1. INTRODUCTION

1.1 Preamble

A Generative Adversarial Network is a class of machine learning system invented by Ian Goodfellow and his colleagues in 2014.GAN's belong to the set of generative models. Generative model is a class of models for unsupervised learning where goal is to generate new data from given distribution. Given a training set this technique learns to generate new data which closely resembles the training data models. GAN's take a game theoretic approach and learns to generate from training distribution through 2 player game. The GANs are deep neural network architectures comprising of two neural networks competing against one another. The two neural networks are generator and discriminator. Discriminator is trained with the training data set and generator is fed with some random noise and it is able to generate fake images which appear similar to authentic images. The role of discriminator is to classify real images (from the training data set) as real and generated images as fake.

1.2 Motivation

- In today's world, GAN (Generative Adversarial Networks) is an insanely active topic of research and it has already attracted a lot of creative applications.
- Its application is widely seen in the world of fashion, advertising, generating anime and cartoon characters.

1.3 Objectives of the project

- Generated images by generator should be of various classes.
- It should be applicable to all GAN models which are generating biased images.

1.4 Literature survey

1. Paper Name: GAN -What is Generative Adversarial Network[10]

Author-Jonathan Hui

In this article the author gives brief introduction to Generative Adversarial Network, what does GAN do, the main focus for GAN, their ability to generate data from scratch, the working of two neural networks Generator and Discriminator, how they compete with each other in order to get better than the other via min-max game and how the entire system can be trained with back propagation and the algorithm behind it.

2. Paper Name: GAN — A comprehensive review into the gangsters of GANs [9]

Author-Jonathan Hui

This article studies the motivation and the direction of the GAN research in improving GANs. This article gives the description on neural network design, different types of GAN how they differ from each other. The paper also gives insight on the different cost functions, trends to new cost function, new cost functions and new goals, cost vs. image quality, optimization and implementation tips.

3. Paper Name:GAN — DCGAN[11]

Author-Jonathan Hui

This paper gives brief description on Deep Convolutional GAN's,their network design,tuning tips and how it is one of the popular and successful network design for GAN.

4. Paper Name: GAN — Why it is so hard to train Generative Adversarial Networks![6]

Author -Jonathan Hui

Through the study, we understand why is it so difficult to train GAN's, fundamental problems that GAN models suffer from. Major problems discussed in the paper are Non-convergence: the model parameters oscillate, destabilize and never converge, Mode collapse: the generator collapses which produces limited varieties of samples, Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing, Unbalance between the generator and discriminator causing overfitting, & highly sensitive to the hyper parameter selections. The paper also gives insight on Nash Equilibrium, Cost functions, why do GAN models suffer from these problems.

5. Paper Name:GAN — Ways to improve GAN performance[12]

Author- Jonathan Hui

This article looks into ways on how to improve GAN. In particular, change the cost function for a better optimization goal, add additional penalties to the cost function to enforce constraints, avoid overconfidence and overfitting, better ways of optimizing the model, add label and it also gives us brief introduction on how to move towards convergent GAN Training by performing feature matching, minibatch discrimination.

1.5 Problem definition

Engineering a solution to enhance the stability by removing mode collapse in existing Generative Adversarial Network’s (GAN).

2. PROPOSED SYSTEM

2.1 Description of proposed system with simple block diagram

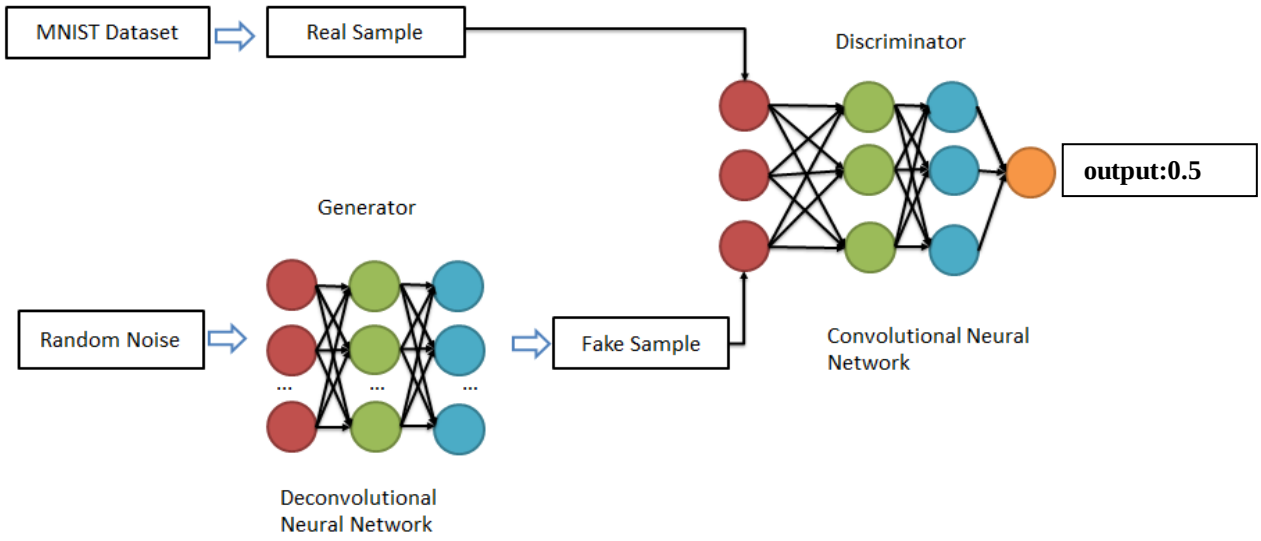

Figure 2.1: Proposed System

2.2 Description of Target users

- The solution should not be restricted to single dataset.
- It should be applicable to all GAN models which are generating biased images.

2.3 Advantages/applications of the proposed system

- Images are generated of various class.
- Its application is widely seen in the world of fashion, advertising, generating anime and cartoon characters.
- It should be applicable to all GAN models which are generating biased images.
- Reduces the use of human resource.

2.4 Scope

- GAN application is widely seen in the world of fashion, drug discovery, generating anime and cartoon characters.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Overview of SRS

A software requirements specification is the basis for your entire project. It lays the framework that every team involved in development will follow.

It's used to provide critical information to multiple teams — development, quality assurance, operations, and maintenance. This keeps everyone on the same page.

Using the SRS helps to ensure requirements are fulfilled. And it can also help you make decisions about your product's life cycle — for instance, when to retire a feature.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

Writing an SRS can also minimize overall development time and costs. Embedded development teams especially benefit from using an SRS.

3.2 Requirement Specifications

Software Requirements Specifications establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It also provides a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

3.2.1 Functional requirements

- The system shall be able to generate images of all classes.
- The system shall be able to train GAN model without getting into mode collapse.

3.2.2 Use Case diagram

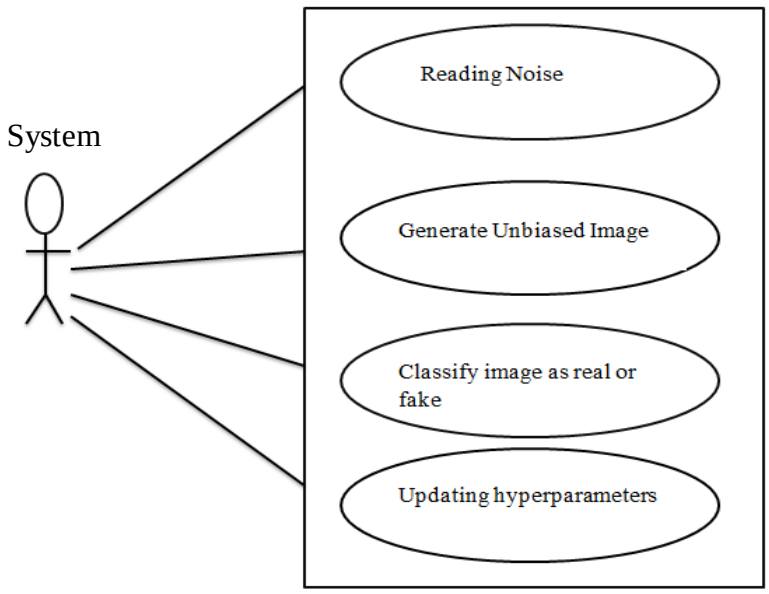

Figure 3.1:Use case diagram

3.2.3 Use Case descriptions using scenarios

Use case: Generating Unbiased Fake images

Actor: System

Pre-condition: Discriminator should be trained to classify real and fake images.

128 GB of RAM for training.

Post-condition: To Generate unbiased images and classify them as real

Success scenario:

1. Generator is fed with the noise (pixel values).
2. It generates unbiased fake images.
3. Discriminator classifies it as real.

Exception: Discriminator classifies it as real

3.2.4 Non-functional Requirements

- Dataset should be loaded within 10 ms.

3.3 Software and Hardware Requirement Specifications

Software requirements:

- Linux platform shall be used in building and training the system.
- Anaconda 3
- OpenCV
- Tensorflow

Hardware requirements:

- RAM :128GB
- Graphics:4GB
- GPU: RTX 2080 Ti/NVIDIA(multiple GPU for fast training)
- Hard drive >= 2TB
- CPU: > 2GHz; CPU should support the number of GPUs that you want to run.

4. SYSTEM DESIGN

4.1 Architecture of GAN

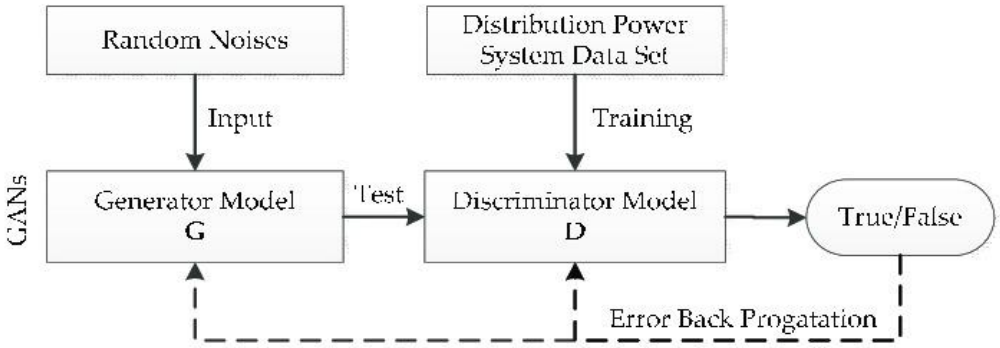

Figure 4.1: GAN Architecture Diagram

4.2 Class Diagram

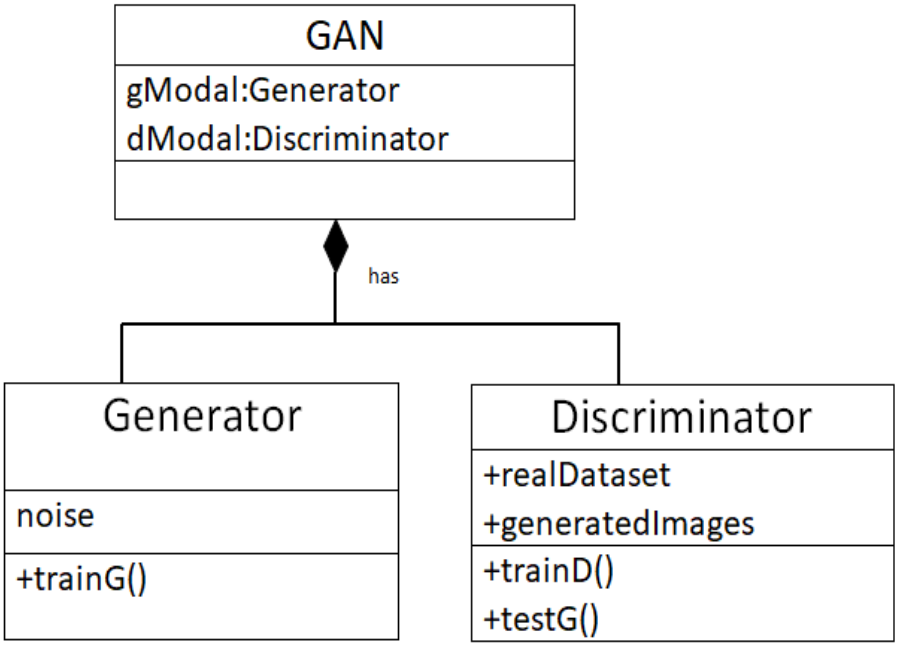

Figure 1.2: GAN Class Diagram

4.3 Flow Chart

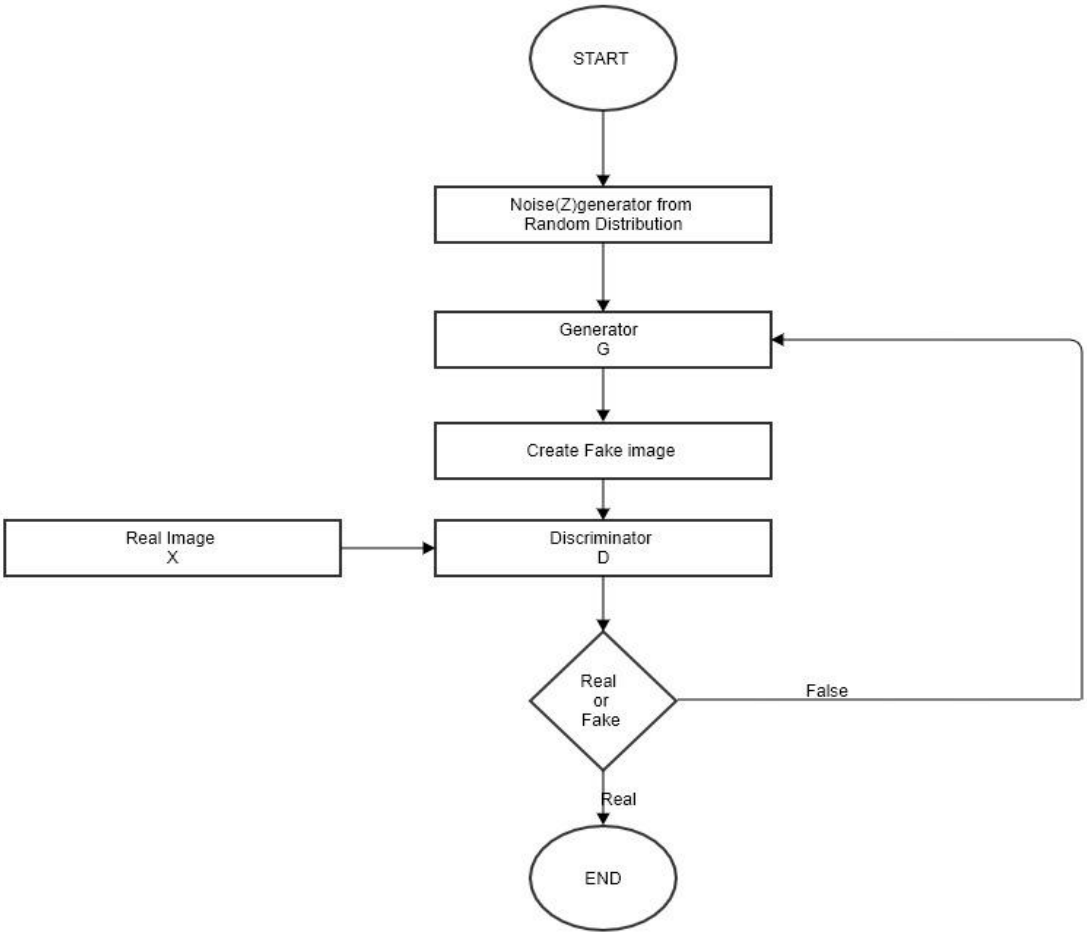

Figure 4.3: Flow Chart

4. IMPLEMENTATION

5.1 Proposed Methodology

Balance between the discriminator and generator

- The non-convergence and mode collapse is due to imbalance between the discriminator and the generator. The proposed solution balances their training to avoid overfitting.

Cost and image quality

- In a discriminative model, the loss measures the accuracy of the prediction and we use it to monitor the progress of the training. However, the loss in GAN measures how well it is doing compared with discriminative model. As, the generator cost increases the image quality is actually improving. We fall back to examine the generated images manually to verify the progress. This makes model comparison harder which lead to difficulties in picking the best model in a single run. We have made model comparison by evaluating four written codes as mentioned below.

Changes in neural network design

Number of epoch and batch size are 15000 and 400 respectively in all the four codes, this will result in 150 number of batches.

- Code 1: In discriminator, activation function used is LeakyRelu with Adam optimizer. In generator, activation function used is LeakyRelu, sigmoid. Binary cross entropy loss function is applied. This gave quite a good result with discriminator loss of 0.6, generator loss of 1.5 and accuracy of 82%.

- Code 2: In discriminator, activation function used is Relu with Adam optimizer. In generator, activation function used is LeakyRelu, sigmoid. Binary cross entropy loss function is applied. This gave discriminator loss of 0.2, generator loss of 1.1 and accuracy of 70%.

- Code 3: In discriminator, activation function used is LeakyRelu with Adam optimizer. In generator, activation function used is LeakyRelu, Tanh. Binary cross entropy loss function is applied. Additionally batch normalization is performed. The result was not up to the mark due to Tanh activation function. This gave discriminator loss of 0.9, generator loss of 0.2 and accuracy of 48%.

- Code 4: In discriminator, activation function used is LeakyRelu with RMSprop optimizer. In generator, activation function used is LeakyRelu, sigmoid. Mean squared error loss function is applied. This gave discriminator loss of 0.2, generator loss of 0.3 and accuracy of 70%.

5.2Description of Modules

1. Discriminator

Input: Image with one channel and 28×28 pixels in size. Real examples with a class label of one from MNIST English dataset, and randomly generated samples with a class label of zero.

Output: Binary classification, likelihood that the sample fed to it is real (or fake)

2. Generator

Inputs: Point in latent space, e.g. a 100 element vector of Gaussian random numbers.
Outputs: Two-dimensional square greyscale image of 28×28 pixels with pixel values in [0,1].

6. TESTING

6.1 Module/Unit Test Plan

Requirement ID	Test ID	Input	Expected Output	Actual Output
3.1	1	Epoch:100 Batch size:234 No. Of batches: 256 Discriminator: Activation function: LeakyReLu Optimizer: Adam Generator: Activation function: Leaky ReLu Sigmoid Loss function: BCE	Good image quality Generate images of all classes	Dloss: ~0.6 GLoss: ~1.5 Accuracy: 70%
3.1	2	Epoch:50 Batch size: 315 No. Of batches: 190 Discriminator: Activation function: ReLu Optimizer: Adam Generator: Activation function: ReLu Sigmoid Loss function: BCE	Good image quality Generate images of all classes	Dloss: ~0.2 GLoss: ~1.1 Accuracy: 48%
3.1	3	Epoch:100 Batch size:234 No. Of batches: 256 Batch normalization Discriminator: Activation function: Leaky ReLu Optimizer: Adam Generator: Activation function: Leaky ReLu Tanh Loss function: BCE	Good image quality Generate images of all classes	Dloss: ~0.9 GLoss: ~0.2 Accuracy: 82%

3.1	4	Epoch:100 Batch size:234 No. Of batches: 256 Discriminator: Activation function: Leaky ReLu Optimizer: RMSprop Generator: Activation function: Leaky ReLu Loss function: MSE	Good image quality Generate images of all classes	Dloss: ~0.2 GLoss: ~0.3 Accuracy: 70%

Table 6.1: Module/Unit Test Plan

7. RESULTS AND DISCUSSIONS

The desired output of the GAN on MNIST dataset should generate handwritten images of all classes from 0 to 9.

Figure 6 shows the output after 10 epochs in which the digits are not recognizable.

Figure 7.1: Images after 10 epochs

Output after training for 15000 epochs, which generates handwritten digits of all classes

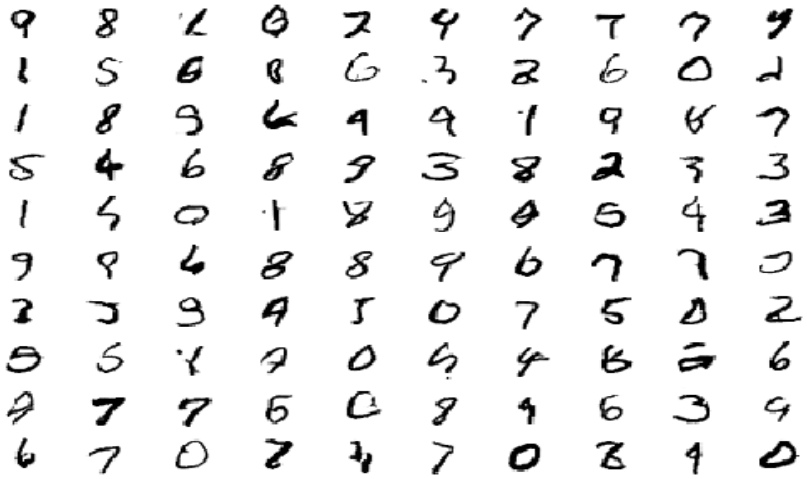

Figure 7.2:Output after 15000 epochs

8. CONCLUSION AND FUTURE SCOPE

We implemented GAN successfully on MNIST dataset by enhancing its stability by evaluating for four written codes and choosing the best among them. It gave us good result of handwritten digits from all the classes with no mode collapse.

The model can be implemented to any dataset which shall show good result with no problem of mode collapse.

9. REFERENCES/ BIBLIOGRAPHY

[1] Medium, https://towardsdatascience.com/understanding-generative-adversarial-networksganscd6e4651a29

[2] Deep Generative Models, https://towardsdatascience.com/deep-generative-models-25ab2821afd3 , 2018

[3] https://towardsdatascience.com/how-to-train-neural-network-faster-withoptimizersd297730b3713

[4]arxiv-Ian Goodfellow, https://arxiv.org/pdf/1406.2661.pdf, 2012

[5] Keep Calm and train a GAN. Pitfalls and Tips on training Generative Adversarial Networks , https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-ontraininggenerative-adversarial-networks-edd529764aa9 , 2018

[6] GAN — Why it is so hard to train Generative Adversarial Networks! https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generativeadvisory-networks-819a86b3750b , 2018

[7] GAN — What is wrong with GAN cost function https://medium.com/@jonathan_hui/gan-what-is-wrong-with-the-gan-cost-function 6f594162ce01

[8] GAN — A comprehensive review into the gangsters of GANs (Part 1) https://medium.com/@jonathan_hui/gan-a-comprehensive-review-into-the-gangstersof-gans-part-1-95ff52455672 ,2018

[9] GAN — A comprehensive review into the gangsters of GANs (Part 2) https://medium.com/@jonathan_hui/gan-a-comprehensive-review-into-thegangsters-of-gans-part-2-73233a670d19 ,2018

[10] GAN —What is generative-adversarial networks and its applications https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09

[11] GAN —DCGAN, https://medium.com/@jonathan_hui/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f

[12] GAN —Ways to improve GAN performance , https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b?

10. APPENDIX

[A] Gantt Chart

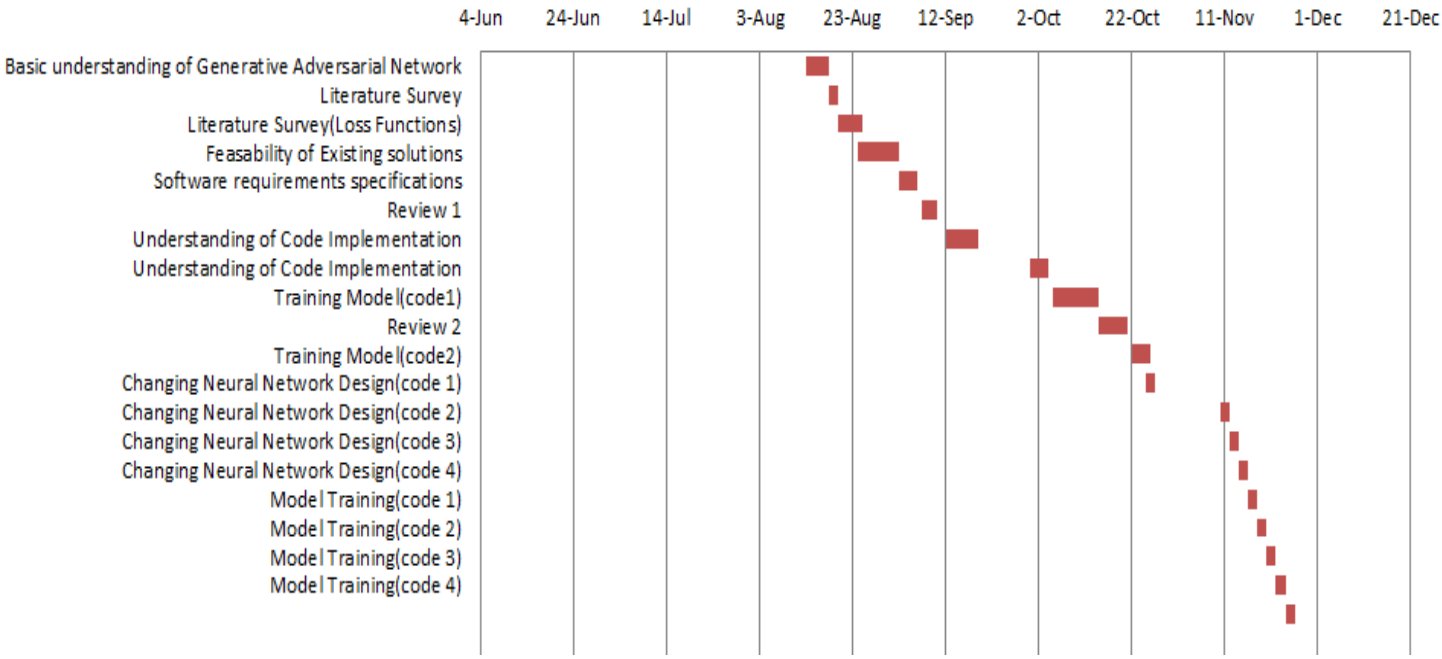

Figure 10.1:Gantt Chart