BlazeStyleGAN: A Real-Time On-Device StyleGAN

Haolin Jia, Qifei Wang, Omer Tov, Yang Zhao, Fei Deng, Lu Wang, Chuo-Ling Chang, Tingbo Hou, Matthias Grundmann
Google
{haolinmz, qfwang, omertov, yzhaoeric, dfei, luwa, chuoling, tingbo, grundmann}@google.com

Abstract

StyleGAN models have been widely adopted for generating and editing face images. Yet, few work investigated running StyleGAN models on mobile devices. In this work, we introduce BlazeStyleGAN — to the best of our knowledge, the first StyleGAN model that can run in real-time on smartphones. We design an efficient synthesis network with the auxiliary head to convert features to RGB at each level of the generator, and only keep the last one at inference. We also improve the distillation strategy with a multi-scale perceptual loss using the auxiliary heads, and an adversarial loss for the student generator and discriminator. With these optimizations, BlazeStyleGAN can achieve real-time performance on high-end mobile GPUs. Experimental results demonstrate that BlazeStyleGAN generates high-quality face images and even mitigates some artifacts from the teacher model.

1. Introduction

Generative Adversarial Networks (GANs) [5] are well known for their impressive image generation capabilities. As a successful example, the StyleGAN family [11–13] has been widely adopted for unconditional face generation and various face editing tasks. However, due to the high computational complexity, many of the applications have to run offline on powerful machines. Very few works investigated running StyleGAN models on mobile devices. An earlier attempt, MobileStyleGAN [2] was distilled from StyleGAN and can run faster than StyleGAN on Intel CPUs. However, it did not achieve real-time performance on mobile devices. Along with the rising need of real-time experiences in mobile apps, including short videos, virtual reality, and gaming, accelerating StyleGAN models to achieve real-time on-device performance will enable more applications.

To improve the efficiency, various GAN compression methods have been proposed. [3, 4] use knowledge distillation [8] for compressing general GAN architectures. [15, 18] further combine channel pruning [6] with knowledge distillation to improve performance. However, these methods are designed for compressing conditional GANs, which usually have paired training data. Unconditional GANs are in general more challenging to compress due to their unpaired training setting. [16] shows that directly applying [18] to StyleGAN2 [13] leads to sub-optimal performance.

In this paper, we propose BlazeStyleGAN, an efficient StyleGAN implementation to optimize model performance and on-device latency. We revisit the complexity of StyleGAN, and notice that the modulated convolutions and the feature-to-RGB modules are taking significant inference time. To address these issues, we simplify the modulated convolution, and design an efficient auxiliary head to convert features to RGB (called UpToRGB) at each level of the generator. At inference time, we only keep the last auxiliary head and remove others. We adopt the popular StyleGAN2 [13] as our teacher model, and perform knowledge distillation to train our BlazeStyleGAN. We also introduce a multi-scale perceptual loss to improve the model's generation capability by learning image generations at multiple levels of granularity. Our BlazeStyleGAN is smaller than MobileStyleGAN [2] in terms of parameter numbers and model complexities. To demonstrate the performance, we benchmark BlazeStyleGAN on various smartphones, where it achieves real-time performance on mobile GPUs. The visual quality of the image generated by BlazeStyleGAN is similar to its teacher model. We also observe that, in some examples, BlazeStyleGAN can even improve visual quality by mitigating the artifacts generated from the teacher model. BlazeStyleGAN achieves fair Fréchet Inception Distance (FID) [7] scores compared to the teacher StyleGAN.

Our contributions can be summarized as follows:

- We design a mobile-friendly architecture by introducing a new auxiliary UpToRGB head at each level of the generator, and only running the last one at inference.
- We improve the distillation strategy by computing a multi-scale perceptual loss with the auxiliary heads and an adversarial loss against real images, which improves image generation and suppresses transferring

artifacts from the teacher model.

- Our BlazeStyleGAN achieves real-time performance on many popular smartphones, while preserving high-quality image generation.

2. Related Work

Early work on unconditional GAN compression [1] focuses on low-resolution images. It uses an MSE loss to make the student generator produce similar images as the teacher generator when given the same latents. With the advent of StyleGANs [11–13], compressing StyleGANs for high-resolution image synthesis has attracted a lot of interest. Content-Aware GAN compression (CAGAN) [16] adapts channel pruning and knowledge distillation proposed in [18] to work with unconditional GANs, and uses a content-of-interest mask to guide the pruning and distillation process. However, this requires the student network to inherit the main structure of the teacher network. Xu *et al*. [19] find that when the student and teacher networks have different architectures, the output discrepancy limits the performance of CAGAN, and propose an initialization strategy to solve the issue. MobileStyleGAN [2] takes a different approach by leveraging frequency-based image representations and using the wavelet transform as the prediction target for the student model. However, we found that frequency-based image representation misses details in generated images, and the model efficiency can be further to achieve real-time performance.

3. Model Architecture

The StyleGAN generator contains two main components, the mapping network and the synthesis network. The mapping network is designed as a multi-layer perceptron (MLP) to map the input latent z to an intermediate latent space W as the style input to the adaptive instance normalization for each convolutional layer in the synthesis network. The synthesis network contains multiple convolutional layers that generate images from style input (A in Fig. 1) and noise (B in Fig. 1). Since the synthesis blocks contribute to the majority of model parameters, we focus on the design of an efficient synthesis network to improve on-device performance.

The StyleGAN synthesis network is composed of a stack of convolutional blocks, with 3×3 convolutional layers, upsampling layers, and adaptive instance normalization layers. Each synthesis block is attached with a latent-feature-to-RGB (ToRGB) block to calculate multi-scale perceptual loss. Such a complicated architecture results in a cumbersome model, which is unfriendly to on-device applications. Following MobileNet [9] for using depth-wise convolution, MobileStyleGAN [2] proposes a modulated depth-wise convolutional (DWModulatedConv) layer to merge the

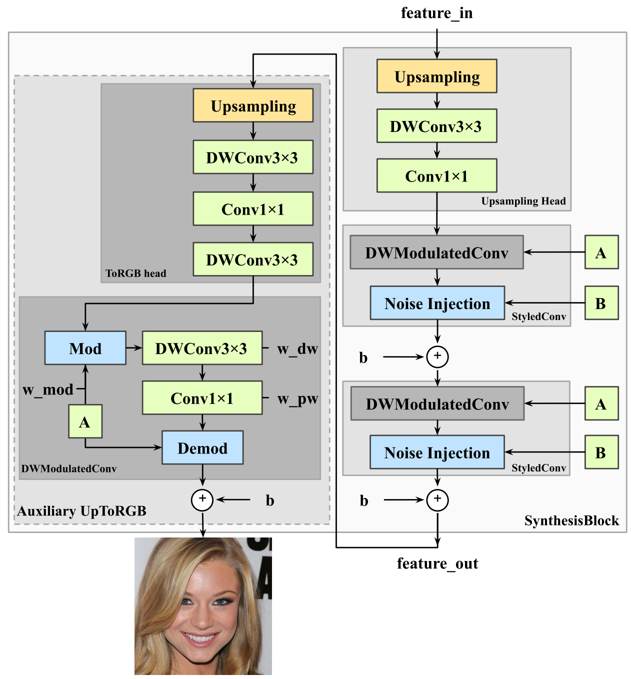

Figure 1. The synthesis block of BlazeStyleGAN. See text for details.

adaptive normalization with the depth-wise convolution. It transforms the latent feature to RGB via a single frequency domain transformation at the end of the synthesis network. Thus, it significantly reduces the number of model parameters from StyleGAN. Although MobileStyleGAN claims the wavelet transformation can enhance the high frequency details, we observe that using a single feature-to-RGB layer at the end of the chains is prone to losing image details.

To achieve better quality, we design an efficient synthesis network with a new feature-to-RGB transformation block, as shown in Fig. 1. Each synthesis block has an auxiliary head (named as UpToRGB) to upsample and transform its latent feature to an RGB image. To reduce the complexity of the synthesis block, we downsample the latent feature map's resolution of each synthesis block in BlazeStyleGAN to $1/4$ of the resolutions of the counterpart layers in the teacher StyleGAN's synthesis blocks. To match the teachers' output resolution, each feature map is further upsampled to the target resolution in the UpToRGB head as shown in Fig. 1. Since only the UpToRGB head processes the full resolution feature map in each block, the complexity is much reduced compared to processing the full resolution feature map in the synthesis block. The RGB images output by the auxiliary UpToRGB heads construct a coarse-to-fine pyramid used for calculating the multi-scale perceptual quality loss.

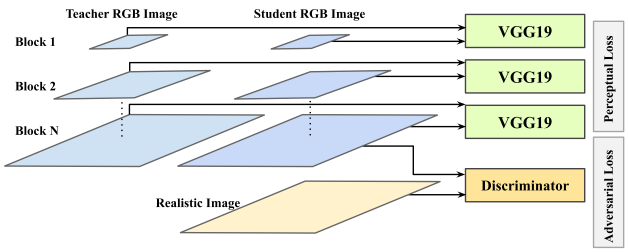

Figure 2. Multi-scale perceptual loss in distillation.

Table 1. Comparison of model complexity.

Model	Image Size	#Params (M)	FLOPs (G)
StyleGAN	1024	33.17	74.3
MobileStyleGAN	1024	8.01	30.2
BlazeStyleGAN	1024	2.07	4.70
BlazeStyleGAN	512	2.05	1.57
BlazeStyleGAN	256	2.01	1.28

4. Distillation

We train the student BlazeStyleGAN by distilling the teacher StyleGAN model, using a multi-scale perceptual loss and an adversarial loss.

Multi-scale Perceptual Loss. The UpToRGB transformation blocks output an RGB map pyramid with coarse-to-fine resolutions (from 8×8 to the target output resolution), as shown in Fig. 2. We resize the output from the teacher model to the same resolutions and calculate the perceptual loss, formulated as Eq. 1.

$$L_p^l(I_s, I_t) = \sum_l (\|\mathrm{VGG}(I_t^l) - \mathrm{VGG}(I_s^l)\|_2), \tag{1}$$

where I_s^l and I_t^l denote the images generated by student and teacher models at level l, respectively. VGG represents the multi-scale feature extractor using VGG19 [10] backbone.

Adversarial Loss. We also use the adversarial loss for training the student model. The loss of the student generator is

$$L_g(\mathrm{A}, \mathrm{B}) = f(-D(G_s(\mathrm{A}, \mathrm{B}))), \tag{2}$$

where G_s is the student generator, D represents the discriminator network, f is a non-saturating function, and A and B represent style and noise, respectively.

Our discriminator loss is defined as

$$\begin{aligned} L_d(\mathrm{A}, \mathrm{B}) = f(-D(\mathrm{real\_image}) + f(D(G_s(\mathrm{A}, \mathrm{B}))) \\ + \frac{\gamma}{2}\mu(\|\nabla D(G_s(\mathrm{A}, \mathrm{B})\|), \end{aligned} \tag{3}$$

where the third regularization term penalizes the discriminator from deviating from the Nash Equilibrium [17].

Instead of using the images generated by the teacher model, we use real images to train the discriminator. That design is motivated by the observation that the teacher model may generate images with strong artifacts, which can mislead student models in the distillation. Our experiments show that the student model can suppress the artifacts from the teacher model and improve the quality when training the discriminator with real images.

Overall training objective The final training objective is defined as a combination of the multi-scale perceptual loss and adversarial loss, represented as Eq. 4 with hyper-parameter weights λ_1, λ_2, and λ_3,

$$L = \lambda_1 \times L_p + \lambda_2 \times L_g + \lambda_3 \times L_d. \tag{4}$$

The weighting hyperparameters are tuned in the experiments for the best performance.

5. Experiments

We use the FFHQ [12] dataset for model training and evaluation. For the teacher model, we re-implement StyleGAN2 [13] using TensorFlow 2.0, and train the model on FFHQ 1024×1024. It has a FID score of 2.92, which approximately matches the official implementation[1]. We train our BlazeStyleGAN models at multiple resolutions. All models are trained on NVIDIA A100 GPUs with a batch size of 32, using the Adam optimizer [14] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and 800K training steps.

Table 1 summarizes the model complexity in terms of the number of parameters and FLOPs. Our BlazeStyleGAN significantly reduces the model complexity. Compared with MobileStyleGAN [2], our model has only 26% parameters and 16% FLOPS.

To benchmark BlazeStyleGAN's performance on mobile devices, we convert the BlazeStyleGAN model to TensorFlow Lite format, and run inference on various mobile devices. Each benchmark reports the average time of 50 iterations of inference on a device. As shown in Table 2, both BlazeStyleGAN-256 and BlazeStyleGAN-512 achieve real-time performance on all GPU devices in the benchmark. It can run in less than 10 ms runtime on high-end cellphones' GPU. BlazeStyleGAN-256 can also achieve real-time performance on the iOS devices' CPU. The bold numbers represent the real-time performance in the benchmark.

Fig. 3 shows the images generated from the teacher StyleGAN and our BlazeStyleGAN at 256×256 and 512×512 resolutions. Overall BlazeStyleGAN generates images with similar visual quality to the teacher model, despite minor facial detail loss caused by model compression. Some results demonstrate the student BlazeStyleGAN suppresses the artifacts from the teacher model in the distillation.

[1] https://github.com/NVlabs/stylegan2

Table 2. Benchmark of inference time (ms) on various high-end mobile devices.

Model	Chip	iPhone 11	iPhone 12	iPhone 13 Pro	Pixel 6	Pixel 7	Galaxy S10	Galaxy S20
BlazeStyleGAN-256	CPU	**23.02**	**18.87**	**16.05**	43.14	37.83	52.83	40.23
	GPU	**12.14**	**11.99**	**7.22**	**12.24**	**17.00**	**17.01**	**8.95**
BlazeStyleGAN-512	CPU	38.39	37.33	**29.46**	67.57	62.23	87.50	56.89
	GPU	**18.46**	**15.66**	**9.29**	**16.71**	**21.81**	**24.05**	**14.14**

Figure 3. Generated images of teacher StyleGAN and our BlazeStyleGAN at 256×256 (top two rows) and 512×512 (bottom two rows).

Table 3. FID scores of teacher StyleGAN and BlazeStyleGAN.

	Teacher StyleGAN	BlazeStyleGAN
FID-256	6.64	9.94
FID-512	4.33	8.96

The FID scores are reported in Table 3. Our BlazeStyleGAN is able to preserve the generation quality from the teacher StyleGAN. Specifically, comparing to the teacher StyleGAN-256 achieves FID score as 6.64 and teacher StyleGAN-512 has 4.33, BlazeStyleGAN improves FID to 9.94 and 6.64 for the resolution of 256 and 512 respectively.

6. Conclusions

In this work, we present the first StyleGAN model (BlazeStyleGAN) that can generate high-quality face images in real-time on most high-end smartphones. Efficient on-device generative models are an early and open research topic with a lot of challenges. We design an efficient architecture for the StyleGAN synthesis network, and optimize distillation strategy to mitigate artifacts from the teacher model. By significantly reducing the model complexity, our BlazeStyleGAN can achieve real-time performance on mobile devices in our benchmark.

References

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

[2] Sergei Belousov. MobileStyleGAN: A lightweight convolutional neural network for high-fidelity image synthesis. *arXiv preprint arXiv:2104.04767*, 2021. 1, 2, 3

[3] Ting-Yun Chang and Chi-Jen Lu. TinyGAN: Distilling BigGAN for conditional image generation. In *ACCV*, 2020. 1

[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 *AAAI*, 2020. 1

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 1

[6] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1

[7] 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. *NIPS*, 30, 2017. 1

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

[9] 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. 2

[10] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 3

[11] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021. 1, 2

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 2, 3

[13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 1, 2, 3

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[15] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. GAN compression: Efficient architectures for interactive conditional gans. In *CVPR*, 2020. 1

[16] Yuchen Liu, Zhixin Shu, Yijun Li, Zhe Lin, Federico Perazzi, and Sun-Yuan Kung. Content-aware GAN compression. In *CVPR*, 2021. 1, 2

[17] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 3

[18] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. GAN slimming: All-in-one GAN compression by a unified optimization framework. In *ECCV*, 2020. 1, 2

[19] Guodong Xu, Yuenan Hou, Ziwei Liu, and Chen Change Loy. Mind the gap in distilling StyleGANs. In *ECCV*, 2022. 2