GENERATION OF ANIME FACES USING GANs

A DISSERTATION OF MAJOR PROJECT-II

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

INFORMATION SYSTEMS

Submitted By:

PRATEEK KATIYAR

(2K/20/ISY/16)

Under the Supervision of

Prof. Dinesh Kumar Vishwakarma

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042, INDIA

2020 - 2022

CANDIDATE'S DECLARATION

I **Prateek Katiyar** Roll No. 2K20/ISY/16 student of M. Tech Information Systems, hereby declare that the project Dissertation titled "**Generation of Anime Faces Using GANs**" which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, Diploma Associateship, Fellowship or other similar title or recognition.

Prateek Katiyar

Place- Delhi **Prateek Katiyar**
Date- 13/05/2022

CERTIFICATE

I hereby certify that the Project Dissertation" **Generation of Anime Faces Using GANs**" which is submitted by Prateek Katiyar, Roll No 2K20/ISY/16, in department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision.

[illegible]

Place- Delhi
Date- 30/05/2022

Prof. Dinesh Kumar Vishwakarma
Supervisor

ACKNOWLEDGMENT

I express my gratitude to my major project Mentor **Prof. Dinesh Kumar Vishwakarma**, Professor Department of IT, Delhi Technological University, for the valuable support and guidance he provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for his constructive criticism and insight without which the project would not have been shaped as it has. I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Prateek Katiyar

Prateek Katiyar
Roll No. 2K20/ISY/16
M. Tech (Information Systems)

ABSTRACT

Generative Adversarial Networks are a topic of interest in the world of DNN (Deep Neural Networks). GAN frameworks were designed by Ian Goodfellow and his colleagues in June 2014. This idea took the world of deep learning by storm. In GAN two neural networks fight against each other, giving feedback to each other in the process, and eventually become very good at a particular task. This idea was so powerful that lots of applications of GANs emerged. Due to GANs, many futuristic concepts came to life. One such cool application is generating real-looking fake images. These images did not exist before but were completely formed by the Generative Adversarial Network. Many more applications like Text-to-image translation, video prediction, face aging, etc. were made possible using GANs. In this report, I will specifically present anime faces generation using GANs.

TABLE OF CONTENTS

Table Of Figures

Chapter 1

1. Introduction

Deep Neural Networks are usually used for supervised learning (Data with Labels) i.e., regression or classification. Generative Adversarial Networks use NN (neural networks) for quite different purposes that is Generative modelling. Generative Model is an unsupervised learning (No Labels) method in deep learning that requires spontaneously finding and understanding the patterns in data in such a way that the model can be used to produce unique examples that could have been fetched from the initial dataset used for training the generator.

There is a site called www.thispersondoesnotexist.com. Each time we refresh the site, a unique image of the face of a person is generated then and there. The results are quite mind blowing. All the generated images of persons are fake, but they look real. These persons never actually existed. Generative Adversarial Networks opens many doors to build and innovate new world changing technologies.

Generative Adversarial Networks, however, can be very hard to train, and are very sensitive to changes to hyperparameters, regularization and activation functions.

1.1 Motivation

GANs are dominating AI research. There are various applications of GANs. GANs are solving various real problems efficiently. One such application is Image Generation; this application of GAN tries to form real looking fake images i.e., images that are indistinguishable from real images for a normal observer. Times are gone for paper cartoons; in today's day and age everything is going digital. For many years we have been consuming digital animated content. What if we can automate the process of character generation? Different famous anime characters popped up in the last decade, they are good looking and liked by the masses. People even choose their avatar as an anime character on social media. But they all look the same. It is like being at a function or celebration and everyone is dressed up the same way. This is where GANs can prove to be gamechanger. Generative adversarial networks take lots of images as training

1.2 Objective

This project is made to perform the image generation using GAN for anime characters, given sufficient data model used for this project can be used to generate any types of images. Already available animation images are used for model training, and then Deep Convolutional Generative Adversarial Network (DCGAN) is used to form realistic looking fake animation characters. DCGAN is used to let the discriminator and generator methods iteratively provide feedback to each other to produce satisfactory results.

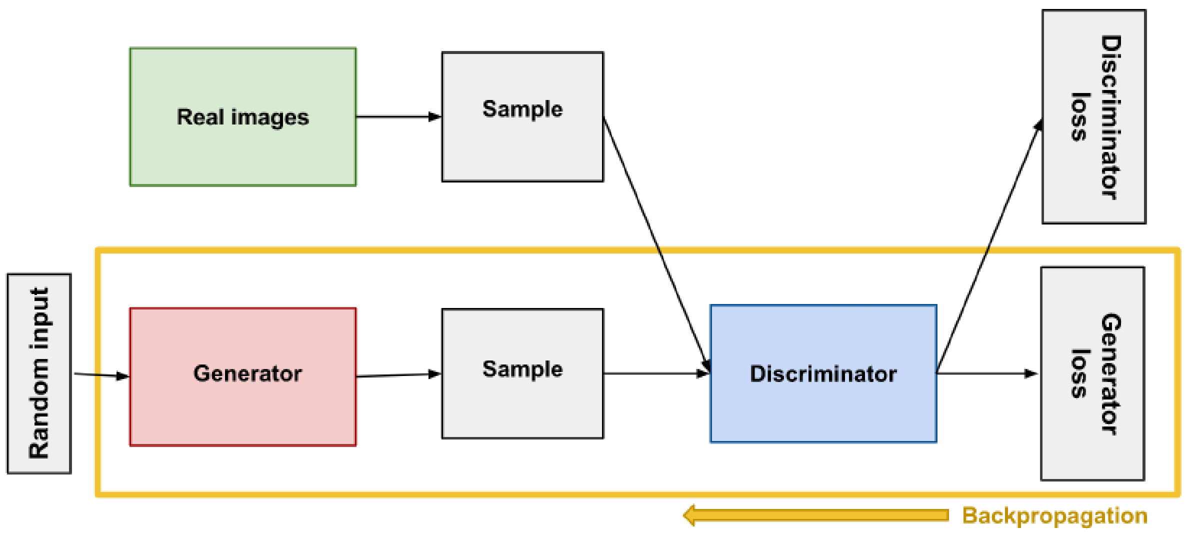

Figure 1 General Representation Of GAN

Chapter 2

2. Background and Related Work

2.1 Literature Review

Image to image translation [1] requires creating a new manufactured kind of image of a given image with a specific alteration, such as converting the image of a donkey to zebra. Many GAN models are proposed for this use case. We tried to understand and review a few of them [2].

For the purpose of Image generation, we analysed various types of GAN models.

1. CycleGAN

It is also known as the Cycle-Consistent GAN. CycleGAN [3] is a type of generative adversarial network model that seeks to answer the image-to-image translation problem. The aim of the image-to-image translation problem is to learn the mapping between an incoming image and an outgoing image using a training set of oriented image pairs. But getting paired examples is not always practical. CycleGAN strives to learn this mapping without requiring coupled input-output images, by using cycle-consistent adversarial networks. Below is the architecture of CycleGAN-CycleGAN utilizes a loss function called cycle consistency loss to empower training without requiring paired data. Or we can say, it can render from a domain to another without the need of a one-to-one mapping relationship between the input and output.

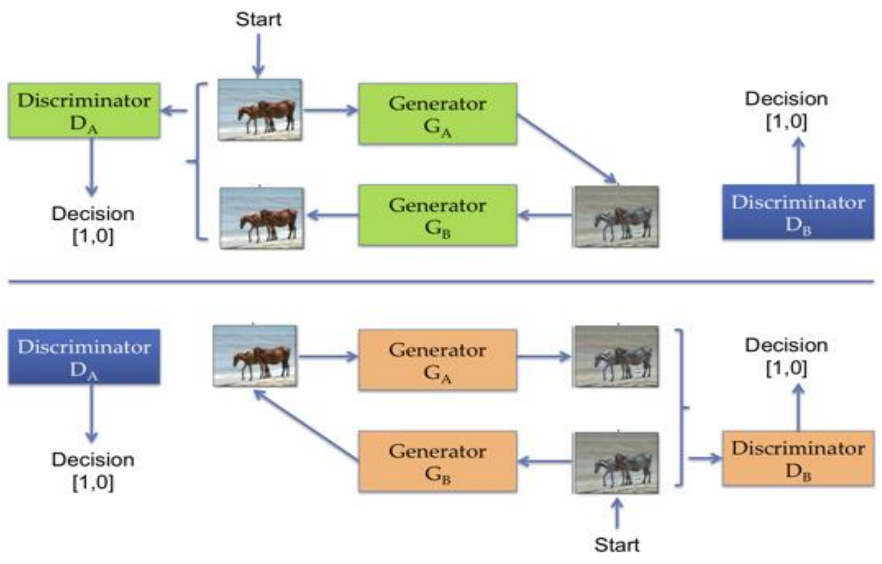

Figure 2 CycleGAN Architecture

Above figure 2 shows the working of CycleGAN. Generator takes the input of the image which need to be translated and discriminator provides feedback to generator to translate the image.

Loss Function

In Cycle-Consistent GAN, there is absence of mapped or labelled data for model training, hence there is no guarantee that the x and y pair where x is input, and y is target are of any use during model training. In order to make the network learn the correct labelling, the concept of cycle consistency loss was introduced.

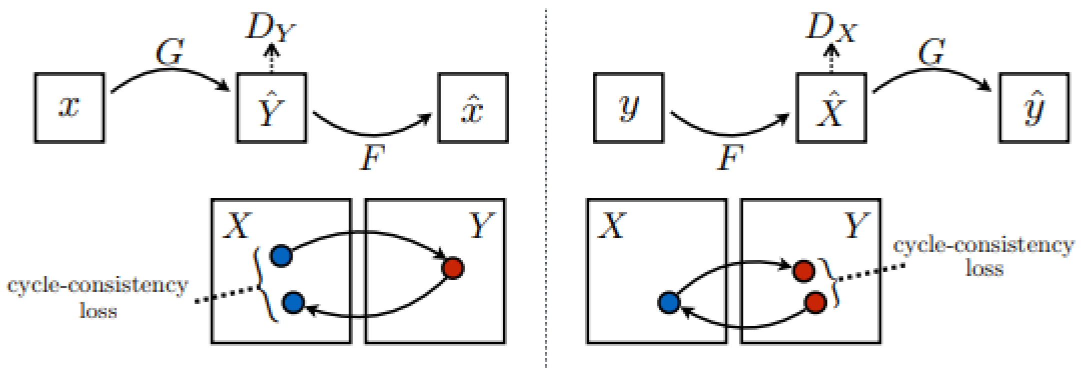

Figure 3 Forward and Backward Consistency Loss

Figure 3 shows the forward and backward consistency loss of CycleGAN.

2. Pix2Pix GAN

It is a type of conditional Generative adversarial network. Pix2Pix [1] GAN not only learns from data and noise but labels too, that is why it is called conditional GAN. Generator of Pix2Pix learns from labels and noise of the dataset.

$$G : \{x, z\} \rightarrow y \tag{1}$$

In Pix2Pix the discriminator is based on Patch GAN which is a convolutional classifier, and the generator is based on U-Net architecture. Due to this kind of setup cGANs are very good architecture for applications like image-to-image translation. Here instead of taking random seed as input generator has labelled data to learn from, it makes the process of image generation a lot faster compared to other architecture.

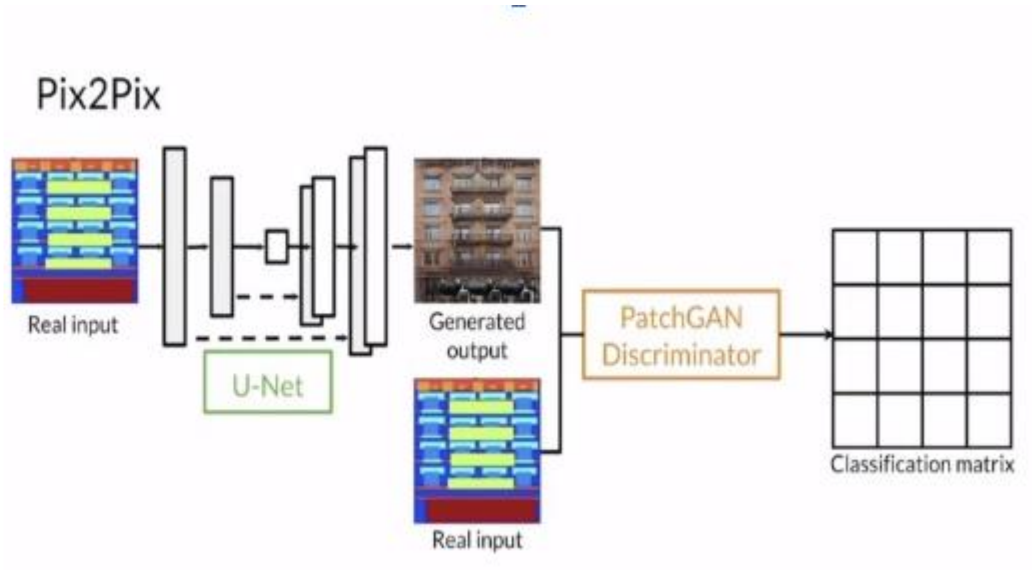

Figure 4 Pix2Pix GAN Architecture

Objective Function of Pix2Pix GAN is

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = &\mathbb{E}_{x,y}[\log D(x, y)] + \\ &\mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]\end{aligned} \tag{2}$$

Here in equation 3, G which is the generator, tries to maximize the objective function against the adversary D i.e., discriminator which tries to minimize it.

3. DCGAN

DCGAN [3] architecture is somewhat similar to GAN, there are few changes though i.e., some layers are replaced with other. Discriminator uses LeakyReLU across layers and batch normalization is also used. The discriminator is formed with conv2d layers, batchNorm2d, and LeakyReLU. The generator consists of convTranspose2d, batch norm batchNorm2d, and ReLU activations. The input to the discriminator of DCGAN is an image of size 3x64x64 and the output is based on the value that sigmoid activation function produces. Less than 0.5 then marked fake otherwise real. Input to the generator is a random vector which acts as a seed. Generator tries to form the image from this random vector, with each iteration this vector is updated after taking back propagated feedback from the discriminator.

Architecture of DCGAN

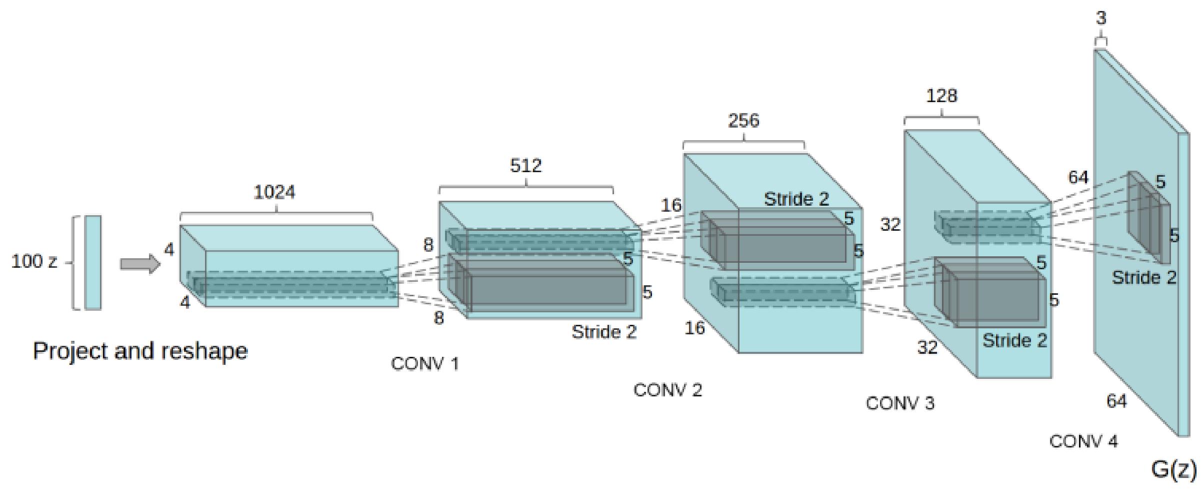

Figure 5 DCGAN architecture

In above figure 5 DCGAN architecture takes 100-dimensional input vector, generator converts this input to an image.

Architecture represented in below figure takes the random vector input of size 100. This input is reshaped into 4 * 4 * 1024 and pass on to convolution layer this layer. After convolution, batch norm image of size 3 * 64 * 64.

Loss function for generator

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \tag{3}$$

Loss function for discriminator

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right] \tag{4}$$

Figure 4 and 5 represents generator and discriminator loss. Generator tries to minimize the loss while discriminator tries to maximize it.

Chapter 3

3. Overview

This section will outline the approach, tools technologies used to navigate this project to completion.

3.1 Deep Learning

Concept of deep learning takes motivation from the human brain; deep learning tries to mimic the way the human brain learns and acts on the environment. Although it is yet to showcase human brain level capability across areas, it is already beating humans at particular tasks. As this technology progresses, we will see many changes in the way we live that are never thought of before. Below figure 6 shows the relationship between machine learning, deep learning and artificial intelligence.

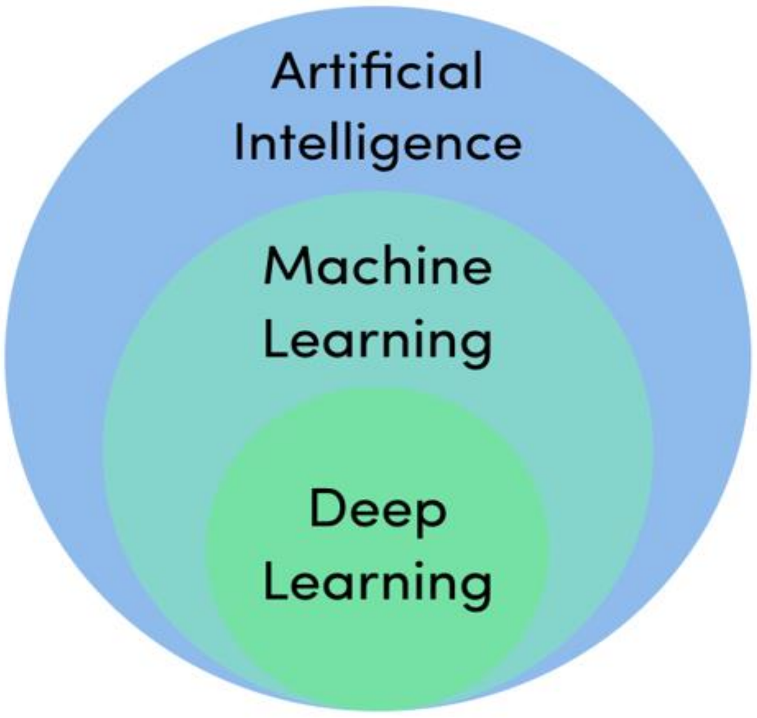

Figure 6 Representation of deep Learning w.r.t AI

A neuron is a basic building block of a neural network, during model training weight is associated with each neuron in the network. Loss is calculated at the last layer, and it is back propagated to adjust weights of the network. This improves accuracy of the model. Deep neural networks are very good at learning patterns compared to traditional machine learning algorithms.

3.2 CNN (Convolutional Neural Network)

Convolutional Neural Networks are at the heart of deep learning. CNNs are instrumental in quickly bridging the gap between humans and machine capabilities, the day is not very far away when machines will outperform humans in almost all areas. CNNs usually takes an Image as input and layer by layer it tries to identify the pattern in the image. We don't have to hand engineer many parameters; it automatically adjusts them. Any Image is broken into pieces as the image goes deeper into the neural network many convolution and pooling operations are performed.

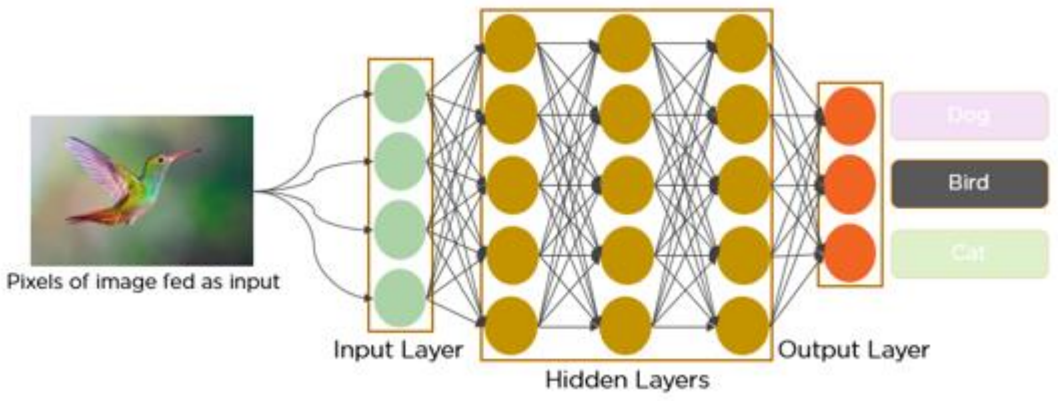

Figure 7 CNN Architecture

Above figure 7 represents CNN architecture. Images are basically a multidimensional matrix, so CNN takes advantage of this idea and by performing many matrix operations It produces optimum results.

3.3 Types of Pooling Layers

There are 3 types of pooling layers used in CNN.

1. Max Pooling Layer
2. Min Pooling Layer
3. Average Pooling Layer

3.3.1. Max Pooling Layer

In case of max pooling maximum pixel value is picked up from a group of pixels. It chooses the brightest pixel; Max pooling layer is suitable when the background is dark. It highlights the image against the dark background by choosing the only pixels that can be clearly seen.

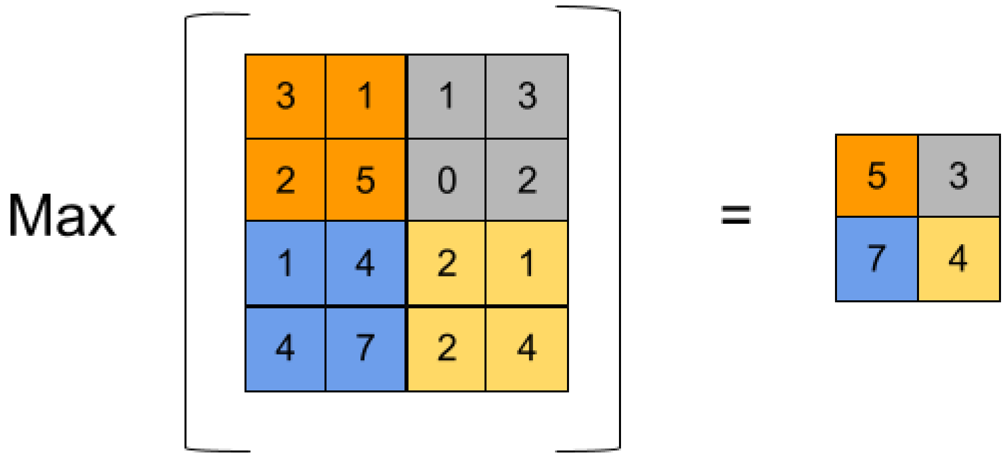

Figure 8 Representation of max pool operation

As we can see from figure 8 a 2 * 2 feature map is taken, and maximum value is taken from the available values.

3.3.2. Average Pooling Layer

As the name suggests It averages the values of pixels in a particular stride. So, the whole Image is down sampled to average pixel values. Average pooling layer helps to extract smooth features of an image.

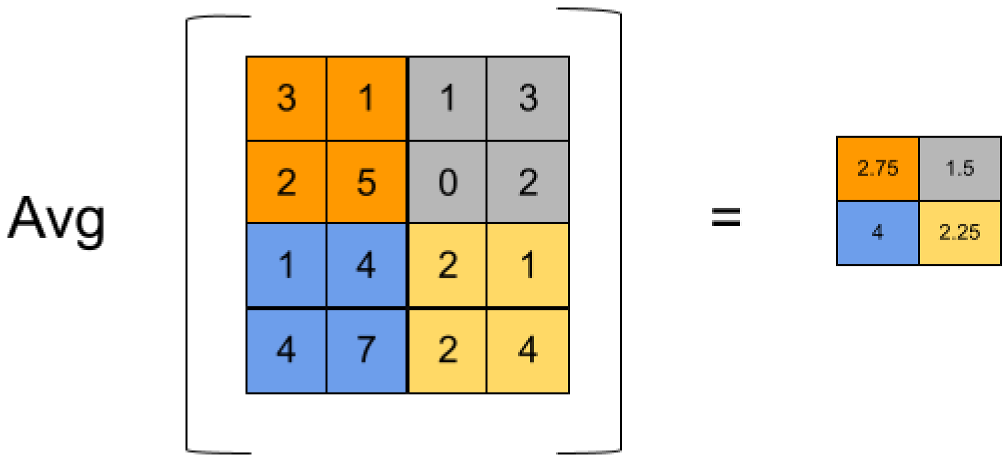

Figure 9 Representation of Average pool operation

As visible from the figure 9, the average pooling layer gives the average of the area covered by the feature map. Another use case of pooling layers is to reduce the dimensions of an image and save on the computation cost and reduce the number of trainable parameters. It also gives a summary of features in a feature map.

3.3.3. Min Pooling Layer

When we want to highlight an object in an image against the light background then the min pooling layer is used in a convolutional neural network.
It selects the features that are not very prominent and tries to highlight them for the neural network to learn.

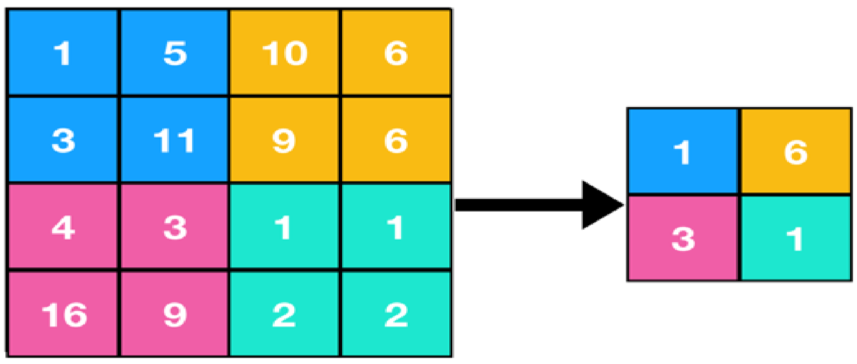

Figure 10 Representation of min pool operation

Here In the figure 10, we can see that 2 * 2 size feature map is taken and min value is picked up from the available values.

3.4 Types of Activation Functions used in this Project

1. Sigmoid
2. TanH
3. ReLU
4. LeakyReLU

Sigmoid

Sigmoid transforms the value of weights between 0 and 1. It is a non-linear activation function.

$$f(x) = \frac{1}{1+e^{-x}} \tag{5}$$

Equation 5 generates the output between 0 and 1.

TanH

TanH activation function is somewhat similar to sigmoid activation function, only difference is that it is symmetric around zero. So, its range is between -1 and 1.
It means that input to the next layer can be of opposite sign.

$$tanh(x) = 2 * sigmoid(2 * x) - 1 \tag{6}$$

Equation 6 generates the output between -1 and 1.

ReLU

ReLU is also a type of non-linear activation function. It is pronounced as Rectified Linear Unit. Main feature of ReLU is that it does not activate all the neurons, it activates only those neurons where value is positive and deactivates the neurons where the value is less than zero.

$$f(x) = max(0, x) \tag{7}$$

Equation 7 we can understand that ReLU discards negative values.

LeakyReLU

As the name suggests, it is nothing but an improved version of Rectified Linear Unit. In ReLU what happens is if the value is 0 or less than 0 i.e., < 0 then the neuron is deactivated so due to this there can be many dead neurons in a particular region to address this issue LeakyReLU uses extremely small linear component of a given variable.

$$f(x) = 0.01x, x < 0 \, or \, x, x >= 0 \tag{8}$$

As can be seen from equation 8 a small slope is introduced for values less than 0 so it avoids discarding all the negative values in this way.

3.5 Types of Layers Used in Model

1. Conv2d
2. ConvTranspose2d
3. BatchNorm2d
4. Flatten

Conv2d

This layer is a part of PyTorch deep learning library. It applies two-dimensional convolution over the input image, where all the features of the image are given like size, width, length, channels. A convolution operation is carried out on image represented as 2D matrix

These are the parameters used while using conv2d.

- Kernel Size
- Stride
- Padding
- Dilation
- Bias

ConvTranspose2d

In Pytorch this module is a gradient of conv2d. ConvTranspose2d is also known as deconvolution.

These are the parameters used with convTranspose2d

- Kernal_size
- Stride
- Padding
- Output padding

BatchNorm2d

BarchNorm2d is used to apply batch normalization over an input which is of 4 dimensions. The standard-deviation and mean are calculated for each dimension over the mini-batches and gamma and beta are learnable parameters of size C.

Parameters used in BatchNorm2d-

- Num_features

Flatten

Flatten layer is used to convert a 2D matrix to a one-dimensional vector. Its output is fed to a fully connected layer.

Chapter 4

4. Model Used for This Project

4.1. Approach

We have implemented a deep convolutional generative adversarial network model [3] which is also known as DCGAN.
In many ways DCGAN architecture is similar to original GAN architecture but there are some differences.

- DCGAN uses batch normalization in the discriminator and generator.
- All layers of the discriminator use the LeakyReLU activation function.
- Fully connected layer which takes input from a flatten layer is removed.
- Generator uses ReLU activation function but for the last layer it uses Tanh activation function.
- Pooling layers are removed.

As the training progresses the generator produces better and better images to make it harder for the discriminator to distinguish between real images from dataset and generated images. But with the training process the discriminator also becomes better at classification. Whole process reaches completion when the discriminator can’t classify that it’s a fake image.

4.2 Model Architecture

4.2.1. Generator

Generator basically generates the images from random input vector and passes the image to the discriminator. Then the discriminator classifies between real image and generated image and calculates the loss, after that discriminator passes on the feedback to the generator to adjust the input vector. So, this process goes on iteratively until the generator starts producing images that are impossible for the discriminator to classify as fake. Our generator neural network has 14 layers which are stacked sequentially. Below figure shows how these layers are stacked -

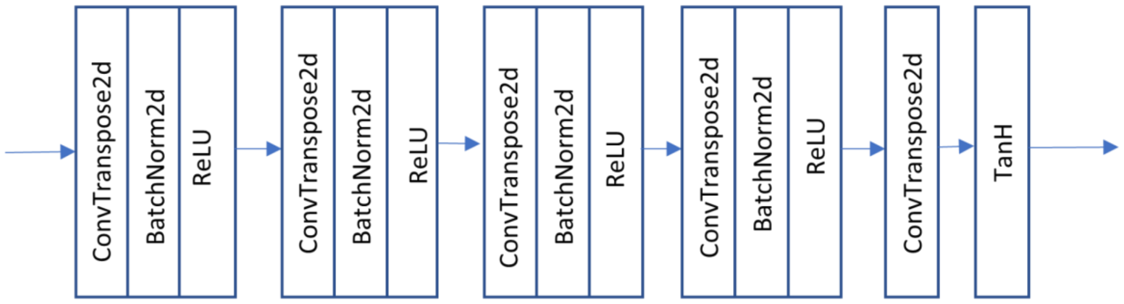

Figure 11 Arrangement of Layers for Generator

Above figure 11 represents the generator architecture used in the model. The generator takes in a 128-dimensional noise vector. Convolution layers transform the input vector and convert it to the 64 * 64 * 3 image.

4.2.2. Discriminator

Like any other classifier, a discriminator's job is to classify the images. It works like a binary classifier. Image generated from the generator goes to the discriminator then discriminator classifies between real or fake. Our Discriminator has 15 layers which are attached sequentially.

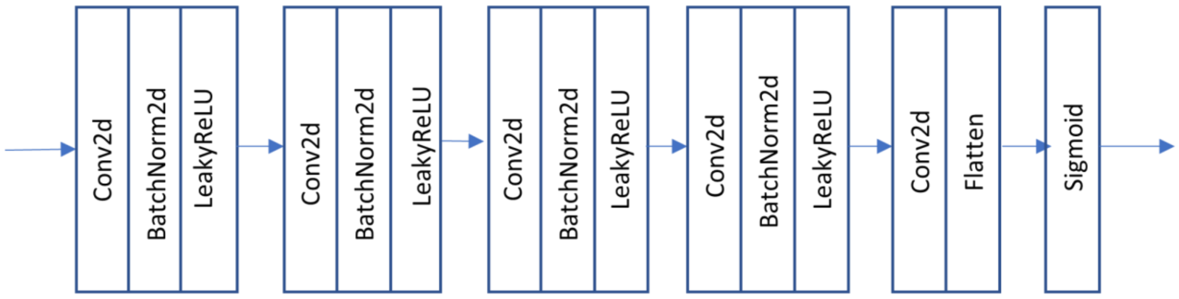

Figure 12 Arrangement of layers for discriminator

Figure 12 represents the architecture of discriminator. At the last layer It uses sigmoid activation function, the range of sigmoid is between 0 and 1 so it tries to classify at the last layer if the image is real or fake. If the value is less than 0.5 then it is classified as fake otherwise if it is greater than 0.5 then it is classified as real. Whenever it classifies as real training is usually stopped.

4.3 Dataset

For this project we used a dataset from Kaggle. This dataset has 63632 high quality images of anime faces.
Below Image shows the sample of the dataset

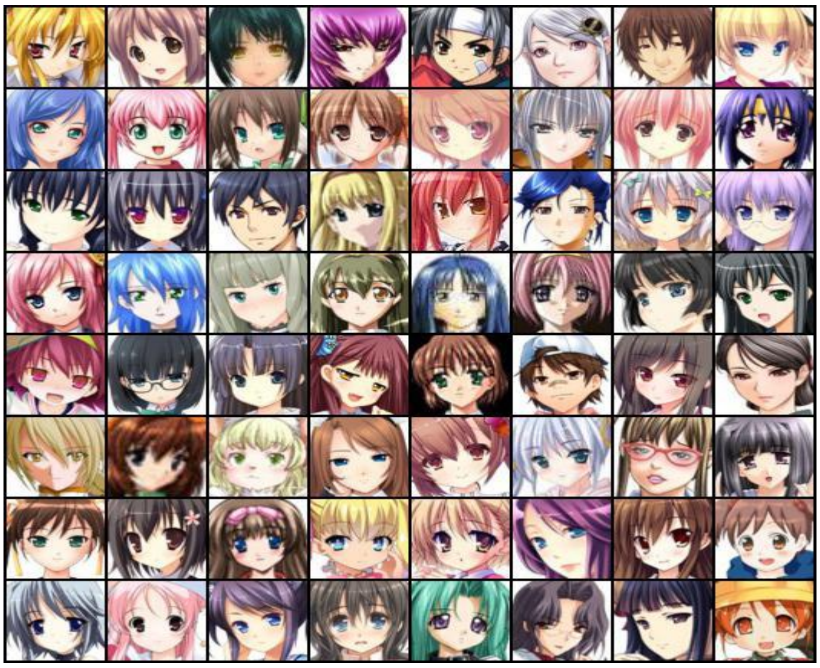

Figure 13 Anime Images From the dataset

As can be seen from the figure 13, this dataset contains pictures of anime faces of different characters. These images are taken from various Japanese anime series. Images are fed to the model in the batches of size 128 and image size is taken as 3 * 64 * 64.

4.4 Results

After training for 20 epochs, the model was generating real looking fake anime faces. Result after 1 epoch was complete noise because the generator started with a random vector. After the 2nd epoch the model started to show something like an anime face visible in figure 2 but still very noisy. Below is the snapshot from the model training-

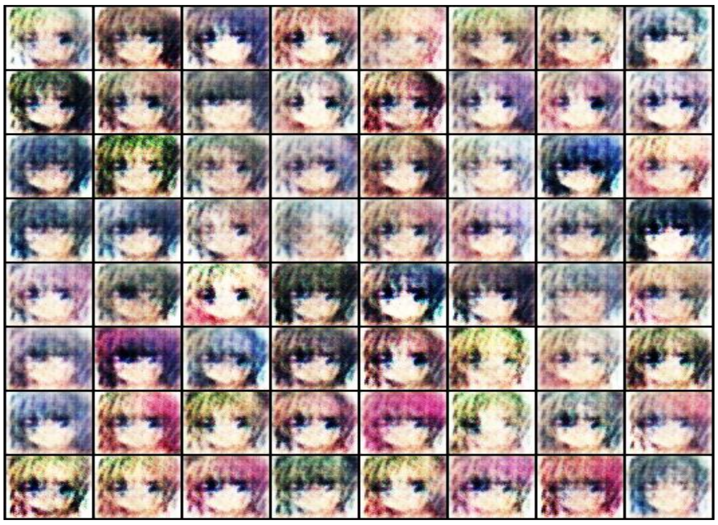

Figure 14 Generated Images after 2 epochs

After 20th epoch results were very close to anime faces as can be seen from figure 15.

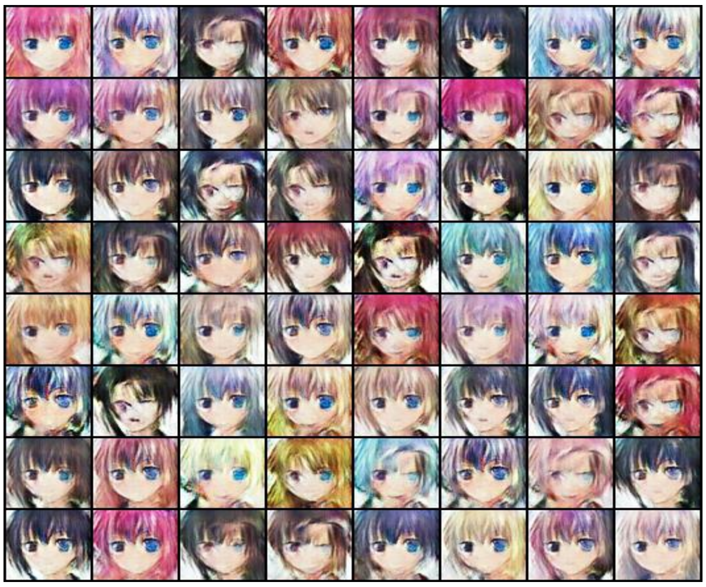

Figure 15 Generated Images after 20 epochs

Plot for Generator and Discriminator Losses

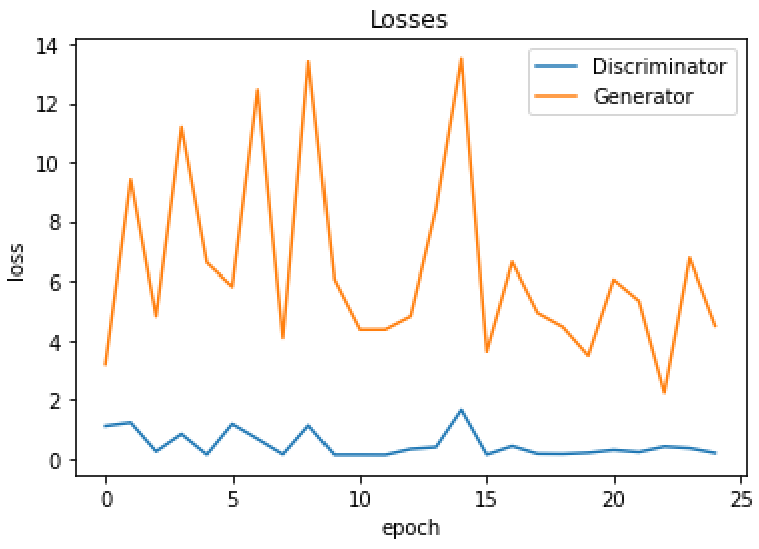

Figure 16 Generator and Discriminator Loss for 25 epochs

Plot for Real Score and Fake Score

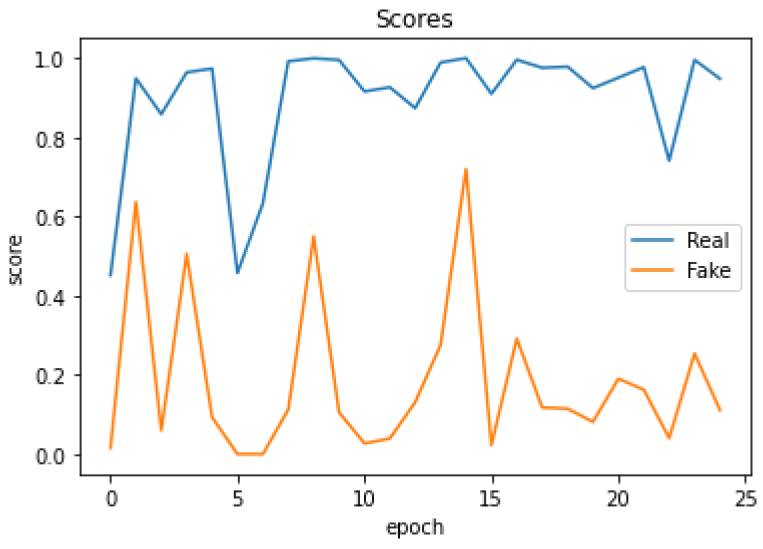

Figure 17 Fake Score and Real Score of Discriminator

4.5 Comparison of StyleGAN2 and DCGAN

We tried training the dataset on StyleGAN2 [3] and DCGAN. As there is no matrix to check the quality of the image so we checked the results manually. We found out that the results were quite similar, a little better with DCGAN.

As features of the image in the dataset are not very complex so they produce similar results. For our project DCGAN is perfect as it is a little less complex architecture compared to StyleGAN2 with similar or better results.

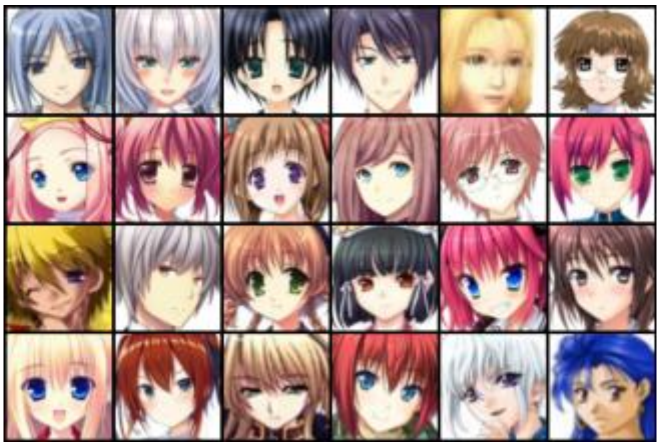
Figure 18 Images from Dataset

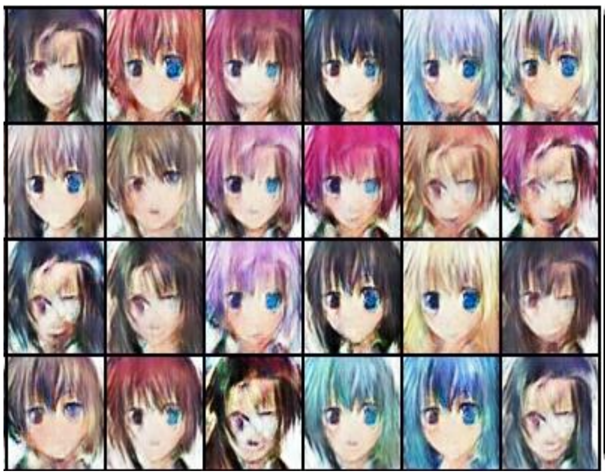
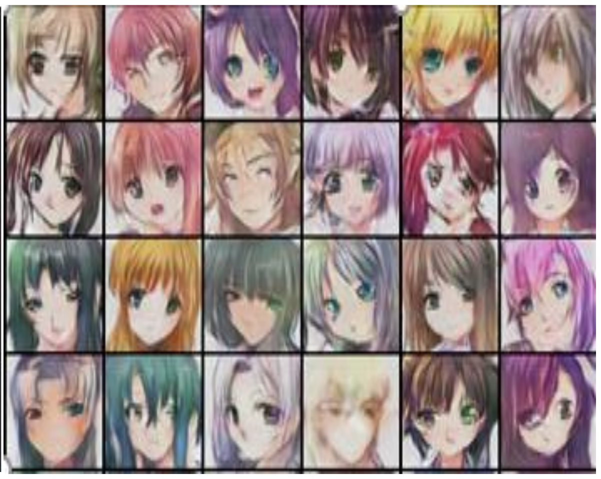
Figure 19 Image Generated by DCGAN

Figure 20 Image Generated by StyleGAN2

As can be seen from the figure 19 and 20, some facial features are more prominent in DCGAN generated images like eyes. If we see the figure 18 which are images from, the dataset then we find that images generated by DCGAN are more polished and better looking compared to StyleGAN2. Although quality of generated images is more or less similar.

5. Conclusion

We can conclude that GANs are one of the best architectures when it comes to image generation. Image Generation brings a wide range of applications across industries. It can help the entertainment industry, media industry, artwork etc. Adobe is working on such a software where artists can describe the image in words and software will form the image with the help of AI, it is a direct application of GAN's text to image synthesis capability. In the coming years we will explore many more applications.

References

[1] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017.

[2] L. Wang, W. Chen, W. Yang, F. Bi and F. R. Yu, "A State-of-the-Art Review on Image Synthesis With Generative Adversarial Networks," no. IEEE, 2020.

[3] J.-Y. Zhu, T. Park, P. Isola and A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *IEEE International Conference on Computer Vision (ICCV)*, Venice, 2017.

[4] Z. Li and Q. Wan, "Generating Anime Characters and Experimental Analysis Based on DCGAN Model," in *2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, Shenyang, China, 2021.

[5] H. A. Wibowo, "Generate Anime Style Face Using DCGAN and Explore Its Latent Feature Representation," TowardsDataScience, April 2019. [Online]. Available: https://towardsdatascience.com/generate-anime-style-face-using-dcgan-and-explore-its-latent-feature-representation-ae0e905f3974.

[6] T. Karras, S. Laine and T. Aila, "A style-based generator architecture for generative adversarial networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, California, 2019.

[7] A. Radford, L. Metz and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *ICLR*, San Juan, Puerto Rico., 2015.