Compressing SinGAN with Pruning and Quantization

Meng Wei
Department of Computer Science and Technology
University of Cambridge
Cambridge, CB3 0FD
mw825@cam.ac.uk

Abstract

Single Image GAN (SinGAN) [25] is a generative adversarial neural network that only requires a single natural image as the training data and synthesizes realistic images sharing similar contents. SinGAN has a multi-scale architecture, where each scale has a pair of a discriminator and a generator that captures the patch distribution at the current scale. However, because of this multi-scale architecture, training and inferring SinGAN is memory intensive especially for high-resolution images. To cope with these issues, given the success of pruning and quantization on deep neural networks, we enhance the SinGAN with these techniques for resource efficiency. We argue that the generators of the SinGAN are over-parameterized because they are modeling the low-dimensional Laplacian information of the image. We apply four model compression methods with ablation studies to reduce the memory occupancy of the SinGAN during the inference time. The proposed methods support the applications of SinGAN on high-resolution images and deployment on small memory devices or GPUs. We compare the generating capacities of these methods in terms of the Single Image Fréchet Inception Distance (SIFID) and demonstrate the memory efficiency of the model compression methods during the inference time with almost zero trade-offs of the performance.

1 Introduction

Generative Adversarial Networks [7, 14, 15] have demonstrated their promising performance in generating objects in various categories, including faces, architectures, and animals. Most GANs need large training datasets and require enormous computation resources. Shaham et al. [25] propose a Singel Image GAN (SinGAN) that improves the generating capacities of GANs and only requires a single natural image as the training data, but synthesizes realistic images sharing visual contents similar to the training image. SinGAN could be used to handle multiple traditional image manipulation tasks, like super-resolution, image harmonization and image shuffling. Because SinGAN is unconditioned and only require a single image as the training data, deploying SinGAN on cameras or mobiles has potential applications.

The deployment of SinGAN on small memory devices needs further improvements to compress the model because training and inferring SinGAN is memory intensive. SinGAN has a multi-scale architecture, where each scale has a pair of a discriminator and a generator that captures the patch distribution at the current scale. This architecture enriches the training data space with distributions from the same image at different scales and enables SinGAN to model the overall structures and detailed information from the coarsest to the finest scale. Despite its good performance, training and inferring using the multi-scale architecture requires a large amount of memory, especially for high-quality images because they have more levels of scales. Therefore, refining the model structures while maintaining the performance of SinGAN is necessary for the deployment on small memory devices or applications on high-resolution images.

Preprint. Under review.

Network pruning [6, 8, 17, 10, 18, 20, 19, 27] and weights quantization [8, 24, 5] techniques are widely used to compress Neural Networks for resources efficiency. Non-structured [6, 8, 17] and structured pruning methods [17, 10, 18, 20] eliminate over-parameterized weights or channels according to some criteria. They usually follow the training-pruning-finetuning [19] pipeline to recover the pruned model's performance. Weights quantization methods also compress the model by decreasing the precisions of model parameters. The quantization could be achieved by clustering [8] or merely reducing the number of bits required to represent model weights [24, 5]. Recent studies of pruning techniques have demonstrated their efficiency on Convolutional Neural Networks [8] and Recurrent Neural Networks [22], which implies their potentials on compressing SinGAN. Song et al. [8] also combine the pruning and weights quantization strategies to compress the model. We want to analyze the capability of compressing SinGAN using pruning and weights quantization techniques.

SinGAN has multiple pairs of generators and discriminators. Because usually only the generators are used during the inference, we aim to compress generators from SinGAN. The overall multi-scale architecture of the SinGAN is similar to the Laplacian Pyramids [3], where each generator models the Laplacian distributions of the image at the current scale. The image Laplacian is low-dimensional and compact, which suggests that learning and modelling the Laplacian distributions should not require a deep and involved model. Therefore, we argue that the original generators used by SinGAN are over-parameterized and applying pruning and weights quantization techniques on them are plausible.

Contributions. We apply four model compression methods to compress the generators of SinGAN, including 'structured pruning from scratch [27]', 'non-structured pruning [8]', 'structured pruning [17]', and 'the mixture [8] of weights quantization [24] and structured pruning [17]'. We also conduct ablation studies of structured pruning to achieve an optimistic results. We then compare their results using the Single Image Fréchet Inception Distance (SIFID) and demonstrate the memory efficiency of our proposed methods during the inference time with almost zero trade-offs of the performance.

2 Related Work

2.1 Single Image GAN (SinGAN)

SinGAN [25] models the internal distribution of a single natural image x by constructing a multi-scale architecture to learn the image's patch distributions from the roughest scale to the finest scale. The multi-scale architecture is a pyramid of various fully convolutional GANs [23] $\{(G_0, D_0), (G_1, D_1), \ldots, (G_N, D_N)\}$, where each GAN is trained on the down-sampled images $\{x_0, x_1, \ldots, x_N\}$. x_0 is the image with the maximum image size we want to generate and is down-sampled from the original image x for some scaling factor $r^n, 0 < r < 1, n \in \mathbb{Z}$; x_N is the smallest image down-sampled by the scaling factor r^{n+N}. Each fully convolutional GAN (G_i, D_i) at the scale i consists of a generator G_i that generates realistic images $\tilde{x}_i$ similar to the down-sampled real image x_i and a Markovian discriminator (PatchGAN) [13, 16] D_i that discriminates patches in $\tilde{x}_i$ from real patches in x_i. These PatchGANs D_i have receptive fields of the same size for all levels. In other words, the patch size remains unchanged while the image sizes increase as the level increases. Therefore, generators at the upper level learn the detailed information of the input image, whereas lower-level generators capture the overall patch distribution of the down-sampled picture. Both the generator and the discriminator have the Convolution-BatchNorm[12]-LeakyReLU[21] architecture, similar to the DCGAN [23].

Training and inferring the SinGAN follow a bottom-up procedure, from the roughest scale x_N to the finest scale x_0, illustrated by Figure 1. The bottom generator G_N is fully generative, which means it takes a random Gaussian noise z_N as the input and synthesizes images $\tilde{x}_N$ similar to the real image x_N. After training the low-level generator G_i, $0 < i \leq N$, G_i is fixed and used for the residual learning [9] to train G_{i-1} at the next scale. The generator G_{i-1} takes up-sampled synthetic images $(\tilde{x}_i) \uparrow^{1/r}$ from the previous scale and injects random Gaussian noises z_{i-1} to produce realistic images $\tilde{x}_{i-1}$. Here, $0 < r < 1$ is the mentioned down-sampling scaling factor, and its inverse $\frac{1}{r} > 1$ is used for up-sampling. In general, we have

$$\tilde{x}_i = \begin{cases} G_N(z_N) & i = N \\ G_i(z_i, (\tilde{x}_{i+1}) \uparrow^{\frac{1}{r}}) & 0 \leq i < N \end{cases} \tag{1}$$

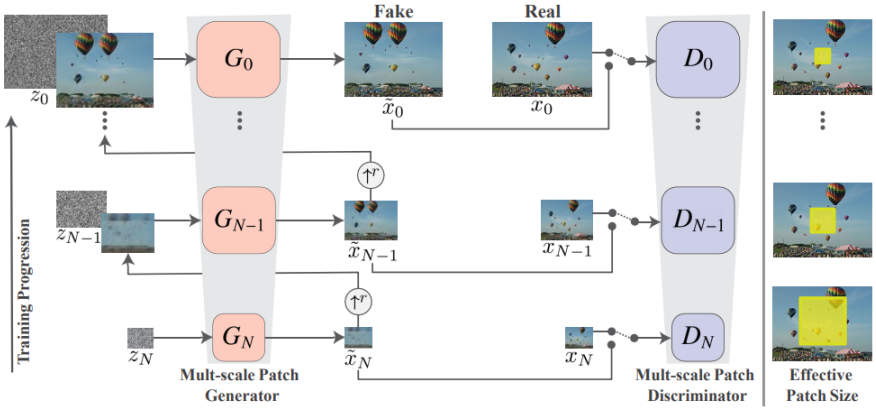


Figure 1: SinGAN architecture (*source:* [25]). SinGAN has a multi-scale architecture, where each scale has a pair of discriminator and generator (G_i, D_i). The generator takes the up-sampled outputs from the previous level and models patch distributions at the current scale, while the Markovian discriminator (PatchGAN) distinguishes real or fake patches. The Markovian discriminators have same receptive fields for all scales, which capture fine details at the upper scale while model the overall image structures at the lower level.

The overall multi-scale architecture of the SinGAN is similar to the Laplacian Pyramids [3]. In fact, because each generator takes synthetic images from the previous scale and performs the residual learning [9] to map noises with image differences, each generator G_i models the Laplacian distribution of x_i at the current scale i. Compared with the original raw image data x_i, which has high dimensionality, the Laplacian information of x_i is compact and has low dimensions. Therefore, we argue that the original generators used by SinGAN are over-parameterized and applying pruning and quantization methods on them is plausible.

2.2 Non-Structured and Structured Pruning

Over past years, deep neural networks have increasing parameters because of complex architectures, which results in intensive memory requirements and expensive computation costs [1]. Pruning strategy aims to compress models and accelerate forward computations by removing insignificant weights or channels from the over-parameterized models.

The pruning strategy is initially unstructured, which operates on individual weights globally or layer-wisely and eliminates insignificant weights according to some criteria (usually by magnitudes) [6, 8]. Although unstructured pruning introduces the sparsity to the model and presents light architectures, it has some drawbacks. The non-structured sparsity might not speed-up the forward propagation nor support model compressions because general-purpose GPUs operate in parallel and cannot benefit from non-structured sparsity [8].

On the contrary, the structured pruning strategies [17, 10, 18, 20] prune weights at the level of channels or even layers. They preserve the structured (compact) representations of weights, and thus suit the general-purpose GPUs. Therefore, compressed models using structured pruning techniques have accelerated computations and small memory. Hao et al. [17] prune model channels layer-wisely based on the L1-norm of weights. Yihui et al. [10] iteratively prune redundant channels and reconstruct output features by minimizing reconstruction errors.

Although pruning strategies compress the model and reduce the memory requirements, pruning usually diminishes original models' performance because they eliminate trained weights or channels. Without recovering the original model's performance, pruning strategies cannot be applied iteratively to achieve desired sparsities because remaining weights significantly influence the outputs. Therefore, pruning strategies usually follow a training-pruning-finetuning pipeline [19]. A large dense model is pre-trained first to obtain its initial weights, which include redundant parameters. Then, a structured or non-structured pruning strategy is applied to compress this model. It eliminates unimportant

parameters by measuring their magnitudes [17] or minimizing the output features' reconstruction errors [10, 20]. The obtained light architecture is finally fine-tuned to recover performance.

Fine-tuning a pruned model usually uses inherited trained weights from the 'training' stage because they are considered essential and influential. However, Zhuang et al. [19] find that for the structured pruning, the 'fine-tuning' from scratch (with randomly initialized weights) can achieve an equivalent or even better performance compared with continuing training using the inherited weights. Zhuang et al. [19] state that 'fine-tuning' from the inherited weights could cause the compressed model to be trapped at the local minimum, even if the inherited weights are representative. Their discovery implies the pruned architecture is more important for the model performance than its inherited weights. Besides [19], Wang et al. [27] argue that the 'training' stage in the pruning pipeline might not be necessary either. They assert that directly searching the optimal architecture for the compressed model would produce better results. Therefore, structured pruning from scratch might save time and provide a broader search space for the architectures.

In our study, because the generators of SinGAN are fully-convolutional and have a simple Convolution-BatchNorm[12]-LeakyReLU[21] architecture, structured pruning from scratch [27] in our case is equivalent to changing the number of filters of inner convolutional layers. We also follow the training-pruning-finetuning pipeline and apply the non-structured and structured pruning strategies to achieve model compressions. We follow the work of [8, 17] and globally prune the insignificant weights based on their magnitudes. More design details and results are presented in Section 3 and 4.

2.3 Weights Quantization

Weights quantization strategies aim to reduce the model size by changing the number of bits of parameters. Instead of using 32 bits to represent parameters or activations, Courbariaux et al. [5] and Rastegari et al. [24] change the weights to half-precision (16 bits) or even binaries. These weights quantization strategies could compress models and accelerate GPU computations. However, although low precision neural networks have dramatically decreased memory sizes, they might have unstable performance as the trade-off. Instead, Song et al. [8] explicitly measure the weights distributions and map parameters with similar values to various targets using clustering. Their weights sharing strategy needs specially designed GPUs or software to support the indirect matrix index lookup [8], thus cannot be generally deployed.

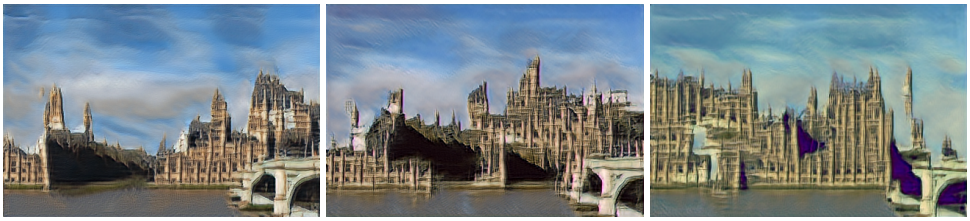
Given the success of the pruning and weights quantization techniques, a mixture method of both techniques could achieve a maximal compression ratio. Song et al. [8] utilize this idea and apply it on various neural networks and achieve almost zero accuracy loss. We follow their ideas and compress the generators of SinGAN and want to achieve similar results.

2.4 Other Memory Reduction methods

Knowledge distillation [11] reduces the memory occupancy by training a light neural network to imitate a complicated one. This technique has been applied on the GANs to compress models or help understand their architectures. David et al. [2] use the knowledge distillation strategies to find an inversion of every single layer of generators inside a GAN. They use the distilled neural networks to test the generatability of GANs. Hanting et al. [4] utilize the knowledge distillation to compress and construct a portable GAN. However, finding the distilled model is non-trivial, especially for the SinGAN, which consists of multiple fully convolutional GANs. Therefore, we stick to the pruning and weights quantization methods to enhance the SinGAN for memory resource efficiency. We focus on the generators of SinGAN because usually only the generators are used for the inference.

3 Algorithms

The memory occupancy of SinGAN's generators is determined by the scale number N and generators' size at each scale. In this section, we apply four model compression methods to compress SinGAN's generators for inference memory efficiency. We propose 'structured pruning from scratch [27]', 'non-structured pruning [8]', 'structured pruning [17]', and 'the mixture [8] of weights quantization [24] and structured pruning [17]' to eliminate redundant weights of the generators in SinGAN. We also analyze the importance of fine-tuning in the three-stage pruning pipeline [19]. Moreover, because



(a) Without Pruning (b) Fine-Tuning After Pruning (c) Pruning Without Fine-Tuning

Figure 2: Comparison between the pruning without fine-tuning image (c) and the fine-tuning after pruning image (b). Both images are pruned using the 50% sparsity non-structured pruning method. Image (b) has colours similar to the ground truth image (a) while image (c) has inconsistent colours.

of the bottom-up training procedure of the SinGAN, we discuss that missing information caused by model compression methods at the lower level can be regained at the upper level of the SinGAN.

3.1 Method 1: Structured Pruning from Scratch.

Wang et al. [27] and Zhuang et al. [19] discover that instead of following the training-pruning-finetuning pipeline, finetuning or even pruning from scratch could achieve an equivalent performance for the structured pruning method. Their discoveries suggest that the pruned model's architectures are essential for the model performance other than the inherited weights. We follow their ideas and use the 'Structured Pruning from scratch' as our first compressing method. Because the SinGAN generators have a simple structure, applying 'structured pruning from scratch' on these generators is equivalent to directly reducing the number of filters of convolutional layers and training the generators from the reduced architectures.

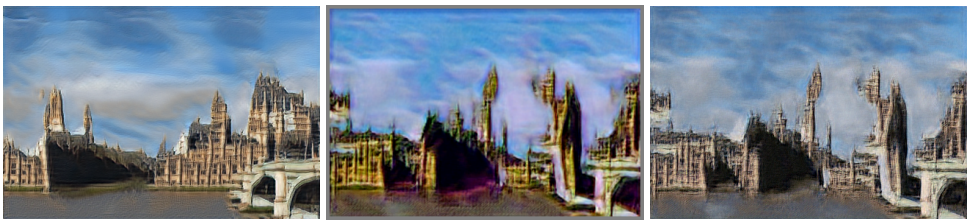
Training generators in Generative Adversarial Networks require the discriminators' signals because generators and discriminators perform Min-Max games against each other during the training [7]. Merely compressing models through structured pruning from scratch, which in our case is equivalent to reducing the number of channels, might not work for the GANs because reducing generators' architectures may break the training balance between discriminates and generators. We could simultaneously reduce the discriminators' architectures to handle this problem, but because discriminators perform the classification tasks during the training, pruning would dramatically decrease their performances. Therefore, we leave the architectures of discriminators unchanged and only reduce the number of generators' filters in each convolutional layer in the SinGAN.

3.2 Method 2: Non-structured Pruning

Both non-structured and structured pruning methods eliminate redundant weights and keep representative parameters. Determining which weights are insignificant is non-trivial. It could be achieved by merely measuring the magnitudes of weights [8] or calculating the pruning's reconstruction errors [17]. Because we aim to reduce the generators' memory requirements instead of proposing an optimal pruning strategy for GANs, we keep the simplicity and follow [8] to prune weights with least magnitude. We compare weights' magnitudes globally instead of layer-wisely to achieve desired sparsities. Moreover, because we are compressing the unsupervised generators other than supervised models like classification, we do not have the ground truth targets to evaluate and guide the model during the training. Therefore, we only perform the pruning strategy once instead of iteratively pruning and fine-tuning models.

The fine-tuning stage of the pruning pipeline [19] is essential, especially for the non-structured pruning. Without fine-tuning, the model's performance would decrease dramatically even if we merely eliminate insignificant weights [8, 26]. Figure 2 visualize the effects of fine-tuning when we apply non-structured pruning to achieve 50% model sparsity. Without fine-tuning, models would produce inconsistent colors or structures, while the fine-tuning stage recovers the model performance.

Information regained at upper levels. SinGAN has a bottom-up training procedure, where generators at the upper-level model the differences between up-sampled outputs from the previous scale and the real down-sampled image. Therefore, when we perform the pruning strategies on the lower-level



(a) Baseline (b) Outputs from Previous (c) Information Regained

Figure 3: Illustration of Information Regain. Figure (b) is the output from the previous generator after 50% sparsity non-structured pruning without fine-tuning. Figure (c) has the up-sampled (b) as the input and models the missing information.

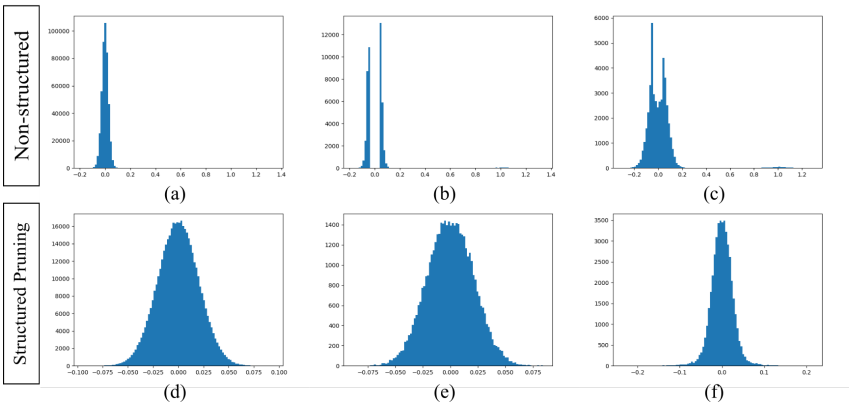


Figure 4: Comparison between structured and non-structured pruning parameters. (a) and (d) are original parameters. (b) and (e) are pruned parameters. (c) and (f) are fine-tuned parameters.

generators, the missing information caused by the model compression methods could be regained at the upper-level. Figure 3 illustrates this phenomenon where we apply the 50% sparsity non-structured pruning method without fine-tuning and directly pass damaged outputs to the next level. Although the upper-level generators have re-learned most missing information, the overall performance is not trustworthy if we do not fine-tune the pruned model.

3.3 Method 3: Structured Pruning

One major drawback of the non-structured pruning method is that it cannot produce dense models suiting for general-purpose GPUs [8]. On the contrary, the structured pruning method eliminates filters or even layers of the model, resulting in the dense representations of the remaining architecture. Figure 4 visualizes the weights of structured and unstructured methods. As we can see, the weights of structured pruning method remain the Gaussian distribution similar to the original weights.

For the structured pruning, we follow the similar idea in the previous section and perform the pruning based on weights magnitudes of each filter [17]. We argue that unlike the good outputs of applying structured pruning on the supervised models, eliminating filters of convolutional layers in the GAN may result in unwanted effects. More visualizations are presented in the Experiment Section 4.

3.4 Method 4: Mixture of Structured Pruning and Weights Quantization

The final model compression method we propose is the mixture of the structured pruning and weights quantization. The mixture [8] is generally used in the model compression to achieve the maximal compression ratio. We use a simple half-precision to quantize weights, and the reduction of parameters' bits is performed after we fine-tune the pruned model at each scale.

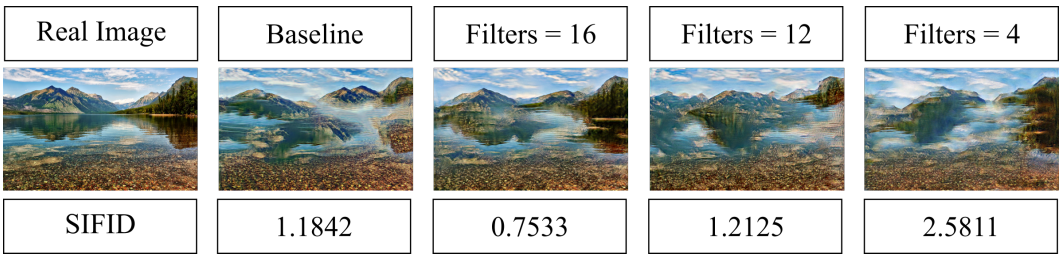


Figure 5: Comparison among images with different filter sizes for the 'Structured Pruning from Scratch' method. The SIFID scores are at the magnitude of e^{-05}, which implies all these pictures are realistic. However, images with filters=$16, 12$ are visually better than filters=4.

4 Experiments

In this section, we evaluated our proposed model compression methods on the generators of SinGAN during the inference time. Because training the SinGAN was the process of min-max optimization between generators and discriminators, accelerating and compressing the training stages of GANs were non-trivial and beyond the scope of this report. We used the uncompressed SinGAN as the ground truth and baseline methods and compared our compressed models' performance and memory requirements. The architecture of the baseline SinGAN model we used followed the default settings in [25], which had '5 convolutional blocks' with '32 filters' initially and doubled every 4th scales. The downsampling factor we used was'$0.75 = 3/4$' which would produce good results according to [25]. We trained the SinGAN with '2000' epochs and '3' generator and discriminator inner steps. The SinGAN [25] architecture we were using was implemented in PyTorch by the original authors. We modified some parts of the code to fit our purposes, and the code repository was published online[1]. We compared mentioned model compression algorithms both visually and quantitatively using the Single Image Fréchet Inception Distance (SIFID), which would be discussed in the following section 4.2.

4.1 Settings for the purning strategy.

For the 'Pruning from Scratch' method, we constructed generators with 'filters$= 4, 12, 16$' which would be doubled every 4th scale as usual. Compared with the default '32 filters', they achieved around '$87.5\%, 62.5\%$ and 50%' compression ratios because the generators were fully convolutional the most dominant features were from the convolutional layers. For the non-structured and structured pruning methods, we pruned unimportant weights based on their magnitudes. We found that using '$L1$' or '$L0$'-norm did not influence the pruning performance; thus, we stuck to the $L1$-norm. We argued that iteratively pruning layerwisely did not suit the generation task because we did not have a ground truth value to map; thus, we pruned weights globally at once for simplicity. We tested the pruning strategies with '50% and 90%' sparsities and followed the training-pruning-finetuning pipeline [19]. According to [19] and [27], the performance of the structured-pruning method was highly correlated with the introduced light architecture instead of inherited weights. Therefore, we warmed up the SinGAN with 100 epochs while performing the structured pruning strategy and finetuned with '$2000 - 100 = 1900$' epochs after pruning. As for the non-structured pruning method, pre-training was relatively important [19]; thus, we warmed up the SinGAN with 1000 epochs and finetuned with $2000 - 1000 = 1000$ epochs after pruning.

4.2 Evaluation Metrics

We evaluated the model compression methods on the generators of SinGAN both visually and quantitatively. We aimed to measure the compressed model size of generators during the inference and their generating capacities after the compression. According to Nimit et al. [26], the generators' model memory could be grouped into three categories: weights, activations and optimizer memory. The pruning strategies introduced sparsities to the model, which reduced the weights and optimizer memory; whereas, the weights quantization caused all three categories' memories to decrease.

[1] https://github.com/Meng-Wei/SinGAN

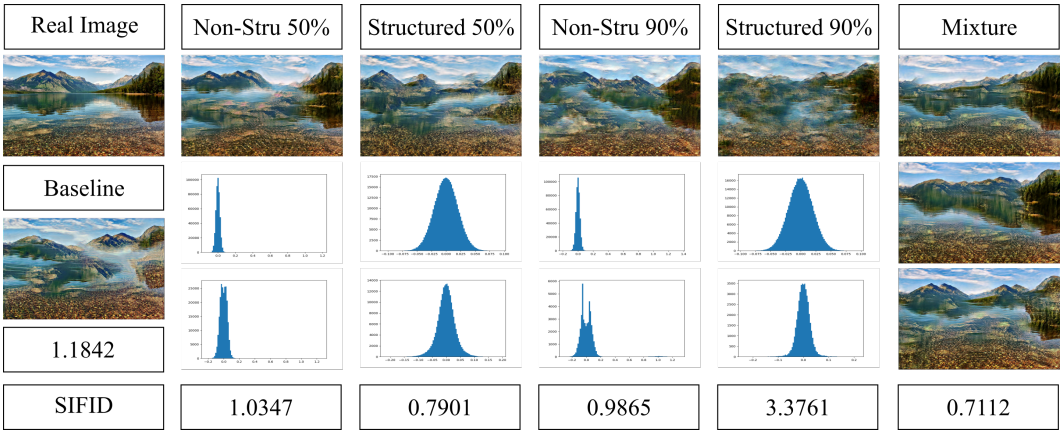


Figure 6: Comparison among proposed methods. 50% and 90% are weights sparsities. The first row contains generated images. The second row represents un-pruned weights. The Third row represents pruned weights after fine-tuning. The last row is the SIFID of each image at the magnitude of e^{-05}.

Because we focused on the generators of SinGAN, we evaluated the compressed model size in terms of their weights memory for efficiency. Moreover, for a fair comparison between non-structured and structured pruning, we evaluated the sparsities of the weights.

Because SinGAN had a single natural image as the training data, the traditional Fréchet Inception Distance (FID) score could not be applied in our cases. We followed [25] to quantify the performance of compressed SinGAN in terms of the Single Image Fréchet Inception Distance (SIFID). The SIFID [25] measured the deviation between deep features of patch distributions of real and fake images. Similar to FID, lower SIFID implies better generating capacities of the model.

4.3 Quantitative and Visual Comparisons

Figure 5 and Figure 6 illustrated the compressed models' performance. All these SIFIDs are at the magnituede of e^{-05}. From these images, we could observe that the compressed models could achieve quantitatively equivalent results compared with the baseline method. The 90% structured pruning technique in Figure 6had a blurred effect and the highest SIFID. Moreover, the Structured Pruning from Scratch with filters=4 in 5 also had rough textures. Unlike the supervised models, structured pruning for the GAN (either from scratch or not) did not necessarily achieve better or comparable performance with the non-structured pruning, because pruned filters might be necessary to the generating capacities of GANs. The mixture method we presented here used the 50% structured pruning method and half-precision, which also produced realistic results, suggesting we were safe to use the high-precision neural networks to generate realistic images with reduction of weights and activation memories in half.

The compression methods could be applied to the generators of the SinGAN and achieved desired sparsities with almost zero trade-offs of the SIFID. Their success implied the redundant and over-parameterized architecture of the SinGAN because they were modeling the low-dimensional image Laplacian. The information regaining discussed in 3.2 also enhanced the model compression methods' performance. The structured pruning strategy's failure suggested that pruning filters of the GANs could not achieve comparable performance like their applications on supervised models.

5 Conclusion and Future Work

In this paper, we utilize four model compression methods to save the memory requirements of the generators of the SinGAN during inference with almost zero trade-offs in terms of the SIFID. Their success implies the over-parameterized architectures of the SinGAN, because they model the low-dimensional image Laplacians which do not require large-dense models. Future work could include modifying and designing a concise architecture of the SinGAN or deploying SinGAN on small memory devices.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27:2654–2662, 2014.

[2] David Bau, Jun-Yan Zhu, J. Wulff, W. Peebles, Hendrik Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4501–4510, 2019.

[3] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[4] 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 *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3585–3592. AAAI Press, 2020.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 3123–3131. Curran Associates, Inc., 2015.

[6] Yann Le Cun, John S. Denker, and Sara A. Solla. *Optimal Brain Damage*, page 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[8] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[9] 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.

[10] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406, 2017.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[13] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[16] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 702–716, Cham, 2016. Springer International Publishing.

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[18] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2763, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society.

[19] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[20] J. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5068–5076, 2017.

[21] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. JMLR, 2013.

[22] Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. Exploring Sparsity in Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1704.05119, April 2017.

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[24] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing.

[25] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[26] N. S. Sohoni, C. R. Aberger, Megan Leszczynski, Jian Zhang, and C. Ré. Low-memory neural network training: A technical report. *ArXiv*, abs/1904.10631, 2019.

[27] Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:12273–12280, 04 2020.