Style-Based Generative Adversarial Networks

DILEEP KUMAR LATCHIREDDI
dlatchir@gmu.edu

RUSHIL DESETTY
rdesetty@gmu.edu

YASHWANTH REDDY GUNDEPALLY
ygundepa@gmu.edu

The key motivation for selecting this topic on implementing StyleGAN is the argument that StyleGAN generates high-quality images while producing various style combinations of the generated images. Our motivation is also upon a zeal to understand the functionality of Generative Adversarial Networks, and working on this project is an excellent way to comprehend StyleGAN. Our project implementation depicts the findings we found most intriguing about StyleGAN. The image generation by progressive growth gives stable training and high-quality images steadily. With the concurrency of AdaIN and the Mapping Network, we can establish style mixing.

Style-Based Generative Adversarial Networks [3]

1 INTRODUCTION

The quality of images generated by Generative Adversarial Networks has seen rapid growth in recent years. The generators are usually considered the source for constructing a collection of high-quality images from a simple random input. One of the most prevalent GAN architectures is StyleGAN, which is well proficient than other GAN models by contrasting the image content and style of the image. The StyleGAN is capable of generating high-quality images with unprecedented realism and detail.

However, despite its success, StyleGAN still faces several challenges. One of the primary challenges is the elevated computational cost required to train the model when generating high-quality images. Another challenge is the potential for producing biased or unpleasant content due to distribution bias in the training data. Finally, overfitting is risky, leading to inferior quality or unrealistic generated images. Apart from these, there are issues concerning the quality of the images generated, which are shown further ahead in the paper.

In our implementation, we re-implement the Style-based Adversarial Network to better understand the StyleGAN and to find out the improvements that can enhance the performance of the StyleGAN while determining the factors that are causing the existing StyleGAN to flounder.

We implement the generator architecture in a way that reveals novel controls for the image generation process, which is inspired by style transfer literature [1]. The "style" of the image is adjusted at each convolution layer depending on the latent code in our generator, which starts with a constant learning input and directly controls the strength of image characteristics at various scales. This architecture allows for intuitive scale-specific mixing and interpolation operations while also enabling automatic, unsupervised separation of high-level attributes (such as pose and identity) from stochastic variation (such as freckles and hair) in the generated images.

The generator transforms the latent input code into an intermediate latent space, which significantly alters how the network's components of variation are visualized. We can portray that some degree of entanglement is inevitable because the latent input space must adhere to the probability density of the training data. That restriction is absent from our intermediate latent space, allowing for its detachment. We suggest two new automated measures, perceptual path length, and linear separability, for measuring these features of the generator since earlier techniques for determining the level of latent space disentanglement are not directly applicable in our instance.

The StyleGAN we implemented was inspired by the framework of ProGAN [2] (ProgressiveGAN). The ProGAN trains the generator network and the discriminator network over time in a hierarchical fashion. High-resolution images can be delivered using this method, with each hierarchical level of the network learning to provide increasingly precise features of the output image. Highly realistic images of faces, creatures, and terrains can be produced using PROGAN, among other things. In general, PROGAN outperforms traditional GANs in the context of picture-generating tasks.

The concept of progressive training has various advantages. In the primitive stages, the generation of low-resolution images is substantially more stable because there is more undersized class information. As the resolution increases, we let the network train for a more prominent latent vector. In the trial, it

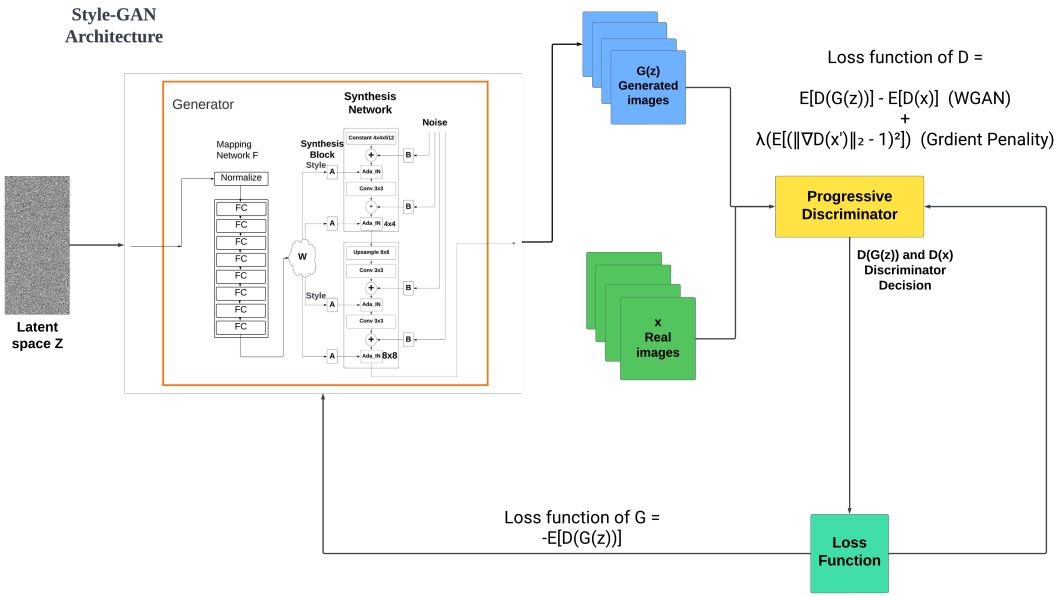

Fig. 1. StyleGAN Architecture

quickens and stabilizes the training adequately to synthesize high-resolution images using WGAN-GP loss dependably.

Finally, we utilized a brand-new high-resolution human face dataset (Flickr-Faces-HQ, FFHQ) that delivers remarkably more suitable quality and covers a tremendous range of variance.

2 DETAILS OF THE APPROACH

This section describes the implementation of StyleGAN . Fig 1 depicts the architecture of StyleGAN. StyleGAN is a state-of-the-art generative model that generates high-quality images with unprecedented realism and detail. The StyleGAN consists of a generator, and discriminator networks, which are both trained in a progressive fashion. The individual components of those networks define the quality of images generated. The final architecture of the proposed SyleGAN comprises several techniques and functions embedded in the network to generate the required images.

2.1 Dataset

We utilized the FFHQ (Flickr-Faces-HQ) dataset, which consists of 70,000 human face images with variations in age, image background, and ethnicity at 1024x1024 resolution. Due to restrictions on our resources, we used the 256x256 resolution of those high-quality images for the implementation. This dataset was employed because it was initially created as a benchmark for evaluating Generative Adversarial Networks(GAN).

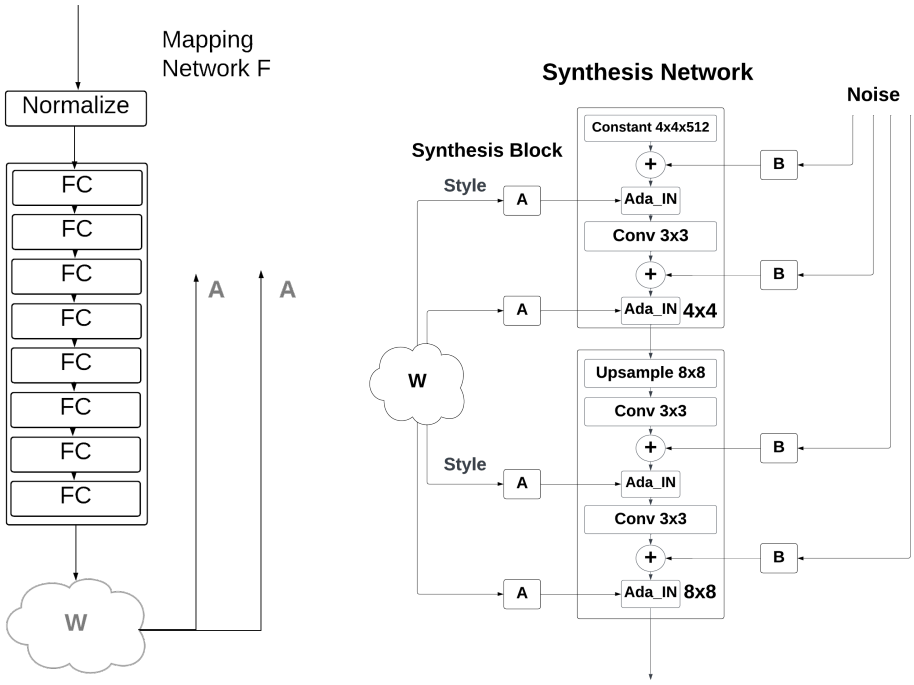

Fig. 2. Mapping Network Architecture (Left) and Synthesis Network Architecture (Right)

Input : The input for a StyleGAN is a random vector called a "latent vector" or "latent code". This latent vector is sampled from a normal distribution, typically with a mean of zero and a standard deviation of one.

Hyper-parameters : Some of the hyperparameters used are learning rate, batch size, latent noice dimensions, generator layers, discriminator layers

2.2 Mapping Network

The mapping network is one of the networks in StyleGAN that maps a latent code Z to an intermediate latent space W, which is used as an input to the generator. Figure 2 depicts the architecture of the mapping network. The mapping is a continuous set of fully connected linear layers with ReLU activations. Continuous mapping in the latent space W is chosen because it does not require sampling from a fixed distribution. Initially, the latent code Z is normalized using the Pixel Norm. The Pixel Norm is a pixel-wise normalization technique where normalization is applied at each pixel of the latent code. The normalized latent code is passed through 8 fully connected linear layers to obtain a latent space W. The continuous mapping implicitly induces a sampling density in W and ‘unwraps’ the space, promoting a disentangled representation of factors of variation. This leads to a more linear representation of factors of variation, making the realistic image generation process more straightforward than the entangled representation. This technique is unsupervised, and the W space is expected to be less entangled than the Z space. The mapping network plays an essential role in style generation by controlling the variation

and style of the generated images, which enables the generator network to generate high-quality and diverse images.

Algorithm 1 Style GAN

Define the generator and discriminator architectures
for n epochs **do**
1. Progressively increase the image size
2. Train the mapping network
3. Train the generator
4. Calculate and update loss

2.2.1 Control of Variation. One of the capabilities of a mapping network is to provide the generator with a means to generate a wide variety of images with distinct styles and characteristics. It is observed that we can control facial emotions, facial tone, accessories, and backgrounds of the generated emotions by altering the intermediate latent code. Moreover, the mapping network allows us to control the randomness of the generated image. By allowing randomness, we can balance the concept of realism and creativity of the images.

2.3 Synthesis Network

The synthesis network is the StyleGAN generator network that creates an image from a latent code intermediate. Figure 2 (Right) depicts the architecture of the mapping network. The generator network comprises a set of convolutional layers with progressively higher spatial resolution, followed by upsampling layers and skip connections, a fade-in layer that aids in maintaining the generated image's structure and finer features.

The synthesis network is created to produce images of the highest quality, with realistic details and precise control over their styles and traits. Progressive growth, constant input, and adaptive instance normalization (AdaIN) are three unique strategies embedded into our StyleGAN Synthesis Network.

2.3.1 Noise Injection. The noise injection layer is used as a form of regularization to prevent overfitting in a neural network. It takes an input and adds Gaussian noise to it by first creating a tensor of random noise samples called noise with the same shape as the input. It then multiplies noise element-wise with the parameter of 4 Dimensional that is of shape [1,1,in_channels,1] to control the magnitude of the noise.

2.3.2 Progressive Growing. The synthesis network in StyleGAN uses a progressive growth approach, which means that it starts with a small spatial resolution and gradually increases it as the network goes deeper. This allows the network to capture both global and local features of the image, resulting in high-quality and detailed output while reducing overfitting.

While performing progressive growth, there is upsampling at every block except the first, each block has two convolutional networks with leaky ReLU activation, two Adaptive Instance Normalizations, and two noise vectors of input channel dimensions injected into the block before being fed to the Normalization.

The generator and discriminator both begin with the extremely small 4 by 4 image. To train the model, the original photos are resized to 4 by 4. Training will be quick because these photos are relatively small. Once the discriminator network has received enough 4 by 4 images, we will gradually add more layers to create 8 by 8 and 16 by 16 images until the image resolution reaches 256 by 256. The image was doubled in size and cut in half using bilinear interpolation and average pooling, respectively. By gradually adding new layers, a 4 by 4 network was transformed into an 8 by 8 network. This is known as fade-in new layers.

For the fade-in layers demonstration, initially, 4*4 images will be used to train the network. We will gradually introduce new layers once we have trained it on a large enough sample of photos. While average pooling is used in the discriminator to downsample the image size, bilinear filtering is used in the generator to upsample the image.

A residual block for 8*8 resolution is now introduced in order to expand the network gradually. This new block is introduced gradually during training instead of immediately. In the generator network, this block comprises two convolutional layers and one upsampling layer. In the discriminator network, there are two convolution layers and one average pooling layer. The old (4*4) block is multiplied by (1-α), and the new block is multiplied by α, where the value rises from 0 to 1 linearly. All prior layers in the model will continue to be trainable even after the fading of new layers.

Similarly to this, more layers will be gradually added if you wish to make an image with better quality. The final layer of the generator receives a 1*1 convolution layer to transform into an RGB image. To obtain information from the generated image(RGB image), a 1*1 convolution layer is added on top of the discriminator network.

2.3.3 Adaptive Instance Normalization. StyleGAN uses AdaIN as a normalization technique to translate the style of an input image to a generated image. The feature distribution of the content image is matched with the style image by AdaIN by normalizing the feature maps of the content image using the mean and standard deviation of the style image. The strength of the style transfer is controlled by learning parameters that are applied to the normalized features in order to scale and shift them. AdaIN enables StyleGAN to produce images that not only represent the stylistic characteristics of the style image but also the semantic information of the input image. AdaIN performs style transfer (in the feature space) by aligning the first-order statistics (μ and σ) at no additional cost in terms of complexity. This helps in style transfer, where the style of one image is transferred to the content of another image. By applying AdaIN to the intermediate feature maps of a neural network, it is possible to control the style of the output image while preserving the content of the input image.

In conclusion, StyleGAN's synthesis network is a potent generator network that can create realistic details and high-quality images with granular control over their styles and traits. StyleGAN is one of the most cutting-edge GAN designs for image synthesis because of the usage of progressive growth, constant input, AdaIN, and other cutting-edge methods.

2.4 Discriminator

The discriminator in our implementation is identical to that of the Progressive GAN, which uses the WGAN-Loss function with a gradient penalty. WGAN-GP is a specific kind of generative model (Wasserstein Generative Adversarial Network with Gradient Penalty) that uses the Wasserstein distance to calculate the split between the distributions of the real and generated data. The Gradient Penalty is

added to enforce the Lipschitz continuity, which stabilizes the training process and raises the caliber of generated samples. The ability of WGAN-GP is to produce high-quality images and prevent mode collapse, a significant issue with GANs. Additionally, it enables the generator to be adjusted, increasing its adaptability to various datasets.

In addition, to avoid problems like mode collapse, where the generator generates similar images for all distinct types of noise vectors, we used mini batch standard deviation to add little deviation in the generated images. To implement the minibatch standard deviation, we first find the standard deviation of each feature in the activation map and then average them over the minibatch. The new activation maps acquired are concatenated at the rear of the discriminator network.

2.5 Style Mixing

Style mixing is a technique in StyleGAN that allows for combining different styles or attributes to generate an image. Style mixing involves taking two or more intermediate latent codes (also known as style vectors) and combining them to create a new intermediate latent code used to generate an image with a combination of the styles of two images.

Style mixing is performed by passing the intermediate latent codes of two images through the mapping network. The intermediate codes are separated into two parts, one representing the semantics of the image while the other representing the style of the image.

Next, a new style vector is produced by combining the style vectors of two images, typically via linear interpolation. The latent vector of a target image is then fused with this new style vector to produce a new intermediate latent code that is used to produce an image. While the style or attributes of the combined style vectors will be present in the final image, the semantic content of the target image will still be present.

By fusing the characteristics and styles of several images, the powerful method known as "style mixing" enables the development of incredibly imaginative and unique visuals. It can be used, for instance, to blend the facial features of one person with the color tone of another person or the hairstyle of one image with another's. In order to find unexplored and intriguing styles and qualities, style mixing can also be used to explore the latent space of the generator.

In overview, style mixing is a StyleGAN approach that enables the fusion of many styles or features in creating a picture. Style mixing is producing highly imaginative and unique images with precise control over their styles and attributes by merging the style vectors of two images and creating a new intermediate latent code.

3 RESULTS

3.1 Style GAN Results

In this section, we presented the results of our work. We used the FFHQ dataset (256 *256-pixel quality images) and ran our code for 100 epochs on a single-core GPU, with 80GB memory. It nearly took us 8-9 days to achieve these results. figure 3 depicts some of the generated images using style gan architecture

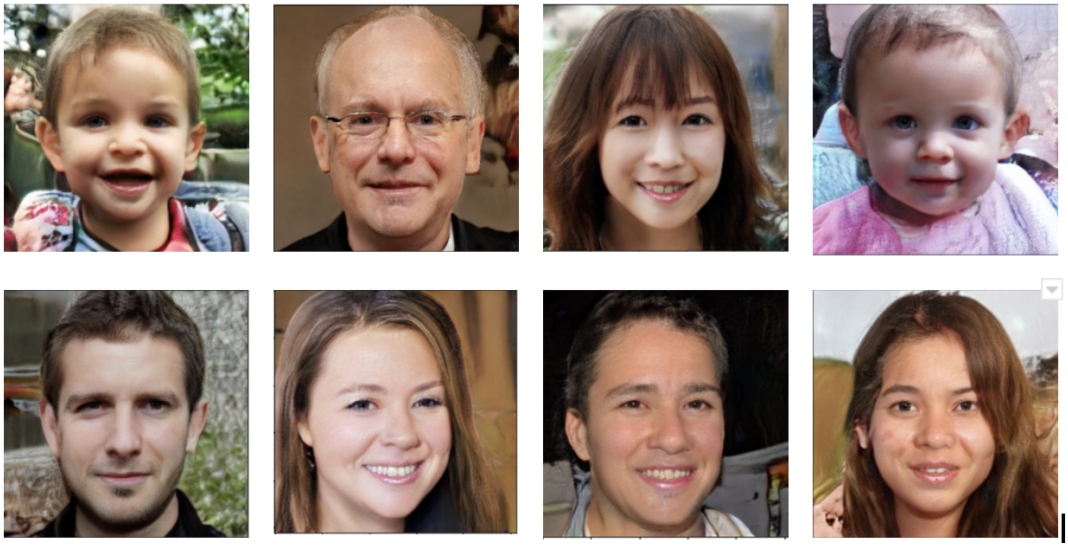

Fig. 3. StyleGAN generated Results

Fig. 4. StyleGAN Transfered Results

3.2 Style Transfer Results

The following results in Fig 4 depict the style Transfer results. This can be achieved by changing a few values in the latent vector. In picture 1 we can see the skin tone change, in picture 2 we can observe the change in the shade of the glasses nad in picture 3 we can observe in the fade of glasses

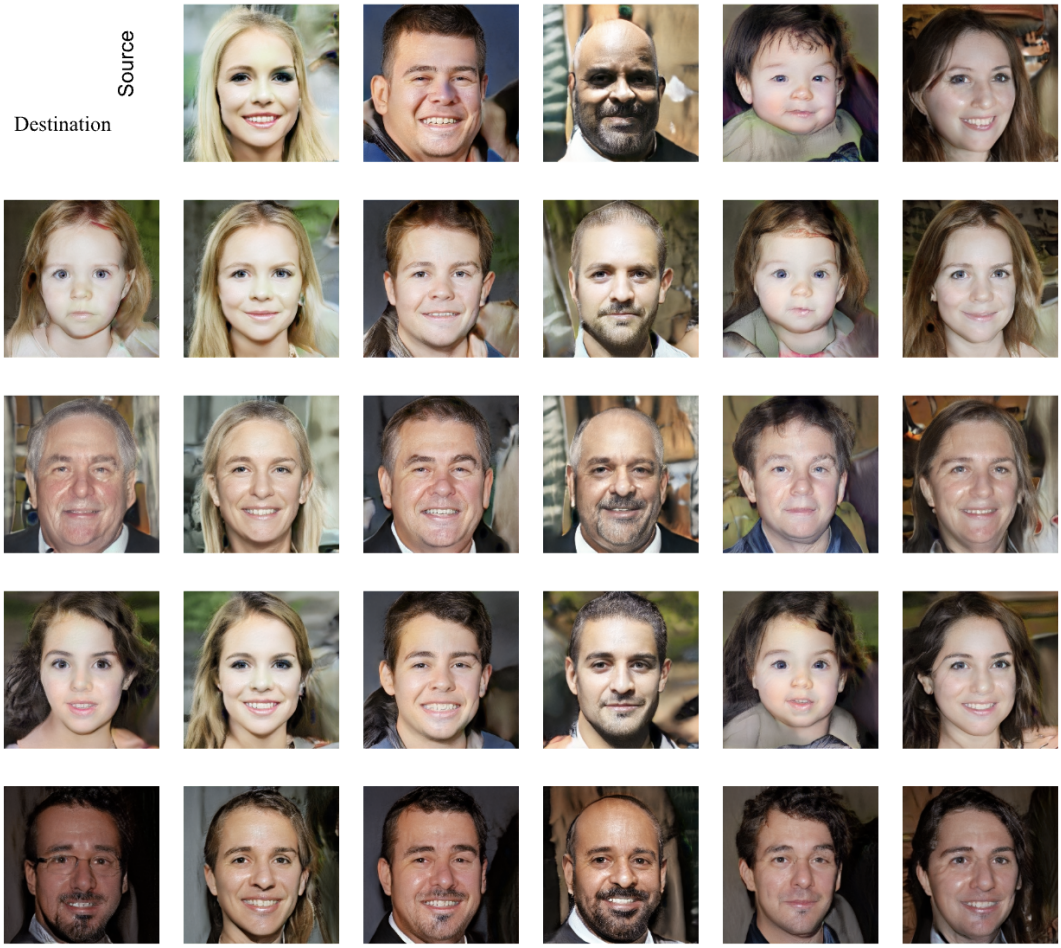

Fig. 5. StyleGAN Mixing Results

3.3 Style Mixing Results

In style mixing we mix two latent vectors in synthesis network. Before sending it to synthesis network we took the average of two latent weights. Fig 5 depicts us the results of style mixing. This is achieved by merging the intermediate latent codes of two images and creating a new intermediate latent code.

4 DISCUSSION CONCLUSIONS

- Based on our results and work on StyleGAN, we have analyzed various components that describe the functionality of StyleGAN. The findings from our implementation were that utilizing a progressive growth structure for StyleGAN helps to train with stability and generate high-quality images. At the primitive stages, the generator learns about the intricate detail of the image. As it progresses and increases its resolution, the model learns about the background and

contrast of the image. The images generated by using such an approach require a vast amount of data, and some other approach could be implemented that requires fewer data and can generate a wide variety of images thoroughly based on the randomness introduced.

- We also observed that increasing the number of layers in the generator could destabilize the training process.
- One can control the styling of the generated images by altering the intermediate latent code. When we alter the W value in the early step of progressive growth, changes are observed in the specific detailed content of the image. When we alter the W value at the rear, changes are observed in the background and the objects of the generated image.
- One big challenge in implementing StyleGAN was the computational cost. However, there were a few techniques like Adaptive Instance Normalization, which does style transfer without additional computation cost.
- While upsampling in StyleGAN, we experimented with bipolar interpolation and nearest-neighbor upsampling. In bipolar interpolation, the new pixel is the weighted average of the surrounding pixels, whereas the nearest neighbor duplicates each pixel to create a larger image. Although we implemented bipolar interpolation in our final proposed approach, the critical finding was the nearest neighbor approach is excellent when training in lower resolutions as it preserves the sharp details of the image. In contrast, bipolar interpolation preserves the image's smoothness, making it shrewd for training at higher resolutions.

5 STATEMENT OF INDIVIDUAL CONTRIBUTION

While implementing the StyleGAN we experimented with different structures and induced various hyperparameters and techniques to evaluate the model that could generate high-quality images. Each of the experiments that we processed was executed for about 7 days.

Both Yashwanth and Dileep worked collabratively on implementing style GAN and Rushil worked on implementation of Style mixing

5.1 Yashwanth Reddy

- He has worked on implementation of Generator, Mapping network, Synthesis network and AdaIN

5.2 Dileep Kumar

- He has worked on implementation of the discriminator and on Style Transfer and on training process

5.3 Rushil Desetty

- He has worked on Style Mixing, Report and aided in training process.

REFERENCES

[1] X. Huang and S. J. Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. (2017). https://doi.org/CoRR2017.

[2] Samuli Laine Jaakko Lehtinen Tero Karras, Timo Aila. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. (2018). https://doi.org/ICLR2018.

[3] Timo Aila Tero Karras, Samuli Laine. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. (2019). https://doi.org/CVPR2019.