Generating Images with StyleGan2

Hanna Foerster

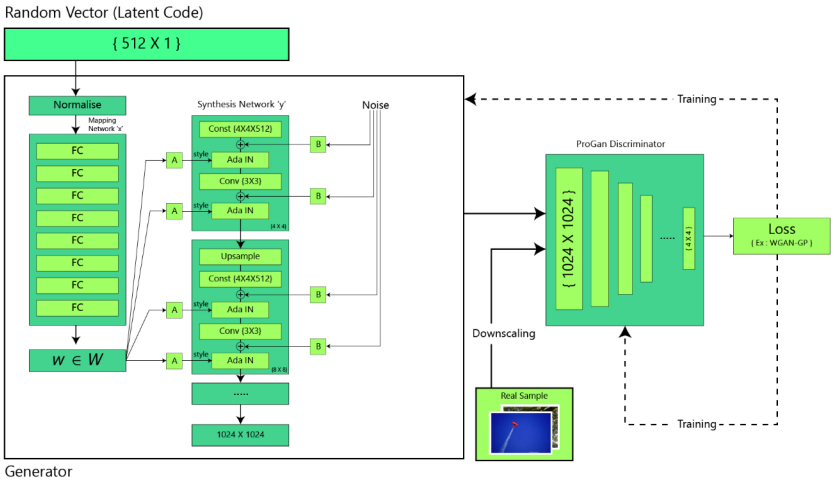
Abstract

This paper proposes using StyleGan2 to generate images. It first explains the structure and methodology of StyleGAN and then notes innovations introduced in StyleGAN2. Lastly StyleGAN2 is compared to StyleNAT and StyleGAN2 is determined superior. StyleGAN is a Generative Adversarial Network (GAN) that trains its model in an incremental process from low to high resolution. StyleNAT is a Transformer-based adaptation of StyleGAN, using attention layers instead of convolutions, but maintaining the incremental expansive process of the model.

1 Methodology

1.1 StyleGAN

Generative adversarial networks (GANs) are an efficient way to generate new high-quality images based on a dataset. GANs are made up of a generator and discriminator model, where the generator model learns how to generate new images from some noise input and the discriminator learns to differentiate between fake images and the real images from the dataset. By passing back to the generator a score of how real the fakely generated images look, the generator can continuously improve to generate better images that look more real, and the discriminator can continuously improve in detecting fake images. This cycle produces high quality generated samples. StyleGAN [2] is based on the idea of progressive growth where the models train from small to large images in an incremental expansive process. For this, generator and discriminator are first fit to smaller resolutions until stable and gradually upscaled in resolution via adding new blocks. The exact architecture of the generator can be seen below:



Style GAN architecture [1]

StyleGAN takes a latent code z as input and a mapping network f (blocks of FC) transforms this into an intermediate latent code w. A style vector is then produced by affine transforms (see blocks A) by this mapping network. This gives us $\boldsymbol{y} = (\boldsymbol{y}_\mathrm{s}, \boldsymbol{y}_\mathrm{b})$ as output. This is integrated into the generator model design via a layer called adaptive instance normalization

(see block Ada IN), which is the controlling style factor in StyleGAN. It takes as input the style vector $\boldsymbol{y}$ and the output of the convolutional layer $\boldsymbol{x}$.

$$AdaIN(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{y}) = \boldsymbol{y}_{\mathrm{s,i}}((\boldsymbol{x}_{\mathrm{i}} - \mu_{\mathrm{i}})/\sigma_{\mathrm{i}}(\boldsymbol{x}_{\mathrm{i}})) + \boldsymbol{y}_{\mathrm{b,i}} \tag{1}$$

At the top of the synthesis network an input of 4*4*512 can be seen. Unalike to other progressive GANs, StyleGAN does not take a random input to create the initial latent, but rather replaces this with a constant initial matrix of 4*4*512. Under this constant, AdaIN and a convolutional layer can be found. Together they comprise a style block. The Upsample block that can be seen under this is constructed by bilinear sampling. A low pass filter filters the activation with a binomial filter after each up and downsampling layer. Noise layers (see block B) are added for randomness i.e. variations in output, stochasticity. These are added to each activation map (output of the convolutional layer) before the AdaIN operations. Lastly, as described above, an intermediate latent w is used for style generation. To avoid training on the correlation between layers, two intermediate latent vectors w and w' are produced from two initial latent vectors that are switched at a random point in training. Feeding different intermediate latents w, w',... to different layers also gives StyleGan the ability to mix styles easily.

1.2 StyleGAN2

StyleGAN2 [3] is StyleGAN's successor to ensure higher image quality. Modifications in StyleGAN2 include the shortening of the StyleBlock, path length regularization and an alternate network architecture to progressive growing. A style block in StyleGAN consists of a convolutional layer and AdaIN. In detail, modulation is implemented with the input from the affine transform of the latent vector w and then passed through the convolutional layer which is then normalized. In StyleGan2 this has been modified. The modulation which scales each input feature map of the convolution on style can be implemented as scaling (scale s) convolution weights:

$$w'_{\mathrm{ijk}} = s_{\mathrm{i}} * w_{\mathrm{ijk}} \tag{2}$$

After modulation and convolution, the output activations have standard deviations of

$$\sigma_{\mathrm{j}} = \sqrt{\sum_{i,k} w'^{2}_{\mathrm{ijk}}} \tag{3}$$

The final normalization operation's goal is to restore the outputs to standard normalized units. This can be achieved by scaling the output feature map by sigma or by just adding this equation to the convolution:

$$w''_{\mathrm{ijk}} = w'_{\mathrm{ijk}} / \sqrt{\sum_{i,k} w'^{2}_{\mathrm{ijk}} + \epsilon} \tag{4}$$

In this way we now have reduced modulation before convolution and normalization after convolution to modulation (2) and demodulation (4) before the convolution step. According to the researchers this also reduces artifacts in the generated images.

In StyleGAN2 the researchers also use the fact that a correlation can be found between Perceptual Path Length (PPL) and image quality. PPL indicates the perceptual smoothness of a latent space and regularizing this forces the generator to minimize changes in the latent variables as much as possible. Researchers found that regularizing the path length in this way leads to more reliable and consistently behaving models.

One last big change that was implemented in StyleGAN2 is the change of the progressive growing structure because this was seen as a cause for artifacts. The artifacts ensued due to the generator having a strong location preference for detail. Instead, skip connections are now used in the generator and residual networks in the discriminator. The aspect of focusing on low-resolution features first and then shifting attention to finer details that we have with progressive growing can be preserved by these architectures.

1.3 SyleNAT

Initially I wanted to submit my assignment using images generated with StyleNAT [8]. This is a new adaption (last quarter 2022) of StyleGAN using Transformers that beat the state of the art in the generation of images for ffhq 256*256 according to Papers with code. Here I will give a short explanation on the differences and in the limitation section, I will explain why I did not submit images generated by StyleNAT. The idea of StyleNAT is to replace convolutional layers in StyleGAN with attention layers. This layer (called Hydra-Neighborhood Attention) is made up of attention heads that span neighbourhoods of pixels in differing distances from the target pixel, thus capturing various receptive fields; finer details and the bigger picture. The transformer-based model has the advantage of being able to capture a global receptive field using self attention instead of only a local receptive field that is perceived by the convolutional model. With the partitioned attention model some heads can pay attention to the local receptive field and some can pay attention to the the midfield, while some can attend the global field.

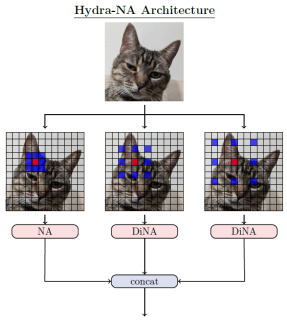
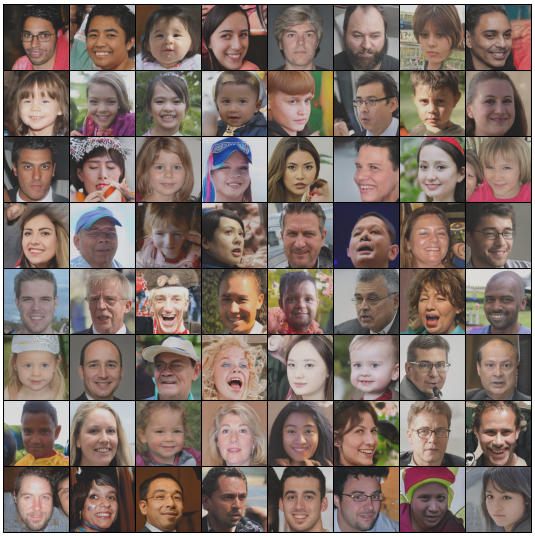


Figure 1: Hydra-NA from [8]

2 Results

I have implemented both StyleGan2 and StyleNAT, where basecode from [5] [6] [4] [7] is used and adapted to run in one jupyter notebook. (I have attached the StyleNAT code below the StyleGAN code.) I have trained both StyleGAN and StyleNAT for roughly 700000 epochs. The following are results when a random batch of samples are picked from StyleGAN2:



Most samples can be identified as human faces, where some are so good that you can not differentiate them to real human face, while others still lack the right background context. For example the neck area might look unrealistic or a part of the face might have an artifact. One can see a few images where heads are surrounded by unrealistic things and heads that have small parts cut off and background starts too soon. Overall the samples are quite diverse in gender, colour, age and context. The hats are not quite up to a realistic level, but more training time could have helped that. The glasses on the other hand look already quite realistic. Nevertheless, overall there are quite a few very realistic looking images.

The next results show images generated by linearly interpolating between points in the latent space:

And here are some cherry-picked samples that show the best outputs StyleGAN2 has generated:

For comparison here are some cherry-picked samples from StyleNAT. One can see that even the best images here have a painting-like quality rather than a picture-like quality.

3 Limitations

There were some limitations in generated images that were mostly caused by the limit I had in the scope of the assignment. This includes having to implement everything in one jupyter notebook file, not having enough GPU RAM, not having the right environment for some code (such as some code with pytorch version and GPU requirements). Specifically, in StyleGAN2 I was not able to use the 2d convolution function that supports higher gradients and so had to use the torch convolution function that does not support higher gradients. Furthermore, there were some functions (Upsample, Downsample, Blur, etc.) that had an underlying C code base dependence. Since we could only submit one jupyter notebook file, I exchanged these with functions that were written in python, which drastically slowed down the operations. In case of StyleNAT the main limitation was RAM. Since an extra library for neighborhood attention used torch and cuda, the RAM was filled up quickly and I had to reduce the batch size from 8 to 4. Here I also had the same C code dependency, which made the code slower as well. All of this meant that the produced samples by StyleNAT were just not good enough compared to my samples from StyleGAN2, even after running for 720000 epochs, although in the paperwithcodes website the generated image scores were ranked top for dataset ffhq 256*256.

In the sample from StyleGAN2, there are still limitations, which could be attributed to training time and the parts of the code that I had to adapt as mentioned above, since they are not existent in the proposed paper.

Bonuses

This submission has a total bonus of 0 marks, as it is trained only with a GAN (-8), but the images are 128*128 pixels (+8) from the ffhq dataset.

References

[1] @pawangfg. *StyleGAN - Style Generative Adversarial Networks*. URL: https://www.geeksforgeeks.org/stylegan-style-generative-adversarial-networks/. (accessed: 04.02.2023).

[2] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: (2018). DOI: 10.48550/ARXIV.1812.04948.

[3] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: (2019). DOI: 10.48550/ARXIV.1912.04958.

[4] NVlabs. *Official Pytorch implementation of StyleGAN3*. URL: https://github.com/NVlabs/stylegan3. (accessed: 04.02.2023).

[5] NVlabs. *StyleGAN2-ADA - Official Pytorch implementation*. URL: https://github.com/NVlabs/stylegan2-ada-pytorch. (accessed: 04.02.2023).

[6] rosinality. *Implementation of Analyzing and Improving the Image Quality of StyleGAN (StyleGAN 2) in PyTorch*. URL: https://github.com/rosinality/stylegan2-pytorch. (accessed: 04.02.2023).

[7] SHI-Labs. *New flexible and efficient image generation framework that sets new SOTA on FFHQ-256 with FID 2.05*. URL: https://github.com/SHI-Labs/StyleNAT. (accessed: 04.02.2023).

[8] Steven Walton et al. "StyleNAT: Giving Each Head a New Perspective". In: (2022). DOI: 10.48550/ARXIV.2211.05770.