A

Seminar Report

A Style-Based Generator Architecture for Generative Adversarial Networks

Submitted in partial fulfillment of the requirements for the Award of the Degree

of

Master of Computer Applications

of

APJ Abdul Kalam Technological University

Submitted by

VYSHAK PUTHUSSERI

RegNo: TVE17MCA054

Department of Computer Applications

COLLEGE OF ENGINEERING TRIVANDRUM

OCTOBER 2019

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING TRIVANDRUM

CERTIFICATE

Certified that this Seminar report entitled, ***"A Style-Based Generator Architecture for Generative Adversarial Networks "*** *is the paper presented by* ***" VYSHAK PUTHUSSERI "(Reg No: TVE17MCA054)*** *in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications of APJ Abdul Kalam Technological University during the year 2019.*

Prof. Jose T Joseph. Prof. Sabitha.

Co-ordinator **Head of the Department**

Acknowledgement

First and for most I thank **GOD** almighty and to my parents for the success of this seminar. I owe a sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my seminar.

I would like to thank **Dr Jiji C V**, Principal, College of Engineering Trivandrum, who helped me during the entire process of work.

I am extremely grateful to **Prof.Sabitha**, HOD, Dept of Computer Applications, for providing me with best facilities and atmosphere for the creative work guidance and encouragement.

I would like to thank my coordinator, **Prof.Jose T Joseph**, Dept of Computer Applications, who motivated me throughout the work of my seminar.

I profusely thank other Asst. Professors in the department and all other staffs of CET, for their guidance and inspirations throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this seminar. No words can express my humble gratitude to my beloved parents and relatives who have been guiding me in all walks of my journey.

Vyshak Puthusseri

Abstract

Generative adversarial networks (GANs) is a type of generative model. It is a powerful class of neural networks that are used for unsupervised learning. A GAN consist of two neural networks, a generator network and a discriminator network. The generator generates the data taking random noise as input. The discriminator has to classify whether data generated by the generator is a real or fake data The generator competes against its adversary, the discriminator. As the training for this generative model progresses, the discriminator learns to classify the fake data accurately, while the generator learns to create realistic samples. An equilibrium is reached when the data created by the generator is indistinguishable from real data. GANs are frequently used in image generation and, they produce sharp images too. A downside for GAN is that it does not have a welldefined loss function, which makes training GANs difficult. Although the whole process was computationally complex as all the possible combinations are tried by the generator to produce realistic data. An alternative generator architecture for GAN was proposed, borrowing from style transfer literature, which reduce the complexity exponentially. The new generator improves the state-of-the-art in terms of traditional distribution quality metrics, leads to demonstrably better interpolation properties. In the end, the result was an exceptionally differed and excellent dataset of human appearances called FFHQ. The resolution and quality of images produced by generative method using style transfer was of 1024 * 1024, which was exceptionally sufficient include all the human face features.

Keywords: Generative algorithms, Adversarial training, Style transfer ,Convolutional Neural Network

Contents

List of Figures

Chapter 1

Introduction

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text.

In Artificial Intelligence (AI), we can think of machine learning (ML) as one of AI's smaller subsets. Machine learning uses statistical techniques to provide computer systems ability to progressively improve its performance on a specific task with data without being explicitly programmed. Unsupervised learning algorithms are a subset of machine learning algorithms which tries to describe the structure of unlabelled data. Generative models is an approach to unsupervised learning. The goal of a generative model is to generate data similar to the ones in the dataset. Generative Adversarial Network (GAN) is a type of Generative Model. Other types of generative models include Variational Autoencoders (VAEs) and autoregressive models like PixelRNN. GANs have been successfully applied to solve problems in various domains like generating images, videos and audio, text to image synthesis etc. GANs were originally introduced by Ian Goodfellow and his collaborators in University of Montreal in 2014 . Yann LeCun, Director of AI Research at Facebook and Professor at NYU called adversarial training as "the most interesting idea in the last 10 years in ML" .

Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as

deep neural learning or deep neural network. The resolution and quality of images produced by generative methods especially generative adversarial networks (GAN) have seen rapid improvement recently. Yet the generators continue to operate as black boxes, and despite recent efforts, the understanding of various aspects of the image synthesis process, e.g., the origin of stochastic features, is still lacking.

The properties of the latent space are also poorly understood, and the commonly demonstrated latent space interpolations provide no quantitative way to compare different generators against each other. Persuaded by style transfer literature, re-designing of the generator architecture in a way that exposes novel ways to control the image synthesis process. The modified generator architecture starts from a learned constant input and adjusts the "style" of the image at each convolution layer based on the latent code, therefore directly controlling the strength of image features at different scales. Combining the gaussian noise injected directly into the network, this architectural change leads to automatic, unsupervised separation of high-level attributes (e.g., pose, identity) from stochastic variation (e.g., freckles, hair) in the generated images, and enables intuitive scale-specific mixing. The discriminator and the loss functions are not altered in the new architecture so that it was orthogonal to the GAN loss functions, regularization, and hyperparameters. The result of the style based GAN was a new dataset of human faces (Flickr-Faces-HQ, FFHQ) . The FFHQ database contains the images with 1024 * 1024 resolution, which is much better than the benchmark image resolutions used for the industrial problems.

1.1 The Perceptron

The Perceptron is one of the simplest ANN architectures, invented in 1957 by Frank Rosenblatt. It is based on a slightly different artificial neuron called a linear threshold unit (LTU): the inputs and output are now numbers (instead of binary on/off values) and each input connection is associated with a weight. The LTU computes a weighted sum of its inputs ($z = w1\ x1 + w2\ x2 + ... + wn\ xn = wT \cdot x$), then applies a step function to that sum and outputs the result. LTU is only a concept, by applying a learning rule to the LTU results in the percepton. The problem with the perceptron is that its only possible for binary classification problems.

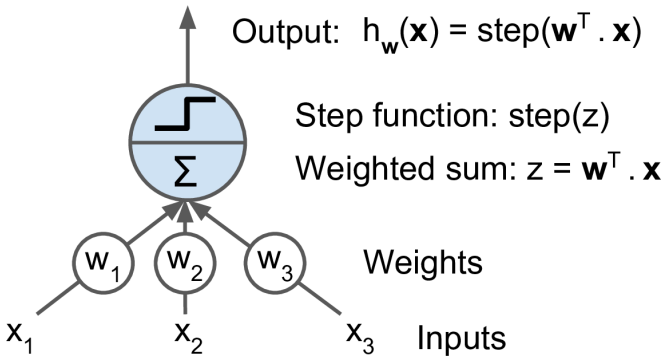

Figure 1.1: Perceptron

1.2 Multi-Layer Perceptron and Backpropagation

An MLP is composed of one (passthrough) input layer, one layers of LTUs, called hidden layers, and one final layer of LTUs called the output layer. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. To create a multi layer perceptron, multiple LTUs are stacked side by side and on the top. As this was mainly used for multi class classification problem, the activation function used will be sigmoid activation.

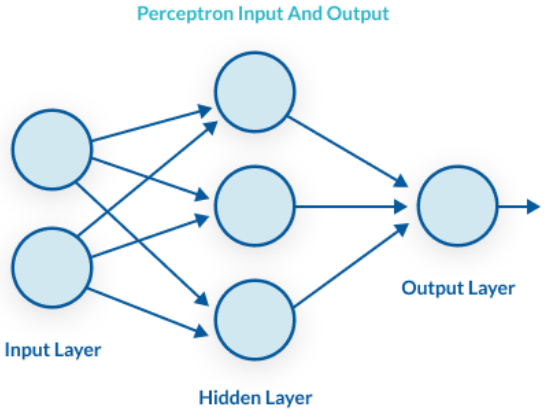

Figure 1.2: Multi Layer Perceptron

1.3 Deep Neural Network

If only single hidden layer was used in an architecture it is known as MLP. If multiple hidden layers are used in the architecture it is known as deep neural networks. A learning weights in a deep neural network is known as Deep Learning.

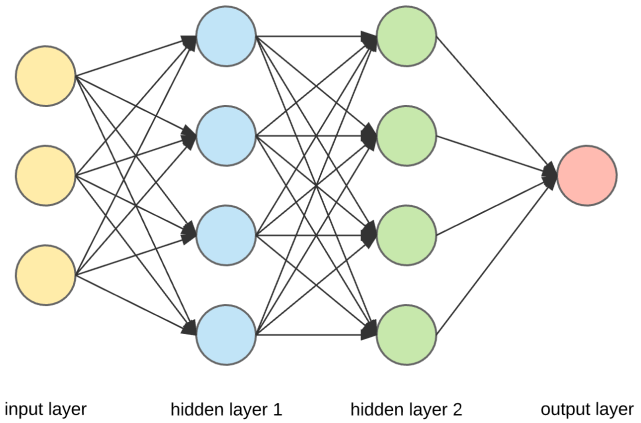

Figure 1.3: Deep Neural Network

1.4 Convolutional Neural Networks

While dealing with image data, deep neural networks fails upto an extend as it cant extract the structural information from the images. CNN image classifications takes an input image, process it and classify it under certain categories. Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values)

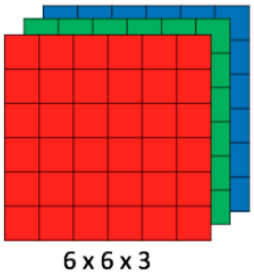

Figure 1.4: Array representation of an image

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernal. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by

applying filters.(Kernels). Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time.Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Min pooling

In the layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like Deep neural network.

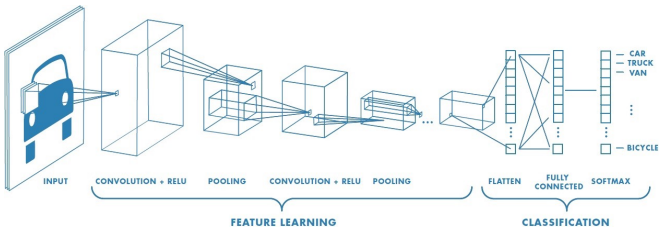

Figure 1.5: Convolutional Neural Network model

Chapter 2

Neural Style Transfer

Style transfer is the technique of recomposing images in the style of other images or we can say style transfer relies on separating the content and style of an image. Given one content image and one style image, we aim to create a new, target image which should contain our desired content and style components:

- objects and their arrangement are similar to that of the content image (feature reconstruction)
- style, colors, and textures are similar to that of the style image (texture synthesis)

2.1 Content representation and loss

We can construct images whose feature maps at a chosen convolution layer match the corresponding feature maps of a given content image. We expect the two images to contain the same content, but not necessarily the same texture and style. Given a chosen content layer **l**, the content loss is defined as the Mean Squared Error between the feature map **F** of our content image **C** and the feature map **P** of our generated image **Y**.

$$Lcontent = \frac{1}{2} \sum_{ij} (Fij - Pij)^2$$

When this content-loss is minimized, it means that the mixed-image has feature activation in the given layers that are very similar to the activation of the content-image. Depending on which layers we select, this should transfer the contours from the content-image to the mixed-image.

2.2 Style representation and loss

We will do something similar for the style-layers, but now we want to measure which features in the style-layers activate simultaneously for the style-image, and then copy this activation-pattern to the mixed-image. One way of doing this, is to calculate the Gram-matrix(a matrix comprising of correlated features) for the tensors output by the style-layers. The Gram-matrix is essentially just a matrix of dot-products for the vectors of the feature activations of a style-layer. If an entry in the Gram-matrix has a value close to zero then it means the two features in the given layer do not activate simultaneously for the given style-image. And vice versa, if an entry in the Gram-matrix has a large value, then it means the two features do activate simultaneously for the given style-image. We will then try and create a mixed-image that replicates this activation pattern of the style-image.If the feature map is a matrix F, then each entry in the Gram matrix G can be given by:

$$Gij = \sum_{k} FikFjk$$

The loss function for style is quite similar to out content loss, except that we calculate the Mean Squared Error for the Gram-matrices instead of the raw tensor-outputs from the layers.

$$Lstyle = \frac{1}{2}\left(\sum_{i=0}^{L}(Gij - Aij)^2\right)$$

As with the content representation, if we had two images whose feature maps at a given layer produced the same Gram matrix we would expect both images to have the same style, but not necessarily the same content. Applying this to early layers in the network would capture some of the finer textures contained within the image whereas applying this to deeper layers would capture more higher-level elements of the image's style. Gatys et. al found that the best results were achieved by taking a combination of shallow and deep layers as the style representation for an image.We can see that the best results are achieved by a combination of many different layers from the network, which capture both the finer textures and the larger elements of the original image.

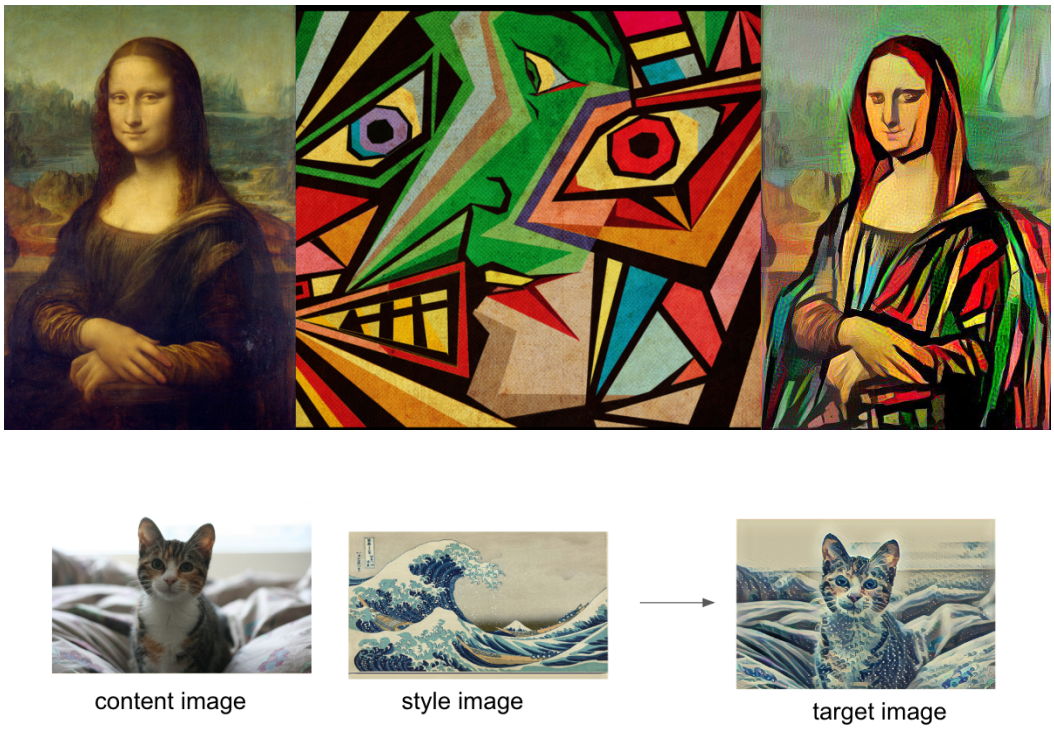

Figure 2.1: Style Transfer

2.3 Optimizing loss function and styling the image

Using a pre-trained neural network such as VGG-16, an input image (i.e. an image which provides the content), a style image (a painting with strong style elements) and a random image (output image), one could minimize the losses in the network such that the style loss (loss between the output image style and style of 'style image'), content loss (loss between the content image and the output image) and the total variation loss (which ensured pixel wise smoothness) were at a minimum. In such cases, the output image generated from such a network, resembled the input image and had the stylist attributes of the style image. The total loss can then be written as a weighted sum of the both the style and content losses.

$$Ltotal = \alpha Lcontent + \beta Lstyle$$

We will minimize our total loss by Adam optimizer. As our loss go down we will go close to our goal of producing a style transfer image Y.

Chapter 3

Generative Adversarial Network

A GAN comprises two components, a generator (G) and a discriminator (D). The goal of the generator model is to produce new data similar to the required one. The discriminator's task is to classify the data presented to it as real or fake. Real data belong to the original dataset, and fake data are those forged by the generator. The generative model competes against its adversary, the discriminative model that learns to determine whether a sample is from the model distribution or the data distribution.

3.1 Analogy

Generative networks can be thought of as a team of counterfeiters trying to produce fake currency notes and use it without being caught by the police. Here discriminative networks play the role of police trying to detect fake currency. Initially, both the police and counterfeiters are not very experienced, but as the game between them progresses, both parties master what they were doing. The game continues until the fake currency notes produced by the counterfeiters are indistinguishable from real currency.

3.2 A Mathematical Model

Assume that the generator represented by the neural network $G(z, \theta_1)$ converts the input noise z into the required output space. Conversely, a second neural network $D(x, \theta_2)$ represents discriminator, and it calculates the probability that x came from the real dataset. Here, θ_1 and θ_2 represents the weights that describe each neural network.

The discriminator is trained to classify the input data as either real or fake. The weights of the discriminator are updated so that it maximizes the probability that real images from the database are classified as real and minimizes the probability that images generated by G are classified as fake. The loss function used for the discriminator maximizes $D(x)$ and minimizes $D(G(z))$.

The generator's weights are trained to maximise the probability that it can fool the discriminator using the images it generates. The loss function maximizes $D(G(z))$.

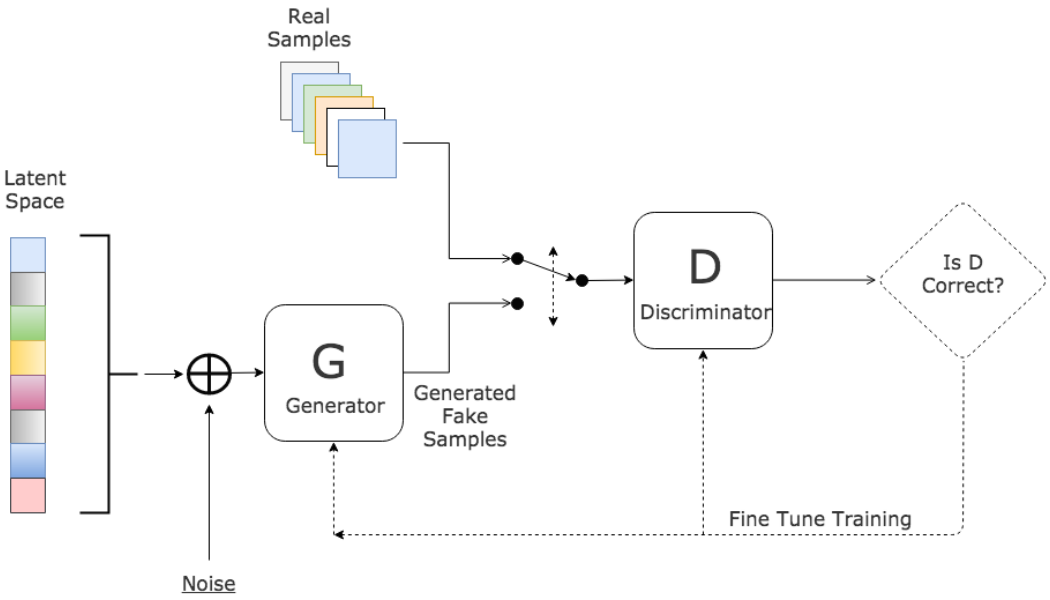

Figure 3.1: Generative Adversarial Network Architecture

Figure 3.1 shows a visual representation of the high level overview of GANs discussed in this section. During training, the generator is trying to maximize $D(G(z))$ and discriminator is trying to minimize $D(G(z))$. Thus we can think of the scenario as the generator and discriminator as playing a minimax game.

3.3 Applications of GAN

3.3.1 Generate Anime characters

Game development and animation production are expensive and hire many production artists for relatively routine tasks. GAN can auto-generate and colorize Anime characters.

Figure 3.2: GAN generated anime characters

3.3.2 CycleGAN

Cross-domain transfer GANs will be likely the first batch of commercial applications. These GANs transform images from one domain to another domain. (For example, it can transform pictures between zebras and horses.

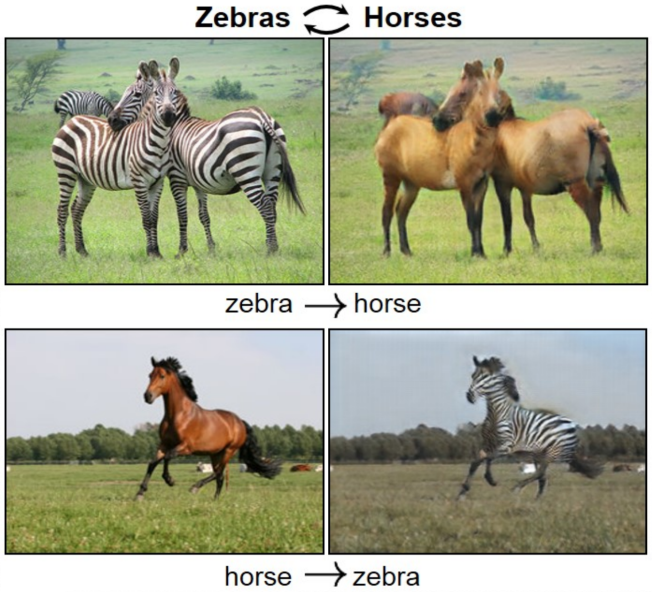

Figure 3.3: CycleGAN horse to zebra

3.3.3 PixelDTGAN

Suggesting merchandise based on celebrity pictures has been popular for fashion blogger and e-commerce. PixelDTGAN creates clothing images and styles from an image.

Figure 3.4: PixelDTGAN

3.3.4 Super resolution

Create super-resolution images from the lower resolution. This is one area where GAN shows very impressive result with immediate commercial possibility.

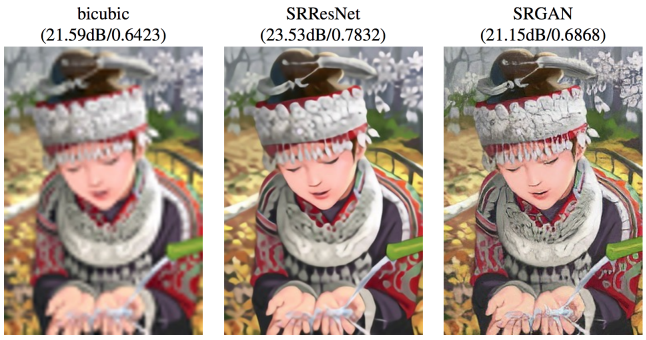

Figure 3.5: SuperResolution using GAN

Chapter 4

StyleGAN

In past, most improvement has been made to discriminator models in an effort to train more effective generator models, although less effort has been put into improving the generator models. 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. The resulting model is capable not only of generating impressively photorealistic high-quality photos of faces, but also offers control over the style of the generated image at different levels of detail through varying the style vectors and noise.

4.1 Lacking Control Over Synthesized Images

Generative adversarial networks are effective at generating high-quality and large-resolution synthetic images. Many improvements to the GAN architecture have been achieved through enhancements to the discriminator model. These changes are motivated by the idea that a better discriminator model will, in turn, lead to the generation of more realistic synthetic images. As such, the generator has been somewhat neglected and remains a black box. This limited understanding of the generator is perhaps most exemplified by the general lack of control over the generated images. There are few tools to control the properties of generated images, e.g. the style. This includes high-level features such as background and foreground, and fine-grained details such as the features of synthesized objects or subjects.

4.2 Control Style Using New Generator Model

The Style Generative Adversarial Network, or StyleGAN for short, is an extension to the GAN architecture to give control over the disentangled style properties of generated images. The StyleGAN is an extension of the progressive growing GAN that is an approach for training generator models capable of synthesizing very large high-quality images via the incremental expansion of both discriminator and generator models from small to large images during the training process. In addition to the incremental growth of the models during training, the style GAN changes the architecture of the generator significantly. The StyleGAN generator no longer takes a point from the latent space as input; instead, there are two new sources of randomness used to generate a synthetic image: a standalone mapping network and noise layers. The output from the mapping network is a vector that defines the styles that is integrated at each point in the generator model via a new layer called adaptive instance normalization. The use of this style vector gives control over the style of the generated image. Stochastic variation is introduced through noise added at each point in the generator model. The noise is added to entire feature maps that allow the model to interpret the style in a fine-grained, per-pixel manner. This per-block incorporation of style vector and noise allows each block to localize both the interpretation of style and the stochastic variation to a given level of detail.

4.3 Architecture

The StyleGAN is described as a progressive growing GAN architecture with five modifications, each of which was added and evaluated incrementally in an ablative study. The incremental list of changes to the generator are:

- Baseline Progressive GAN.
- Addition of tuning and bilinear upsampling.
- Addition of mapping network and AdaIN (styles).
- Addition of noise to each block.
- Addition Mixing regularization.

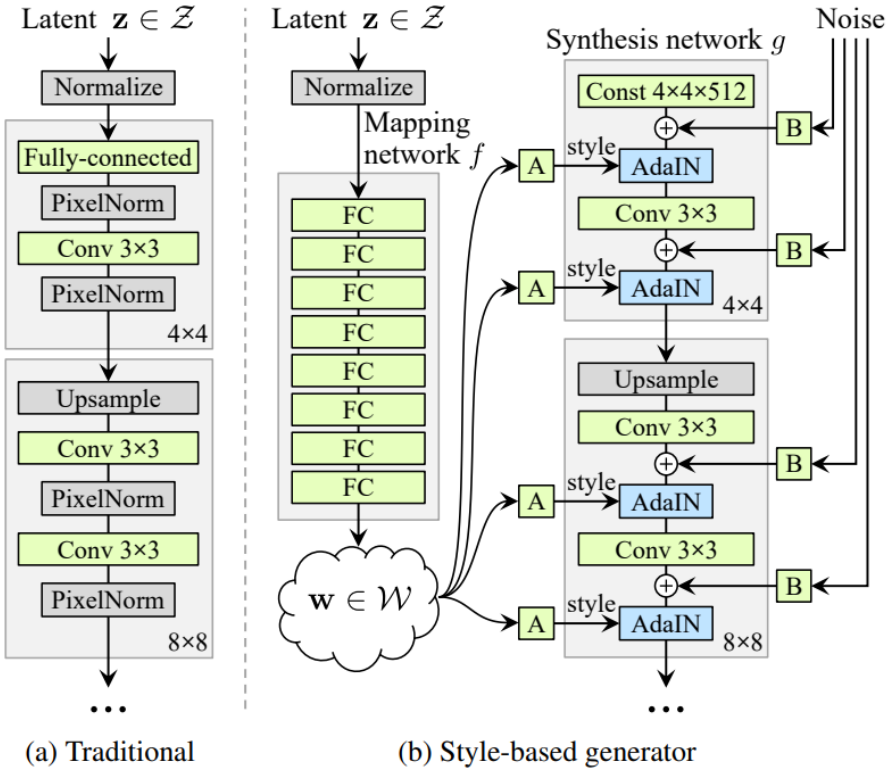

Figure 4.1: Style GAN Architecture

4.3.1 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.

4.3.2 Bilinear Sampling

The progressive growing GAN uses nearest neighbor layers for upsampling instead of transpose convolutional layers that are common in other generator models. The first point of deviation in the StyleGAN is that bilinear upsampling layers are unused instead of nearest neighbor.

4.3.3 Mapping Network and AdaIN

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. The style vector is then transformed and incorporated into each block of the generator model after the convolutional layers via an operation called adaptive instance normalization or AdaIN. The AdaIN layers involve first standardizing the output of feature map to a standard Gaussian, then adding the style vector as a bias term. The addition of the new mapping network to the architecture also results in the renaming of the generator model to a "synthesis network."

4.3.4 Addition of Noise

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. A different sample of noise is generated for each block and is interpreted using per-layer scaling factors. This noise is used to introduce style-level variation at a given level of detail.

4.3.5 Mixing regularization

Mixing regularization involves first generating two style vectors from the mapping network. A split point in the synthesis network is chosen and all AdaIN operations prior to the split point use the first style vector and all AdaIN operations after the split point get the second style vector. 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.

Chapter 5

Conclusion

The computational complexity of the GAN mainly depends on the complexity of the generator network. By using the style based generator architecture the computational complexity gets reduced upto a large extent. As the techniques like batch normalization, dropouts etc are not used in this architecture, the features of the images that can be generated can exponentially increase with less time. Also FFHQ database of images generated by the StyleGAN can be widely used for both finding solutions to both research and industrial problems.

Bibliography

[1] L. A. Gatys, A. S. Ecker, and M. Bethge. *Image style transfer using convolutional neural networks.* In Proc. CVPR, 2016.

[2] I.Goodfellow, J.Pouget-Abadie, M.Mirza, B.Xu, D.Warde-Farley, S. Ozair, A.Courville, and Y.Bengio. *Generative Adversarial Networks.* In NIPS, 2014.

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

[4] D. J. Rezende, S. Mohamed, and D. Wierstra. *Stochastic backpropagation and approximate inference in deep generative models.* In Proc. ICML, 2014

[5] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. *Improved techniques for training GANs.*In NIPS, 2016.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Dropout: A simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 15:1929–1958, 2014