13] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. *ArXiv*, abs/2008.11062, 2020. 1, 3
- [14] Yuchen Liu, Zhixin Shu, Yijun Li, Zhe L. Lin, Federico Perazzi, and S. Y. Kung. Content-aware gan compression. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12151–12161, 2021. 1, 3
- [15] Yuxi Ren, Jie Wu, Xuefeng Xiao, and Jianchao Yang. Online multi-granularity distillation for gan compression. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6773–6783, 2021. 2, 6, 7, 8
- [16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. 2, 5, 7, 8
- [17] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 2
- [18] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8183–8192, 2018. 2
- [19] Xinyang Li, Shengchuan Zhang, Jie Hu, Liujuan Cao, Xiaopeng Hong, Xudong Mao, Feiyue Huang, Yongjian Wu, and Rongrong Ji. Image-to-image translation via hierarchical style disentanglement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8639–8648, 2021. 2
- [20] Yahui Liu, Enver Sangineto, Yajing Chen, Linchao Bao, Haoxian Zhang, Nicu Sebe, Bruno Lepri, Wei Wang, and Marco De Nadai. Smoothing the disentangled latent style space for unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10785–10794, June 2021. 2
- [21] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018. 2

- [22] Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He, Tao Xu, Pengchuan Zhang. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. 2018. 2
- [23] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017. 2, 5, 6, 8
- [24] Peiqi Wang, Dongsheng Wang, Yu Ji, Xinfeng Xie, Haoxuan Song, XuXin Liu, Yongqiang Lyu, and Yuan Xie. QGAN: Quantized generative adversarial networks. *ArXiv*, abs/1901.08263, 2019. 2
- [25] Ting-Yun Chang and Chi-Jen Lu. TinyGAN: Distilling bigGAN for conditional image generation. In *ACCV*, 2020. 2
- [26] Liang Hou, Zehuan Yuan, Lei Huang, Huawei Shen, Xueqi Cheng, and Changhu Wang. Slimmable generative adversarial networks. In *AAAI*, 2021. 2
- [27] Linfeng Zhang, Xin Chen, Xiaobing Tu, Pengfei Wan, Ning Xu, and Kaisheng Ma. Wavelet knowledge distillation: Towards efficient image-to-image translation. *ArXiv*, abs/2203.06321, 2022. 2, 7, 8
- [28] Linfeng Zhang, Xin Chen, Runpei Dong, and Kaisheng Ma. Region-aware knowledge distillation for efficient image-to-image translation. *ArXiv*, abs/2205.12451, 2022. 2, 7, 8
- [29] Qing Jin, Jian Ren, Oliver J. Woodford, Jiazhao Wang, Geng Yuan, Yanzhi Wang, and S. Tulyakov. Teachers do more than teach: Compressing image-to-image models. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13595–13606, 2021. 3, 7, 8
- [30] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. *CoRR*, abs/1605.09673, 2016. 3
- [31] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. *Hypernetwork Functional Image Representation*, pages 496–510. 09 2019. 3
- [32] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. 10 2017. 3
- [33] David Ha, Andrew Dai, and Quoc Le. Hypernetworks. 09 2016. 3
- [34] Christoph von der Malsburg. The correlation theory of brain function. *Models Neural Netw.*, 2, 01 1994. 3
- [35] Jürgen Schmidhuber. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4(1):131–139, 01 1992. 3
- [36] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. 12 2014. 3
- [37] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. *CoRR*, abs/1606.09549, 2016. 3
- [38] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. 3
- [39] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015. 3
- [40] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *Proceedings of the European Conference on Computer Vision*, 2020. 3, 5
- [41] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2016. 5
- [42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6
- [43] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 7
- [44] MUYANG LI, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional GANs. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5283–5293, 2020. 6, 7, 8
- [45] 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. *Advances in neural information processing systems*, 30, 2017. 6
- [46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 7
- [47] Shaojie Li, Mingbao Lin, Yan Wang, Fei Chao, Ling Shao, and Rongrong Ji. Learning efficient GANs for image translation via differentiable masks and co-attention distillation. *IEEE Transactions on Multimedia*, 2022. 7, 8
- [48] Shaojie Li, Jie Wu, Xuefeng Xiao, Fei Chao, Xudong Mao, and Rongrong Ji. Revisiting discriminator in gan compression: A generator-discriminator cooperative compression scheme. In *NeurIPS*, 2021. 7, 8
- [49] Y. Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. AutoGAN-Distiller: Searching to compress generative adversarial networks. In *ICML*, 2020. 8
- [50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 7