Indian Institute of Science

Advanced Image Processing-2022

Comparison of StyleGAN & StyleGAN2

Author:
Sangeeta Yadav
CDS (15318)

April, 2021

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Goal of the project

We attempt to understand and analyze recently proposed Style based image generators, popularly known as StyleGANs [Karras, Laine, Aittala et al. 2020,[Karras, Laine and Aila 2018].

1.2 Main Contributions

- Compared the network architecture of StyleGAN and StyleGAN2
- Proposed error metrics to quantitatively compare these two styleGANs
- Have validated the improvements of StyleGAN2 over StyleGAN
- Implemented the error metrics from scratch

2 StyleGAN

The Style Generative Adversarial Network, or StyleGAN for short, is an extension to the GAN architecture that proposes large changes to the generator model, including the use of a mapping network to map points in latent space to an intermediate latent space, the use of the intermediate latent space to control style at each point in the generator model, and the introduction to noise as a source of variation at each point in the generator model. It yields state-of-the-art results in data-driven unconditional generative image modeling. Its network architecture is shown in figure (2).

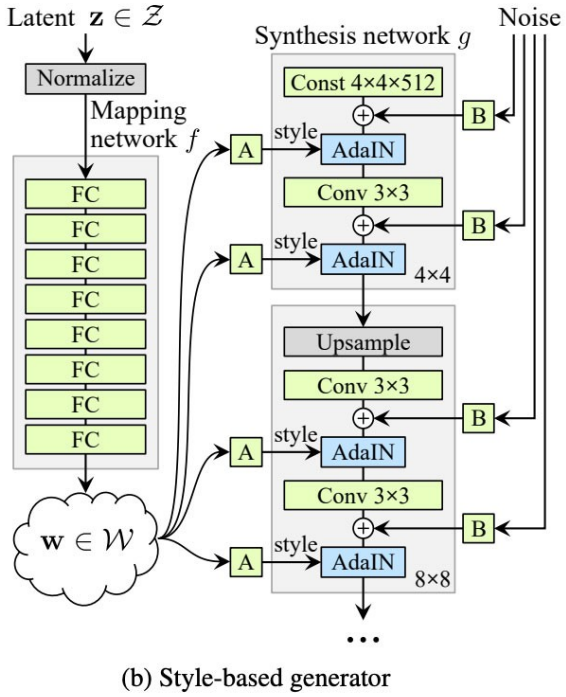

Figure 1: StyleGan: Network architecture

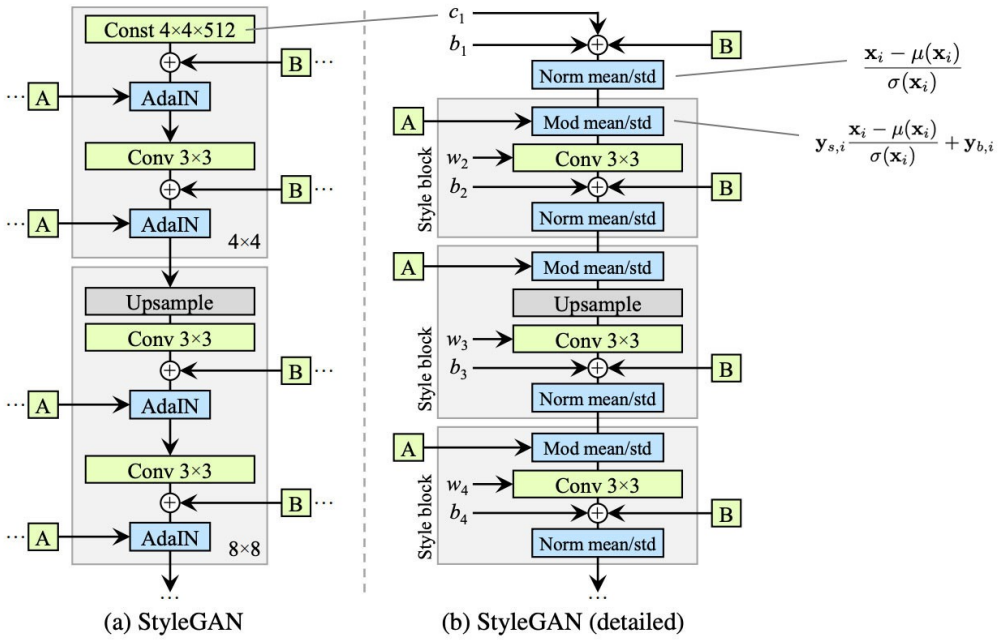

Figure 2: StyleGan: Detailed Network architecture

StyleGAN brings major changes into the generator thus allowing for control over generated images. Below are some salient architectural tweaks done in StyleGAN as compared to a regular GAN

2.1 Salient Architectural Features

- **Baseline Progressive GAN.**

  The StyleGAN generator and discriminator models are trained using the progressive growing GAN training method. This means that both models start with small images, in this case, 4×4 images. The models are fit until stable, then both discriminator and generator are expanded to double the width and height (quadruple the area), e.g. 8×8. A new block is added to each model to support the larger image size, which is faded in slowly over training. Once faded-in, the models are again trained until reasonably stable and the process is repeated with ever-larger image sizes until the desired target image size is met, such as 1024×1024.

- **Addition of tuning and bilinear upsampling.**

  Instead of transpose convolutional layers that are common in other generator models, StyleGAN uses nearest neighbor layers for upsampling

- **Addition of mapping network**

  Next, a standalone mapping network is used that takes a randomly sampled point from the latent space as input and generates a style vector. The mapping network is comprised of eight fully connected layers, e.g. it is a standard deep neural network.

- **Removal of latent vector input to generator.** This involves modifying the generator model so that it no longer takes a point from the latent space as input. Instead, the model has a constant 4x4x512 constant value input in order to start the image synthesis process.

- **Addition of noise to each block.**

The output of each convolutional layer in the synthesis network is a block of activation maps. Gaussian noise is added to each of these activation maps prior to the AdaIN operations. This noise is used to introduce style-level variation at a given level of detail.

- **Addition Mixing regularization**

  Instead of one style vector images a fraction of images are generated using two style vectors, seemingly mimicking to mix different styles across images.This encourages the layers and blocks to localize the style to specific parts of the model and corresponding level of detail in the generated image.

- **Pixel Norm**

  Instead of using batch normalization, as is commonly done, the authors used pixel normalization. This "pixelnorm" layer has no trainable weights. It normalizes the feature vector in each pixel to unit length, and is applied after the convolutional layers in the generator. This is done to prevent signal magnitudes from spiraling out of control during training.

$$b_{x,y} = \frac{a_{x,y}}{\sqrt{\frac{1}{C}\sum_{j=0}^{C} a_{x,y}^{j} + \epsilon}} \tag{1}$$

2.2 Loss

The Wasserstein Generative Adversarial Network, or Wasserstein GAN, is an extension to the generative adversarial network that both improves the stability when training the model and provides a loss function that correlates with the quality of generated images.

2.3 Limitations of StyleGAN

- Blob-like and droplet like artifacts
- Artifacts related to progressive growing
- FID, Precision and Recall were proposed in StyleGAN for quality check. Both FID and P &R are based on classifier networks that have recently been shown to focus on textures rather than shapes, and consequently, the metrics do not accurately capture all aspects of image quality.

3 StyleGAN2

StyleGAN2 model is composed of a mapping function (Z to W), affine transformations (W to S), feature convolution layers, and tRGB convolution layers that transform feature maps to RGB images. Its network architecture is shown in Fig. 3

3.1 Improvements done in StyleGAN2

- In StyleGAN2, the generator is regularized with the path length. This makes it easier to attribute a generated image to its source
- **Weight Demodulation**

  StyleGAN uses adaptive instance normalization to control the influence of the source vector w on the resulting generated image. In the second version of StyleGAN, the authors restructure

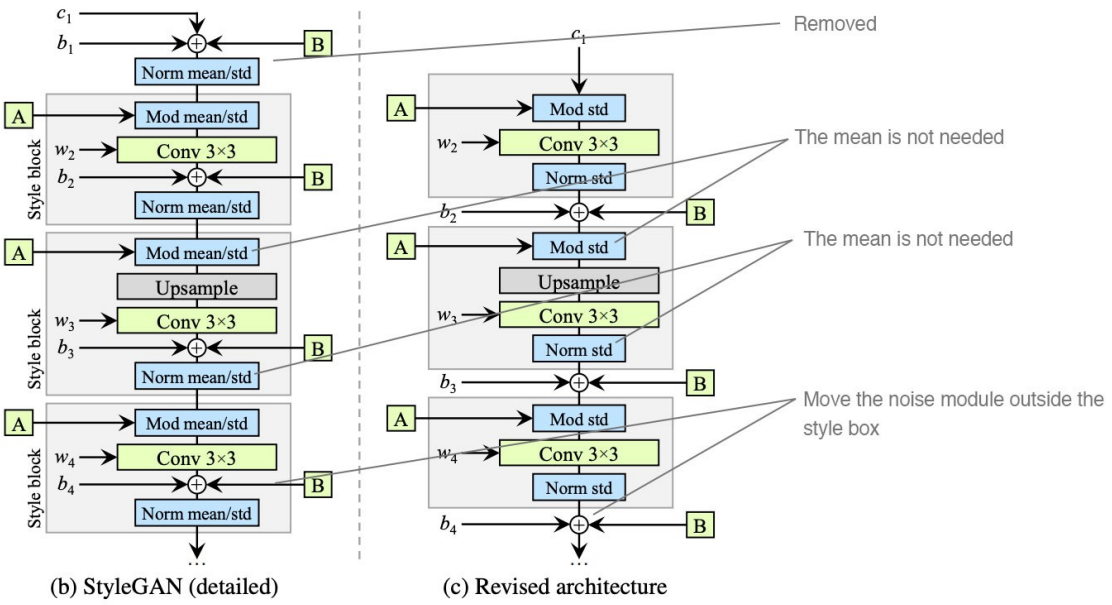

Figure 3: StyleGan2: Network architecture

the use of Adaptive Instance Normalization to avoid these water droplet artifacts as explained in figure 4

- **Removing Progressive Growing**

  StyleGAN images have a strong location preference for facial image features like noses and eyes. The authors attribute this to progressive growing. Progressive growing describes the procedure of first tasking the GAN framework with low-resolution images such as 4^2 and scaling it up when a desirable convergence property has been hurdled at the lower scale.

- **Path Length Regularization**

  StyleGAN2 introduces a new normalization term to the loss to enforce smoother latent space interpolations. Latent space interpolation describes how changes in the source vector z results in changes to the generated images.This is done by adding the following loss term to the generator:

$$\mathbb{E}_{w,y\sim\ \mathcal{N}}(||J_w^T y||_2 - a)^2$$

  Better existing distribution quality metric

- **Deepfake Detection via Projection**

  One application of projection would be to use it to find the source of a given generated image to tell if it is real or fake, the idea being that if the image is fake then, we can find the latent vector that produced it otherwise we cannot.

- **Perceptual Path Length (PPL)**

  We observe that the perceptual path length (PPL) metric, originally introduced as a method for estimating the quality of latent space interpolations, correlates with consistency and stability of shapes.

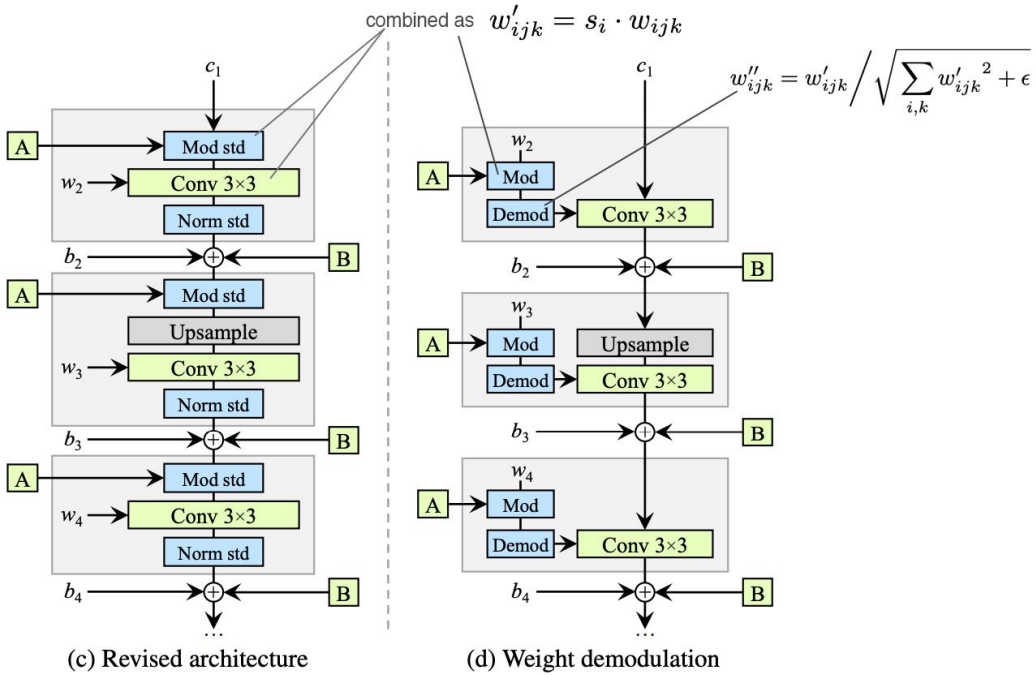

Figure 4: Weight Demodulation

4 Analysis: Quality assessment of StyleGAN and StyleGAN2

Quantitative analysis of the quality of images produced using generative methods continues to be a challenging topic. We quantify the improvements of StyleGAN2 over StyleGAN in terms of following error metrics.

4.1 Fréchet inception distance (FID)

Particularly tailored to assess the quality of images created by GANs. Unlike Inception Score, which evaluates only the distribution of generated images, FID compares the distribution of generated images with the distribution of real images that were used to train the generator. To compute Frechet Inception Distance, we pass generated and true data through an ImageNet pretrained Inception V3 model to obtain visually relevant features. Let (M_t, C_t) and (M_g, C_g) represent the mean and covariance of the true and generated features respectively, then compute

$$FID = ||M_t - M_g||_2^2 + Tr(C_t + C_g - 2(C_t C_g)^{\frac{1}{2}}) \tag{2}$$

Write $g(z)$ for an image generator to be evaluated, y for a label, $p(y|x)$ for the posterior probability of a label computed using Inception V3 model on image x $p(y) = \int p(y|g(z))dz$ for the marginal class distribution, and $\mathcal{D}(p||q)$ for the KL-divergence between two probability distributions p and q. The Inception Score for a generator is

$$exp[\mathbb{E}_z \sim p(z)[\mathcal{D}(p(y|g(z))||p(y))] \tag{3}$$

[Chong and Forsyth 2019] proposed two unbiased and more effective error metrics namely $\overline{FID}_\infty$, $\overline{IS}_\infty$. We calculate both these metrics for StyleGAN and StyleGAN2 and the results are given in [1].

Figure 5: Images generated by StyleGAN

Figure 6: Images generated by StyleGAN2

4.1.1 Properties of $\overline{FID}_\infty$ and $\overline{IS}_\infty$

- FID and IS are biased, and hence cannot be effectively used to compare generators.
- The extrapolation of FID_N is unreliable. So as to make it more dependable, we propose to use $\overline{FID}_\infty$.
- We calculate FID_∞, IS_∞ by using Quasi-Monte Carlo integration methods for estimation.

4.2 Learned perceptual image patch similarity metric (LPIPS)

It has negative correlation with natural human perception. Higher LPIPS means more difference in the true and generated images.

4.3 Qualitative Comparison

The images generated by StyleGAN and StyleGAN2 are shown in figures [5], [6] respectively. We compute all the error metrics for a set of 2000 images generated from StyleGAN and StyleGAN2. These measures are very different from what is mentioned in [Karras, Laine, Aittala et al. 2020] because we have generated a small dataset as compared to the main research paper.

Table 1: Quantitative comparison of StyleGAN and StyleGAN2

	FFHQ 1024x1024		LSUN Car 1024x1024		LSUN Cat 128x128	
Error Metric	StyleGAN	StyleGAN2	StyleGAN	StyleGAN2	StyleGAN	StyleGAN2
FID	49.99	**44.36**	33.34	**30.68**	55.89	**50.67**
PPL	1234	**1086**	1865	**813**	1067	**998**
$\overline{FID}_\infty$	**49.992**	51.360	29.87	**28.65**	49.87	**45.43**
$\overline{IS}_\infty$	3.537	**3.697**	2.47	**2.95**	2.18	**2.64**
LPIPS	0.18	**0.16**	0.15	**0.13**	0.17	**0.16**

4.4 Implementation

Nvidia labs has provided codes for both StyleGAN and StyleGAN2 on the following links:

- https://github.com/NVlabs/stylegan
- https://github.com/NVlabs/stylegan2

We have implemented all the error metrics explained in Section 4 from scratch. These code are availabale at

- https://github.com/shongi-yd/AIP_project

We have trained the styleGAN and styleGAN2 for fewer epochs. For performance comparison we use the pre-trained checkpoints provided by NVlabs as training was taking a lot of time. These codes were run on

While training, we use different batch size for different resolution, so larger image size could fit into GPU memory. The keys is image resolution in log2

Table 2: Image resolution, Batch size and Train step ratio

Image resolution	Batch size	Train step ratio
2	16	1
3	16	1
4	16	1
5	16	1
6	16	1
7	8	2
8	4	4
9	2	8
10	1	16

We have pre-processed each image to a size of (64,64) and divided the training set in batch size of 32 and have further normalized each image by diving each pixel intensity with 255.

4.5 FID_∞, IS_∞ calculation

- Compute both FID and IS
- Using estimates obtained at regular intervals over N, extrapolate to get FID_∞ and IS_∞ estimates
- Repeat multiple times to get a variance estimate. This corresponds to how reliable the FID_∞ and IS_∞ estimates are

Bibliography

Chong, Min Jin and David Forsyth (2019). 'Effectively Unbiased FID and Inception Score and where to find them'. In: *arXiv preprint arXiv:1911.07023*.

Karras, Tero, Samuli Laine and Timo Aila (2018). *A Style-Based Generator Architecture for Generative Adversarial Networks*. DOI: 10.48550/ARXIV.1812.04948. URL: https://arxiv.org/abs/1812.04948.

Karras, Tero, Samuli Laine, Miika Aittala et al. (2020). 'Analyzing and Improving the Image Quality of StyleGAN'. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116.