A

Project Report On

**High Energy Physics (HEP) Data Analysis with jet images using GANs Networks**

Developed at

# Physical Research Laboratory, Ahmedabad.

Developed by

**Patel Parth Nitinbhai - Department of IT, DD University**

## **Internal Guide: External Guide:**

## **Prof. (Dr.) V.K.Dabhi, Prof. Partha Konar,**

## **Head, associate professor,**

## **Department of InformationTech., Theoretical Physics Div.,**

## **Faculty of Technology, Physical Research**

## **DD University, Nadiad Laboratory, Gujarat.**



**Department of Information Technology**

**Faculty of Technology, Dharmsinh Desai University**

**College Road, Nadiad-387001**

**April – 2020**

**CANDIDATE’S DECLARATION**

I declare that the final semester report entitled “**High Energy Physics (HEP) Data Analysis with jet images using GANs Networks**” is my own work conducted under the supervision of the internal guide **Prof. Partha Konar** from **Physical Research Laboratory, Ahmedabad, Gujarat, 380009.**

I further declare that to the best of my knowledge, the report for B.Tech. Final semester does not contain part of the work which has been submitted for the award of B.Tech. Degree either in this or any other university without proper citation.

Also, I declare that the following students also worked on this project:

Patel Parth Nitinbhai – Department of IT, DD University

Candidate’s Signature:

Candidate's Name: Patel Parth Nitinbhai

Branch: IT

Student ID: 16ITUOS092

**DHARMSINH DESAI UNIVERSITY**

**NADIAD-387001, GUJARAT**



**CERTIFICATE**

This is to certify that the project entitled “**High Energy Physics (HEP) Data Analysis with jet images using GANs Networks**" is a bonafide report of the work carried out by **Mr Patel Parth Nitinbhai**, **Student ID No: 16ITUOS092** of Department of Information Technology, semester VIII, under the guidance and supervision for the award of the degree of Bachelor of Technology at Dharmsinh Desai University, Nadiad (Gujarat). They were involved in Project training during the academic year 2019-2020.

Following student was also involved in this project:

Patel Parth Nitinbhai – Department of IT, DD University.

Prof. (Dr.) V K Dabhi, (Project Guide)

Department of Information Technology,

Faculty of Technology,

Dharmsinh Desai University, Nadiad

Date:

Prof. (Dr.) V K Dabhi,

Head, Department of Information Technology,

Faculty of Technology,

Dharmsinh Desai University, Nadiad

Date:

**ACKNOWLEDGEMENT**

###### Every work that one completes successful stands on the constant encouragement, goodwill and support of the people around. I am hereby, avail this opportunity to express my heartfelt gratitude to a number of people who extended their valuable time, and cooperation in developing this project.

###### I would like to thank this opportunity to express my deep gratitude and sincere thanks to my respected internal guide Prof.(Dr.)V.K. Dabhi Sir and external guide Prof. Partha Konar (Associate Prof., Physical Research Laboratory, Ahmedabad, Gujarat.) provide me with all valuable guidance, encouragement, support and constructive criticism without which this project would not have been materialized.

I would also like to thanks to the other members of PRL who

have helped me in various aspects of deep learning as well as

in different aspects of High Energy Physics. The name of the members are:

1] Mr Ng. Vishal Singh, Senior Research Fellow, Theoretical Physics, PRL  
2] Ms Akanksha Bhardwaj, Senior Research Fellow, Theoretical Physics, PRL

With regards,

PATEL PARTH NITINBHAI

**TABLE OF CONTENTS**

Abstract…………………………………………………………………………………….i

List of Figures……………………………………………………………………………..ii

List of Tables……………………………………………………………………………...ii

Abbreviations……………………………………………………………………………..iii

**1. Introduction 1**

1.1 Introduction to Research Problem 1

1.2 Motivation for the Research Work……………………………………………..3

1.3 Objectives and Scope of the Research Work………………………...…………3

1.4 Technology and Literature Review…………………………….……………….4

1.5 Hardware and Software Requirements…………………………………………5

**2. Background Theory 6**

2.1 Large Hadron Collider(LHC)…………………………………………………..6

2.2 What is High Energy Physics……………..……………………………………7

2.3 Procedure……………………………………………………………………….7

2.4 Representing particle collision as images…..…………………………………..8

2.5 Datasets………...……………………………………………………………….9

2.6 Neural Network Architecture……..…………………………………………...10

**3. Review of Literature…….....…………………………………………………..11**

3.1 Synthesis of different research work…………...……………………………...11

**4. Analysis and Findings……….…………………………………………………15**

4.1 Experimental Setup......….……….……………………………………………15

4.2 Data……………………………………………………………………………16

4.3 Architectures(models Implemented)……………….…….……………………18

4.4 Parameters for Training a Network..……………….…….……………………31

4.5 Results after implementation……...……………….…….……………………32

4.6 Problems Faced During Implementation….……….…….……………………34

**5. Proposed Work……….…………………………………………………………36**

5.1 Loading dataset……......………….…………………………………………..36

5.2 Implementing the Architecture…….…………………………...…………….36

**6. Conclusion………………………………………………………………………40**

References…..………………………………………………………..………………….41

Experience……………………………………………………………..…………………42

Research Publication……………………………………………………………………..43

Curriculum vitae…………………………………………………………………………44

**ABSTRACT**

High Energy Physics (HEP) is also called "Particle Physics". In Compact Muon Solenoid (CMS) detector particles such as the proton, electron, gluon, etc. are allowed to collide with one another. As a result of the collision, new high energy particles which are called muons, quarks, etc. are generated. In order to study the nature and characteristics of the particles, deep learning models are being tested in recent time. This project focuses on detecting Top quark jets as anomalies from an overwhelmingly large background in the form of QCD jets. To achieve such categorisation in the language of machine learning, we investigate a few Generative Adversarial Network (GAN) models in this present study. In a typical computer science scenario, image generation and anomaly detection are quite familiar with the help of GANs. Furthermore, for obtaining the best results, Hyper-parameter tuning is also necessary by providing the range of various hyper-parameters. All the codes have been implemented on Google Colab and also tested on a workstation, consisting of GPU.

Keywords: Deep Learning - Machine Learning - Convolutional Neural Networks -

Particle Physics - OpenData - LHC - CMS

**LIST OF FIGURES**

Fig 1.1.1 Penetration of different particles........................................................................ 2

Fig 2.1.1 Large Hadron Collider(LHC)............................................................................. 6

Fig 2.6.1 CNN Architecture..............................................................................................10

Fig 3.3.7.1 ROC Curve….................................................................................................13

Fig 3.3.7.2 Specificity.......................................................................................................13

Fig 3.3.7.3 True Positive rate............................................................................................14

Fig 3.3.7.4 False Positive rate...…………………………….……………………………14

Fig. 3.3.7.5 Background and signal overlapping curve………………………………….14

Fig. 4.2.1.1 Spherical coordinate system to calculate Azimuthal angle and Pseudorapidity……………………………………………………………………………15

Fig. 4.2.2.1 Jet image…………………………………………………………………….17

Fig 4.3.1.1 Auto-Encoder Architecture …………………….……………..........……….17

Fig. 4.3.2.1 Algorithm of Generative Adversarial Network……………………………..19

Fig. 4.3.2.2 Generative Adversarial Networks(GAN) Architecture……………………..19

Fig. 4.3.2.3 Discriminator Architecture……………....………………………………….20

Fig 4.3.2.4 Generator Architecture …………………….…………….………………….20

Fig. 4.3.2.5 Comparison of Generator and Discriminator……………………………….21

Fig. 4.3.5.1 ANOGAN Architecture(similar to that of GAN)….………………………..24

Fig. 4.3.5.2 Stepwise selection of hyper-parameters…………………………………….25

Fig. 4.3.5.3 Efficient GAN Based Anomaly Detection(EGBAD) Architecture operating on BiGAN network………………………………………………………………………26

Fig. 4.3.5.4 Step wise Hyper-parameter selection……………………………………….27

Fig. 4.3.5.5 GANomaly Architecture……………..…….……………………………….28

Fig. 4.5.1 ROC curve Plot………..…….………………………….…………………….32

Fig. 4.5.2 d\_loss is the discriminator loss and below shown is the generator model accuracy………………………………………………………………………………….32

Fig. 4.5.3 Plot of Anomaly score between signal and background images…………..….33

Fig. 4.5.4 ROC curve values for using all 3 architectures……………………………….33

Fig. 5.2.2.1 Neural Network Flowchart………………………………………………….38

**LIST OF TABLES**

Table 4.1.3.1 Details about dataset............................................................................…….16

Table 4.4.1 Values of Hyperparameters in implementation code.....................................31

Table 4.5.1.1 Comparison of GAN and CNN models…………………….............…….34

**1: INTRODUCTION**

* 1. **INTRODUCTION TO RESEARCH PROBLEM: `**

High Energy Physics (HEP) is the study of the most fundamental building blocks of nature. The goal of high energy physics (also known as particle physics) is to determine the most fundamental building blocks of matter and to understand the interactions between these particles. The ATLAS experiments use colliding beams of protons to achieve the highest possible energies in an attempt to produce high mass fundamental particles which are not available in the present time universe.

The CMS(Compact Muon Solenoid ) and ATLAS (A Toroidal LHC ApparatuS) experiments are the two large experiments located on the ring of the Large Hadron Collider (LHC) at European Organization for Nuclear Research (CERN) situated at France–Switzerland border near Geneva. Most powerful particle accelerators make protons to collide in the experiment at the LHC. Recent researches and approaches have mostly focused on looking for physics beyond the standard model of particle physics by identifying different objects or particular particle types and their momenta and other properties. One of the primary objectives in any new physics search is by correctly identifying the background

events, which is by far dominating in most cases. Here we study the quantum chromodynamics (QCD) jets, which are formed starting from partons (quark or gluon) produced in the high energy collisions. In a typical hadron collider such as LHC, lots of such QCD jets are expected. In this present study, we look into the high transverse momenta (pt) jets which can be formed from the hadronic decay of some of the boosted heavy fundamental particles such as Higgs, W or Z bosons or top quark. Hence the classification of events in two categories: QCD Background images and top-quark signal images reserves the merit in finding a new scenario in the area of particle physics. Recently attention has focused on deep learning to improve sensitivity, as well as tackle the increase in detector resolutions and data rates in HEP experiments.

In this study, we try to distinguish highly energetic QCD jets from top-quark jets. Events are generated in Madgraph5 and showered with Pythia8. We pass the events through light detector simulation software Delphes3. From here, we cluster jets with R=1.0 and minimum transverse momentum of 800 GeV with the anti-kt algorithm in the E-scheme, using the tower constituents using FastJet package. The hardest (most energetic) jets from events having them within the range between 800 GeV and 900 GeV are selected. These jets are then binned in the (Eta, Phi) plane taking the jet centre as origin in 32 x 32 bins. These so-called jet images are the ones used to train and validate our results.

**1.2 MOTIVATION FOR RESEARCH WORK:**

Identification of jets coming from heavy boosted particles and highly energetic light quarks is a very active area of research in the high energy physics community. From a machine learning perspective, jet-images are used to train image-based neural networks to classify jets in a supervised learning paradigm. There are also works with unsupervised learning algorithms to train networks like an autoencoder to identify jets from highly boosted heavy particles by treating them as an anomaly over the huge QCD jets background. In this work, we focus on using another unsupervised learning model: GAN, on identifying top quark jets, by treating them as anomalies. Even though most popular uses of GAN involve the generation, we concentrate on the discriminator. The top quark jets are treated as fake QCD images.

* 1. **SCOPE AND OBJECTIVE OF THE RESEARCH WORK**

**SCOPE:**

Moreover, anomaly detection is also implemented using GANs. One can use Signal and Background images that can be generated from the simulated software. CMS detector can be used further to take the images. But due to time violation, GANs can be used by implementing various architectures such as GANomaly, ANOGAN and EGBAD. With the help of these three architectures, we can perform anomaly detection. Depending upon the problem and dataset, each among these three architectures can give best results. In our case of jet images, GANomaly architecture gave the best results as by measuring the area under the curve. Further, one can implement some other architecture and may obtain optimum results.

**OBJECTIVE:**

In our project of "High Energy Physics Data Analysis of Jet images using GAN" we implemented Auto-Encoder architecture with hybrid convolution neural network to generate the images as well as identify the images. These images of the jet are corresponding energy deposits in the calorimeter, which are generated by using the simulated tools for the LHC collision, event production and detection.

The main purpose of this project is to find the anomalous image from the given sample of images. For this purpose, we implement the Anomaly Detection using GANs to find the signal in the form of an anomaly from the admixture of QCD background and signal images. Earlier, there was also the case with the generation of images, and that is by implementing different models of GANs (e.g. DCGAN, Adversarial Autoencoder, ACGAN). Moreover, after performing anomaly detection, we have implemented the code for Hyper-Parameter tuning using the software AUPTIMIZER(discovered by LG Company). Various hyper-parameters such as learning rate, momentum, batch size, epochs, etc. are optimized by taking the input range of these hyper-parameters.

* 1. **TECHNOLOGY AND LITERATURE REVIEW**
* **Deep learning:** 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.
* **Autoencoder:** An Autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. An Autoencoder aims to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal “noise”.
* **Convolution neural network:** In deep learning, a convolution neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics.
* **Keras –Python machine learning library:** Keras is an open-source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

##### **1.5HARDWARE AND SOFTWARE REQUIREMENT**

##### **Google Colab:**

##### Google Colab is a free cloud service, and now it supports free GPU which can be tried to improve Python programming language coding skills. Also, one can develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch and OpenCV.

1. **Jupyter notebook:**

##### Project Jupyter is a nonprofit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages".

1. **Workstation:**
   * We have been allotted a workstation that consists of a GPU and a CPU. The specification of the GPU is 8 GB GRAM, NVIDIA Quadro M4000. The operating system on the workstation was RHEL 7.

**2: BACKGROUND THEORY**

## **2.1 LARGE HADRON COLLIDER(LHC)**

The Large Hadron Collider (LHC) is the largest and perhaps the most powerful particle accelerator in the world. The Execution of the LHC program started on 10 September 2008. The LHC consists of a 27-kilometre ring of superconducting magnets with a number of accelerating structures to boost the energy of the particles (protons) along the way. The high-energy protons, in bunches, are made to circulate at speed very close to the speed of light inside the ultrahigh vacuum beam pipe. Protons are then collided after coming from the opposite direction at a few interacting points where detectors are situated. The travel direction of the particle is called the beam axis. The strong magnetic field does acceleration and bending of the beam by the surrounded superconducting magnets. The electromagnets are built from coils of special electric cable that operates in a superconducting state, efficiently conducting electricity without resistance or loss of energy. This requires chilling the magnets to -271.3°C – [a temperature colder than outer space](https://home.cern/about/engineering/cryogenics-low-temperatures-high-performance).



Fig. 2.1.1 Large Hadron Collider (LHC)

Major parts of the accelerator are connected to a distribution system of liquid helium, which cools the magnets, as well as to other supply services. A large number of magnets of different varieties and sizes are used to direct the beams inside the accelerator. These include 1232 dipole magnets 15 meters in length which bend the beams, and 392 quadrupole magnets, each 5–7 meters long, which focus the beams. Just before the collision, a different kind of magnet is used to squeeze the particles closer together in order to increase the collision rate. The particles are so small that the activity of making them collide is similar to firing two needles 10 km apart that they meet halfway between them with high precision.

All the controls for the accelerator, its services and technical infrastructure are monitored under one roof at the CERN Control Centre. From the CERN Control Center, the particles are made to collide at four different locations around the accelerator ring, corresponding to the position of 4 particle detectors namely: ATLAS, CMS, ALICE and LHCb.

**2.2 WHAT IS HIGH ENERGY PHYSICS(HEP)?**

High Energy Physics (HEP) is the study of the most fundamental building blocks of nature. It is a branch of physics that studies the nature of fundamental particles that makes up all the matter and radiation of the universe. Here the word particle refers to the point object which does not have substructure or which is not made of some more fundamental object (e.g. [protons](https://en.wikipedia.org/wiki/Protons), electron, quark, gluon, Higgs etc.). Particle physics studies the properties of them and how different particles interact in a way so that three of the four fundamental forces (except gravity) can be explained and unified at the quantum level. The currently dominant theory which explains the fundamental particles and their fields, along with their dynamics, is known as the Standard Model. In 1970, after the discovery of the quarks particle, this model came into implementation. By extracting the parameters of the Standard Model, the work supports the limits of the standard model, and hence it will enhance our perception about nature's building blocks. Those efforts are proved to be quite accurate descriptions of our nature. Strong interactions describe the interaction between quarks and gluons and are described in quantum chromodynamics (QCD). Another effort in theoretical particle physics is string theory. String Theorists aims to construct a brief description of quantum mechanics and general relativity by building theory based on small strings.

**2.3 PROCEDURE:**

There are various Detectors, such as CMS, ATLAS, etc. are used in this experiment. Take for example, inside Compact Muon Solenoid(CMS) detector particles such as proton, electron, gluons, etc. are allowed to collide with one another. Due to this proton-proton collision, new particles are generated. This proton-proton collision takes place at a very high speed and inside the Large Hadron Collider(LHC), which is approximately 27km long. The CMS detector will identify the particles after the collision. As a result of the strong collision energy, new heavy particles (whichever exist in nature) can be produced following the way these fundamental particles interact.

The axis of collision is called "Beam Axis" where the proton-proton collision takes place. These High Energy particles travel further and strike the grid of the LHC detector. A Square grid is fixed to the inner layer of the detector. Now, when these particles strike the grid, the detector will calculate the approximate Energy of that particle (Since a sensor is provided in the detector for sensing energy). Corresponding to the range of the energy deposited, we represent by the colour of that particular grid.

**2.4 REPRESENTING PARTICLE COLLISION AS IMAGES**

Data from the surface of the cylindrical detector is represented as a 2D image with coordinates corresponding to azimuthal angle phi and pseudorapidity eta. For the pixel intensity in this image, we use the transverse momentum deposited in the combined calorimeter. We take the constituents of each jet and translate the (Eta, Phi) coordinates such that the origin is at the jet centre. This is effectively giving the coordinates in terms of the difference in terms of each jet rather than them depending on the exact location in space which is not useful information for our current analysis. These are then binned in the range (-1.6, 1.6) in both eta and phi, in an image of 32 x 32, thus giving a resolution of 0.1 x 0.1. This resolution is close to the experimental granularity of the calorimeter in the central region (0.08 x 0.08). This is reasonable because we are choosing jets which fall within the accepted central regions having an absolute value of eta less than 2.0.

**2.6 NEURAL NETWORK ARCHITECTURE:**

We perform binary classification on the resulting images using a Convolution Neural Net (CNN) comprising 3 convolution+pooling units (or 4 for larger images) with rectified linear unit (ReLU) activation functions. The output from these layers is directly given to the input to the fully connected layers. During this time Softmax function is applied to map signal and background probability distribution over a predicted output class. We use softmax with cross-entropy as the loss function and the ADAM optimizer. The network employed is shown in the figure.



Fig. 2.6.1 CNN Architecture

Source:https://arxiv.org/abs/1711.03573

**3: REVIEW OF LITERATURE**

**3.1 SYNTHESIS OF DIFFERENT RESEARCH WORK**

**3.1.1 Deep Neural Networks for Physics Analysis on low-level whole-detector**

**Data at the LHC:**

**Author:** Wahid Bhimji

**Year:** 2017

**Title:** Deep Neural Networks for Physics Analysis on low-level whole-detector

data at the LHC.

**Aim:** Finding the dataset and the neural network to be implemented, and CNN architecture has been fully explained in detail.

**Method:** Convolution neural network(CNN).

**Dataset:** Images containing QCD background for training and testing.

**Conclusion:** Jet image classification is done with the help of the CNN network. Additionally, the generation of the image is also done.

**3.3.2 A Survey on GANs for Anomaly Detection:**

**Author:** [Federico Di Mattia](https://arxiv.org/search/cs?searchtype=author&query=Di+Mattia%2C+F), [Paolo Galeone](https://arxiv.org/search/cs?searchtype=author&query=Galeone%2C+P), [Michele De Simoni](https://arxiv.org/search/cs?searchtype=author&query=De+Simoni%2C+M), [Emanuele Ghelfi](https://arxiv.org/search/cs?searchtype=author&query=Ghelfi%2C+E)

**Year:** 2019

**Title:** A Survey of GANs on Anomaly Detection.

**Aim:** Implementing various Techniques, i.e. brief description about methods of anomaly detection.

**Method:** Generative Adversarial Networks(GAN).

**Dataset:** Jet images of QCD background and top-quark signal.

**Conclusion:** Successfully performed Anomaly detection for MNIST and CIFAR10 dataset using various techniques, including GAN architecture.

**3.3.3 Semantic Image Inpainting with Deep Generative Networks:**

**Author:** Raymond A. Yeh

**Year:** 2017

**Title:** Semantic Image Inpainting with Deep Generative Networks.

**Aim:** Implementation of ANOGAN Technique with MNIST and CIFAR10 Dataset and also with jet image dataset.

**Method:** Generative Adversarial Networks(GAN).

**Dataset:** Jet images of QCD background and top-quark signal.

**Conclusion:** Successfully performed Anomaly detection for MNIST and CIFAR10 dataset using various techniques including ANOGAN architecture and this architecture has also been implemented on the dataset with jet images.

**3.3.4 Efficient GAN Based Anomaly Detection:**

**Author:** Houssam Zenati

**Year:** 2019

**Title:** Efficient GAN Based Anomaly Detection.

**Aim:** Implementation of EGBAD Technique with MNIST and CIFAR10 Dataset and also with jet image dataset.

**Method:** Generative Adversarial Networks(GAN).

**Dataset:** Jet images of QCD background and top-quark signal.

**Conclusion:** Successfully performed Anomaly detection for MNIST and CIFAR10 dataset using various techniques including EGBAD architecture and this architecture has also been implemented on the dataset with jet images.

**3.3.5 GANomaly: A Semi-Supervised Anomaly Detection via Adversarial training:**

**Author:** Samat Akcay

**Year:** 2018

**Title:** A Semi-Supervised Anomaly Detection(GANomaly).

**Aim:** Implementation of GANomaly Technique with MNIST and CIFAR10 Dataset and also with jet image dataset.

**Method:** Generative Adversarial Networks(GAN).

**Dataset:** Jet images of QCD background and top-quark signal.

**Conclusion:** Successfully performed Anomaly detection for MNIST and CIFAR10 dataset using various techniques including GANomaly architecture and this architecture has also been implemented on the dataset with jet images.

**3.3.6 Pros and Cons of GAN Evaluation Measures:**

**Author:** Ali Borji

**Year:** 2018

**Title:** Pros and Cons of GAN Evaluation Measures.

**Aim:** Evaluating and Measuring how good our GAN model is, by calculating Inception Score(IS) and Frechet Inception Distance(FID).

**Method:** GANs and Inceptionv3 model.

**Dataset:** Jet images of QCD background and top-quark signal.

**Conclusion:** By calculating IS and FID, we can get to know the performance of the model.

**3.3.7 Receiver Operating Characteristic(ROC) Curve:**

**AUC-ROC curve:** ROC is a probability curve, and AUC represents the area under the curve(how much area the graph has covered). It tells how much our model is able to bifurcate between two classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s, and if the AUC is lower, then it will not be able to predict properly. The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis as shown in the below figure.

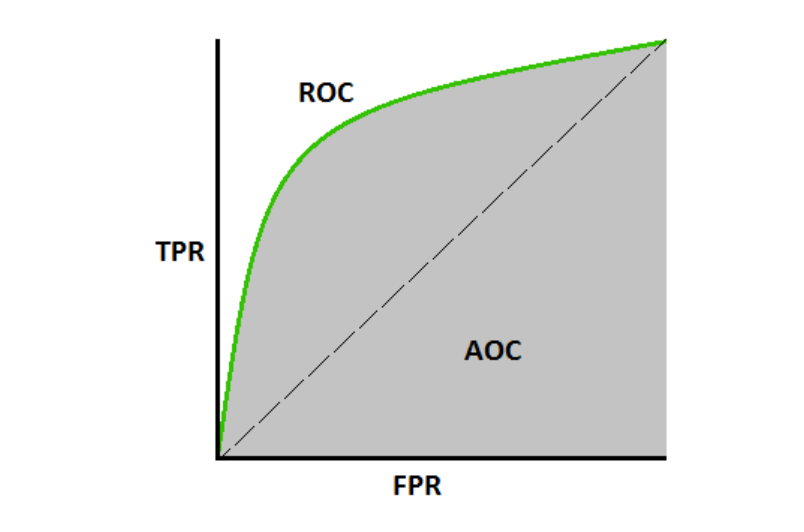


Fig. 3.3.7.1 ROC Curve

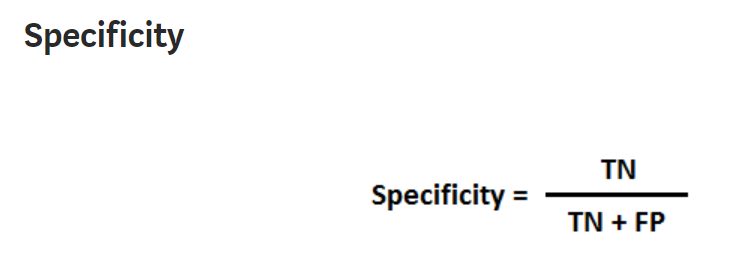


Fig. 3.3.7.2 Specificity

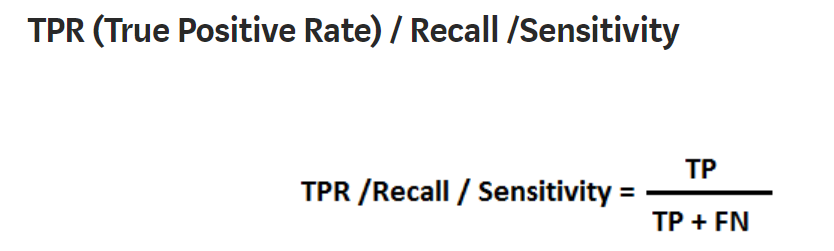


Fig. 3.3.7.3 True Positive Rate

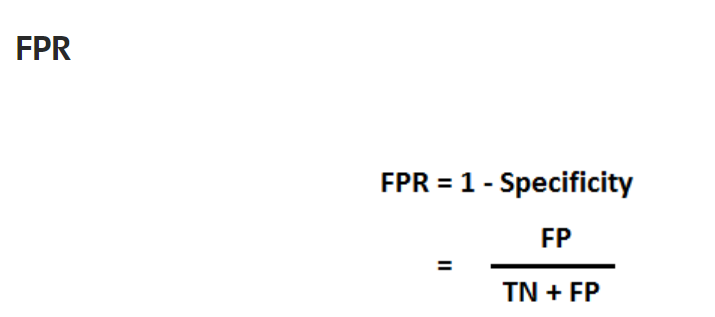


Fig. 3.3.7.4 False Positive Rate

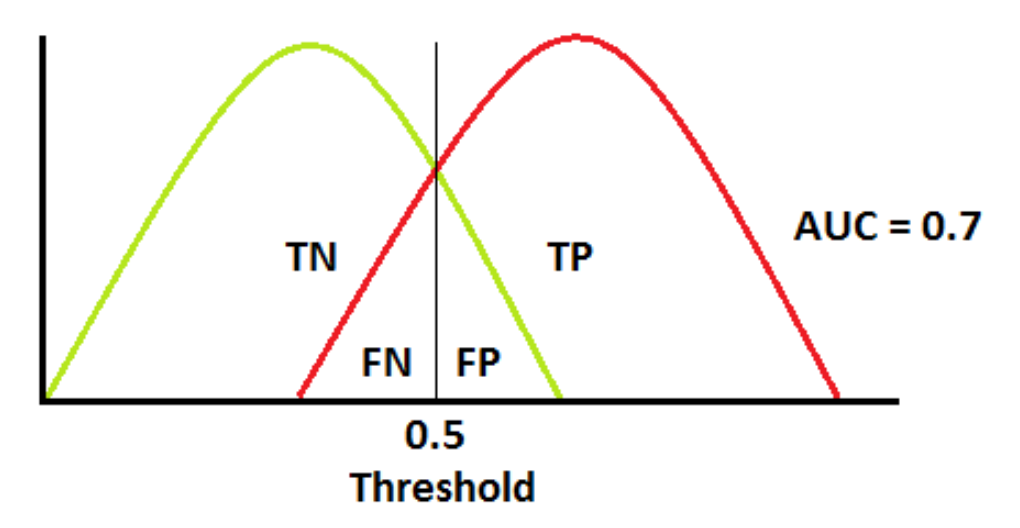


Fig. 3.3.7.5 background and signal overlapping curve

**4. ANALYSIS AND FINDINGS**

**4.1 EXPERIMENTAL SETUP**

**4.1.1 Easy to Setup and Natural:**

All the programs and codes are executed in Google Colab. Moreover, we were working in a workstation, containing GPU and a RAM of 8GB. So, the execution of programs on the workstation very fast and without any difficulty. My implementation part consists of multiple files executing at a time. Furthermore, the data which we were using was huge and occupied lots of space. So, it is inevitable to implement it on the workstation.

**4.1.2 Running Environment:**

* Google Colab
* Jupyter Notebook
* Python 3.7 or higher

**4.1.3 Dataset:**

The dataset consists of jet particle images. A **jet** is a narrow cone of hadrons and other particles produced by the hadronization of a quark or gluon in a **particle physics** or heavy-ion experiment.

|  |  |
| --- | --- |
| **Number of classes** | 2 (Background and Signal) |
| **Number of images in each class** | Background  It consists of 80,000 images in the train dataset and 10,000 images in the test dataset.  Signal  It consists of 10,000 images in the test dataset. |
| **Image Size** | 32 x 32 |
| **Number of channels** | 3 |
| **Total number of images** | Background set has 1,08,887 images.  The signal set has 90,000 images. |
| **Size of Raw Data** | 1GB |
| **Classification** | Binary classification |
| **Raw Data Format** | Pickle |
| **Preprocessing** | No |
| **Format (DataType)** | A dictionary containing NumPy array samples. |
| **Brief Description** | It consists of the images of jet particles. Background dataset consists of QCD(Quantum chromo-dynamics) jet images and the Signal dataset consists of top-quark jet images. |

Table 4.1.3.1 Details about the dataset.

**Training Dataset:** This dataset consists of only Background(QCD jet) images because as a part of Anomaly detection, in order to detect anomalous images such as signal(in this case) Generative Adversarial Networks(GANs) are only trained with one dataset. It consists of 90,000 samples of background. Each image was of size 32 x 32. In the implementation part, we define the number of training steps by giving the epoch value. It will iterate the training set for a number of epoch values. It is important to have a balanced dataset (i.e. having an equal number of images in each class) so that the abundance of a class is not a major factor when classifying new images (i.e. always predicting the more abundant class).

**Testing Dataset:** This dataset consists of both Background(QCD jet) images as well as Signal(top-quark) images because once a model is trained, it can be tested on both sets of images. As a result, the background (non-anomalous) image will not be detected. But when Signal images are tested, they can be easily detected as the model is trained for background images. It consists of 10,000 images of the background set and 10,000 images of the signal set, making a total of 20,000 images in the dataset. Each image was of size 32 x 32.

**4.2 DATA**

**4.2.1 Data Collection and Analysis:**

The collision events at the LHC produce millions of fundamental particles which eventually deposit their energy inside different layers of calorimeter in the dedicated detectors such as CMS and ATLAS. These deposited energies in different cells can be represented as an image of such an event.

Collisions at the LHC produce fundamental particles which can further decay into lighter particles and finally yield millions of fundamental particles to interact with different parts of the detector such as, tracking system, calorimeters, muon system, etc. These are also known as events, recorded in a HEP experiment by a detector like CMS, ATLAS. Final observables are described by a set of variables measured corresponding to the particles detected, e.g. the momenta and energy of muons, electrons, photons or QCD jets from the hadrons. These energy deposit data from the surface of the cylindrical detector can be represented as a 2D image-map with coordinates corresponding to **the azimuthal angle** and **pseudorapidity**.

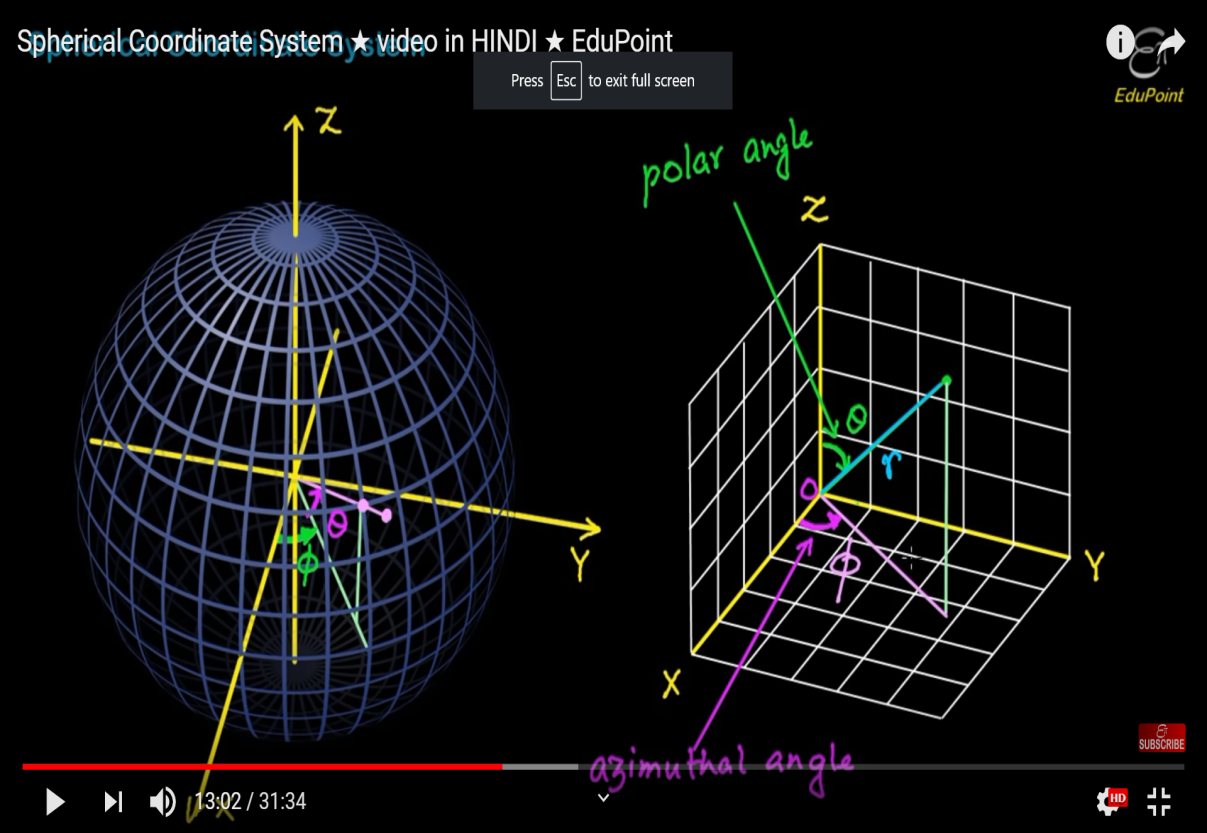


Fig. 4.2.1.1 Spherical coordinate system to calculate Azimuthal angle and Pseudorapidity

Each particle or physics object is represented as a circle with circumference having radius proportional to the particle’s energy, and the position of the particle on the canvas is decided on the basis of the momentum direction. The momentum direction use as coordinates the pseudorapidity η, related to the polar angle, and the azimuthal angle ϕ, which are standard choices in experiments with cylindrical symmetry.

The momentum of a particle consists of 3 things: Azimuthal angle(phi), Transverse momentum and pseudorapidity(eta). In most of the cases, there is a special axis defined through the geometry of particle physics experiments called the beam axis - the axis parallel to the incoming beams or the axis of the particle movement. In experiments, this axis is uniquely defined. However, usually, this beam axis is chosen to be the z-axis. Therefore, it is often convenient to describe four moments of particles by their energy and by two angles θ and φ. Here the "polar" angle θ describes the angle of a particle with respect to the z-axis, and φ is the "azimuthal" angle around the beam axis or angle from x-axis. Transversal Momentum (PT) and Phi (also known as the Azimuthal or Scattering Angle and is measured in radians) are both represented on the xy-plane. Phi is the angle between the x-axis and the ray PT. Transverse momentum (PT) is the momentum perpendicular to the path of the colliding particles. Pseudorapidity (η) is a special graphical coordinate describing the angle of a particle in relation to the particle beam (or z-axis). Pseudorapidity (eta) is calculated from an xz-plane, with the z-axis being the particle beam axis. If one were to take a section of the CMS detector to analyze a collision of particles, it could be divided into two planes: the "transverse" (xy-plane) and the xz-plane. The x and y axis are the same as any average graph with the x being the horizontal axis and the y the vertical axis. The xz-plane includes the z-axis or beam-axis, which is the path the particles of collision of the particles, and the x-axis, which connects both planes.

**4.2.2 Datasets:**

Images are in the form of jets. A **jet** is in shapes like a cone of [hadrons](https://en.wikipedia.org/wiki/Hadrons) and other particles produced by the [hadronization](https://en.wikipedia.org/wiki/Hadronization) of a [quark](https://en.wikipedia.org/wiki/Quark) or [gluon](https://en.wikipedia.org/wiki/Gluon) in particle [physics](https://en.wikipedia.org/wiki/Particle_physics). Particles which possess a colour charge, such as quarks, cannot exist in a free state because of [QCD](https://en.wikipedia.org/wiki/Quantum_chromodynamics) [confinement](https://en.wikipedia.org/wiki/Color_confinement) which only allows for colourless states. The dataset consists of 2 classes of data: background and signal. **Background** dataset consists of quarks and gluons. **The signal** dataset consists of images of top-quark particles, further decomposing into w-bozon and bottom quark particles. These w-bozon particles further decompose into lighter quarks(up, down, strange, charm).

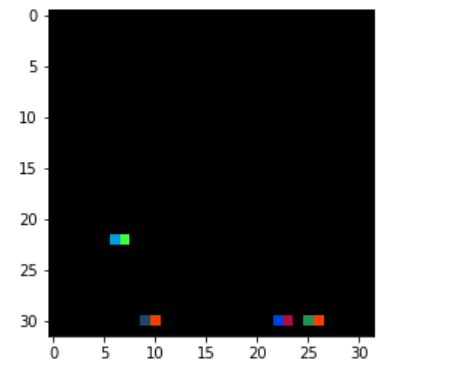


Fig. 4.2.2.1 Jet image

* 1. **ARCHITECTURES(MODELS IMPLEMENTED)**

**4.3.1 Auto-Encoders:**

**What are Autoencoders?**

An autoencoder [neural network](https://www.edureka.co/blog/neural-network-tutorial/) is an [unsupervised machine learning](https://www.edureka.co/blog/what-is-machine-learning/) algorithm that applies backpropagation and generates images which are similar to that of inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation which is called latent space representation and will try to generate the image which acts like the same to the input. The Autoencoder basically compressed the input data, and if anyone needs the original data, they can reconstruct it from the compressed data.

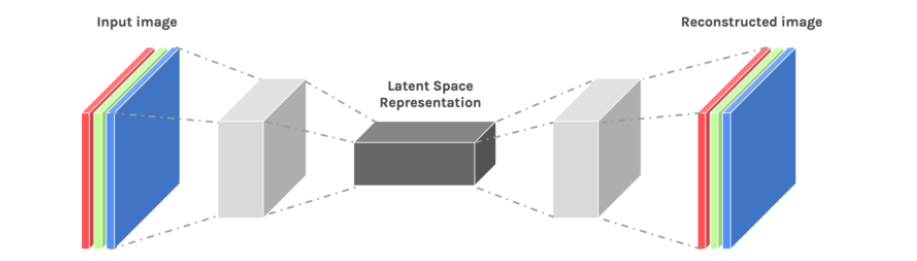


Fig. 4.3.1.1 Auto-Encoder Architecture

The Autoencoder Architecture consists of 3 parts:

* **Encoder:** This part of the network compresses the input into a **latent space representation**. The encoder layer converts the input image into a compressed representation. The dimensionality of the compressed form is less than the dimensionality of the original image. The compressed image is represented by letter z
* **Code:** This part of the network represents the compressed input which is fed to the decoder. This part is also called the Bottleneck layer, and it stores the latent space representation.
* **Decoder:** This layer **decodes** the compressed image back to the original dimension. The decoded image is a lossy reconstruction of the original image, and it is reconstructed from the latent space representation. The newly formed image may be the distorted version of the original one.

The layer between the encoder and decoder, i.e. the code is also known as **Bottleneck**. It does this by balancing two criteria:

* The compactness of representation, measured as the compressibility.
* It regains some important features of the original input data

**Properties of Autoencoders:**

* **Data-specific**: The Autoencoders are trained on some data, and during training, the weights of encoder and decoder are updated. After the testing samples are tested, and they are compressed according to the weights which are updated in training.
* **Lossy:** The decompressed outputs will be degraded compared to the original inputs. The reconstructed output is a distorted version of original data.
* **Learned automatically from examples:**It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

**Hyperparameters of Autoencoders:**

There are four hyperparameters that we need to set before training an autoencoder:

* **Code size**: It represents the number of nodes in the middle layer. Smaller size results in more compression.
* **The number of nodes per layer**: The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder. The decoder is symmetric to the encoder in terms of the layer structure.
* **Loss function:** We either use mean squared error or binary cross-entropy. If the input values are in the range [0, 1] then we typically use cross-entropy; otherwise, we use the mean squared error.

**4.3.2 Generative Adversarial Networks(GANs):**

Generative Adversarial Networks (GAN) is one of the most promising recent developments in Deep Learning. [GAN](http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf), introduced by Ian Goodfellow in 2014, introduces a method of unsupervised learning by training two deep networks, called Generator and Discriminator, which compete and cooperate with each other. In the course of training, both networks are trained one after the other and after training, testing will be done appropriately.

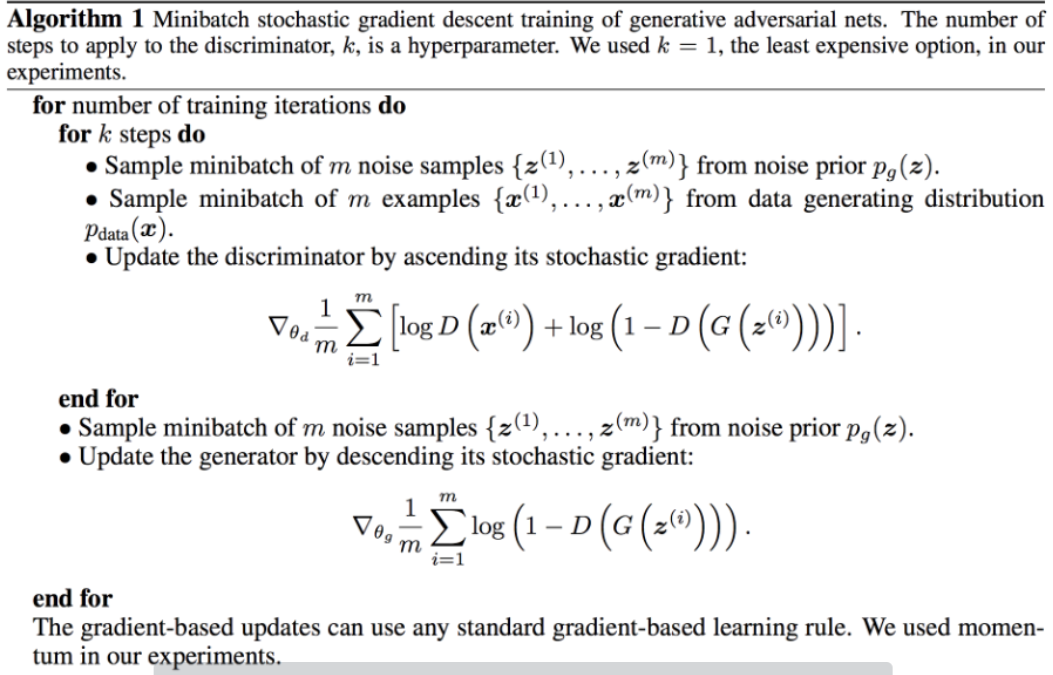


Fig. 4.3.2.1 Algorithm of Generative Adversarial Networks

For example, GAN is almost always explained like the case of a counterfeiter (Generative) and the police (Discriminator). Initially, the counterfeiter will show the police a fake currency. The police declare it as fake, which in turn provides the feedback to the counterfeiter why the currency is fake. The counterfeiter attempts to make a new fake currency based on the feedback it received. The police say the currency is still fake and offers a new set of feedback. The counterfeiter attempts to make a new fake currency based on the latest feedback. The cycle continues indefinitely until the police are fooled by the fake currency because it looks (almost) real.

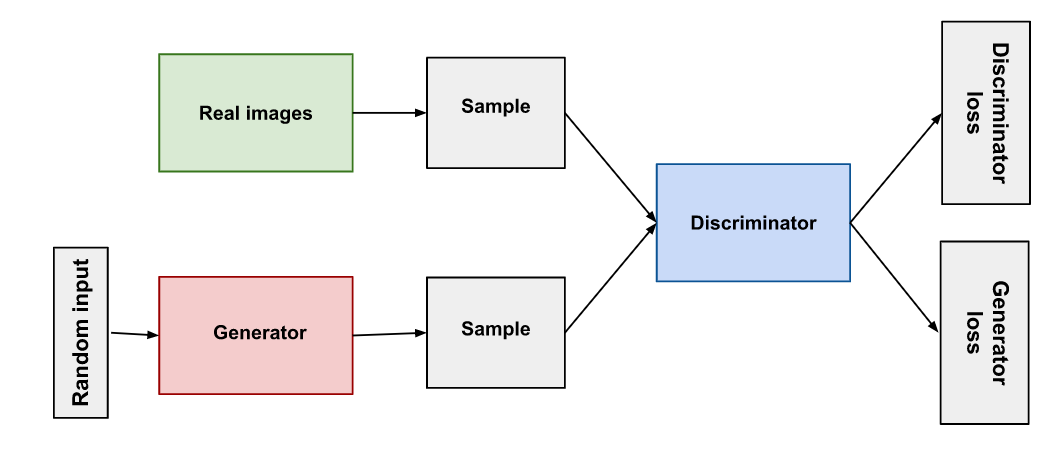


Fig. 4.3.2.2 Generative Adversarial Networks(GAN) Architecture

**Discriminator**

A discriminator is basically a deep Convolutional Neural Network (CNN), and it tells how real an image is, as shown in Figure. For MNIST Dataset, the input is an image (28 height x 28 width x 1 channel). The Sigmoid output is the probability value between 0 and 1(0.0 is certainly fake, 1.0 is certainly real, anything in between is the predicted output). The difference from a typical CNN is the absence of max-pooling in between layers. The activation function used in each Convolution layer is a leaky ReLU. A dropout between 0.4 and 0.7 between layers prevents overfitting and memorization. Overall, the Discriminator will convert 784-dimensional input into a 1-dimensional output.

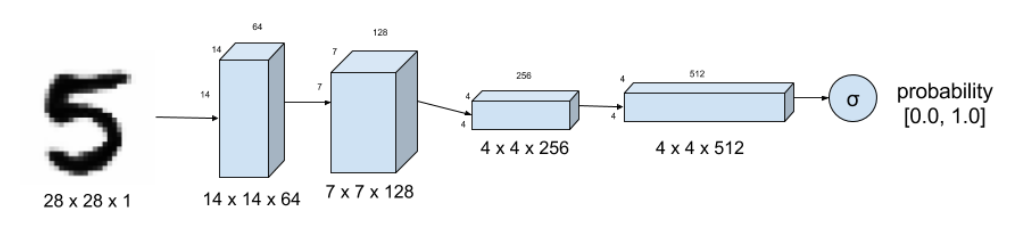


Fig. 4.3.2.3 Discriminator Architecture

**Generator**

The generator synthesizes fake images. The fake images are generated from a 100-dimensional noise (uniform distribution between -1.0 to 1.0) using the inverse of convolution, called transposed convolution. Instead of fractionally-strided convolution as suggested in DCGAN Architecture. In between layers, batch normalization is used to stabilize learning and prevent overfitting. The activation function after each layer is a ReLU. The output of the sigmoid at the last layer produces the fake image. A Dropout of between 0.3 and 0.5 at the first layer prevents overfitting. The Generator will convert 100 dimensional input into 784 dimensional output.

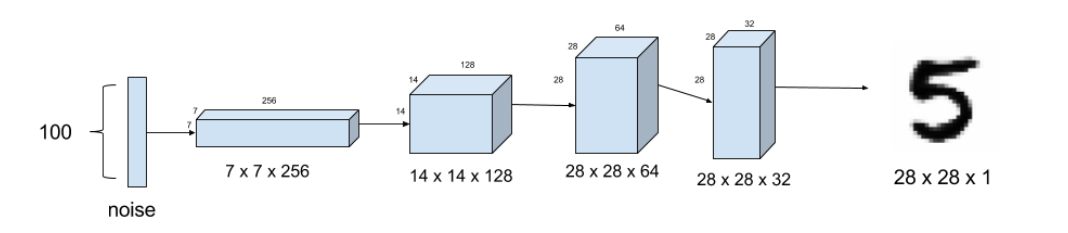


Fig. 4.3.2.4 Generator Architecture

**Discriminator Model**

The Discriminator model will apply strides and filters on the input, converting 784 dimensions into 1-dimensional output, as shown in the Discriminator Architecture figure. Since the output of the Discriminator is sigmoid, we use binary cross-entropy as a loss function. RMSProp as an optimizer generates more realistic fake images compared to Adam for this case. Learning rate is 0.001. Weight decay and clip value stabilize learning during the latter part of the training.

**Adversarial or Generator Model**

The adversarial model is just the generator-discriminator stacked together. The Generator part is trying to fool the Discriminator and learning from its feedback at the same time. It will generate the images, which look similar to the original image but, are actually fake ones. The training parameters are the same as in the Discriminator model except for a reduced learning rate and corresponding weight decay.

**Training**

Training is the hardest part. We determine first if the discriminator model is correct by training it alone with real and fake images. At that time, we are not training Generators. Afterwards, the Adversarial or Generator model is trained one after the other.

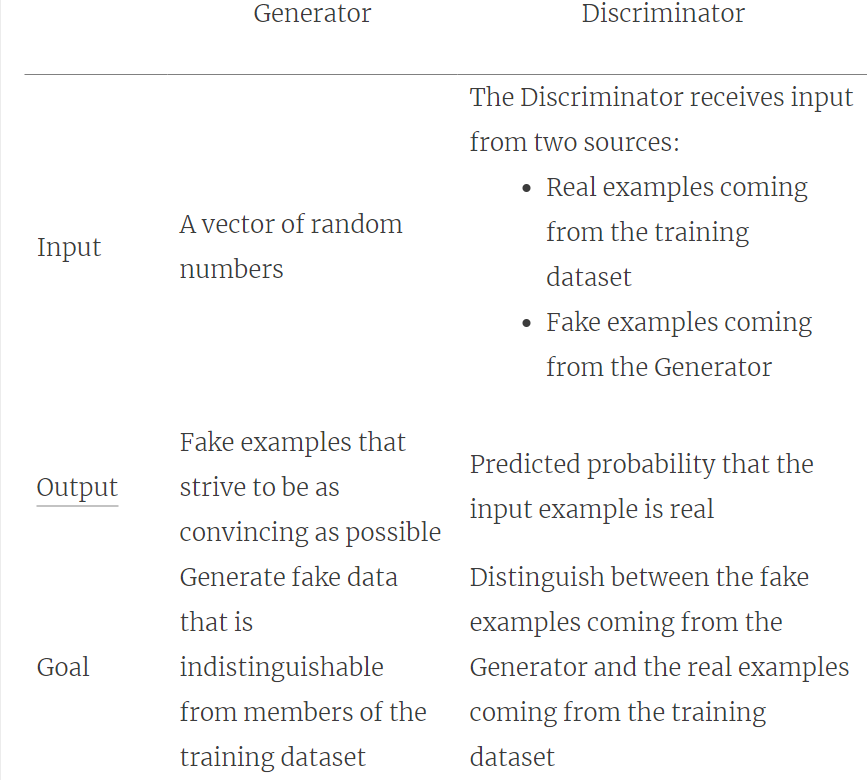


Fig. 4.3.2.5 Comparison of Generator and Discriminator

**4.3.3 Understanding the Generator**

Generator training requires tighter integration between the generator and the discriminator than discriminator training requires. The portion of the GAN that trains the generator includes:

-random input

-generator network, which transforms the random input into an original dimensional data

-discriminator network, which classifies the generated data

-discriminator output

-generator loss, which penalizes the generator for failing to fool the discriminator

In its most basic form, a GAN takes random noise as its input. The generator then transforms this noise into a meaningful output. By introducing noise, we can get the GAN to produce a wide variety of data, sampling from different places in the target distribution. Experiments suggest that the distribution of the noise doesn't matter much, so we can choose something that's easy to sample from, as a uniform distribution. To train a neural net, we alter the net's weights to reduce the error or loss of its output. In our GAN, however, the generator is not directly connected to the loss that we're trying to affect. The generator feeds into the discriminator net, and the discriminator produces the output we're trying to affect. The generator loss penalizes the generator for producing a sample that the discriminator network classifies as fake. This extra chunk of the network must be included in backpropagation. Backpropagation adjusts each weight in the right direction by calculating the weight's impact on the output — how the output would change if you changed the weight. But the impact of a generator weight depends on the impact of the discriminator weights it feeds into. So backpropagation starts at the output and flows back through the discriminator into the generator.

**4.3.4 Performance Evaluation of GAN model**

**Inception Score-** The Inception Score(IS), is a metric for evaluating the quality of generated images, specifically synthetic images output by generative adversarial network models. The probability of an image is predicted considering the number of classes and the class in which the image belongs. These predictions are then summarized into the inception score. The score seeks to capture two properties of a collection of generated images:

a] Quality of the image.

b] Diversity of image.

With the help of these two things, we can evaluate our model. The inception score has the lowest value of 1.0 and the highest value of the number of classes supported by the classification model.

**Frechet Inception Distance-** The Frechet Inception Distance score, or FID for short, is a metric that calculates the distance between feature vectors of the discriminator image and the generated image. The Frechet Inception Distance (FID) is a metric for evaluating the quality of generated images, and its major focus is to evaluate the performance of generative adversarial networks. The goal of developing the FID score was to evaluate generated images based on the statistics of a collection of generated images compared to the statistics of a collection of real images. These statistics are then calculated for the activations across the collection of real and generated images. A lower FID indicates better-quality images; conversely, a higher score indicates a lower-quality image and the relationship may be linear.

**4.3.5 A Survey on GAN for Anomaly Detection:**

Anomalies are patterns in data that do not conform to a well-defined notion of normal behaviour or a small fault in our original data that does not look similar to the input data. Generative Adversarial Networks (GANs) and the adversarial training framework have been successfully applied to model complex and high dimensional distribution of real-world data. This GAN characteristic suggests they can be used successfully for anomaly Detection, although their application has been only recently explored. Anomaly Detection using GANs is the task of analyzing the behaviour using the training process and for the detection of the anomaly. In their original formulation, the GAN framework learns a generator that maps samples from an arbitrary latent distribution (noise) prior to data as well as a discriminator which tries to distinguish between real and generated samples. A learned function that maps input data to its latent representation together with a function that does the opposite (the generator) is the basis of the anomaly detection using GANs.

There are three different approaches that are discussed:

1] ANOGAN implementation

2] EGBAD implementation

3] GANomaly implementation

**1] ANOGAN Approach:**

ANOGAN had an aim to use a standard GAN Architecture, trained only on positive samples so that it can learn a mapping from the latent space representation z to the realistic sample G(z). Here G represents the Generator and G(z) represents the output when z is given the input to the generator. It will use this learned representation to map new samples back to the latent space. Training a GAN on normal samples only makes the generator learn from the input samples. Now when the generator learns how to generate the input samples (due to Updation of weight by backpropagation), when an anomalous image is encoded its reconstruction will be non-anomalous. Therefore, the difference between the input and the reconstructed image will highlight the anomalies.

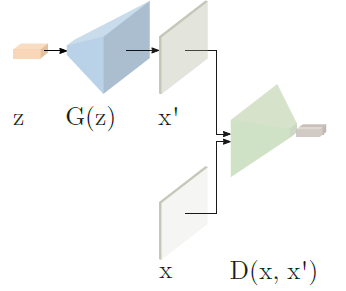


Fig. 4.3.5.1 ANOGAN Architecture(similar to that of GAN)

In this architecture, they have calculated the residual loss, which measures the dissimilarity between background images and the signal images in our case. Following the original idea of one of the researchers Yeh, giving input the generated image G(z) into the discriminator and calculating the sigmoid cross-entropy in the training phase takes into consideration the discriminator that the input sample is derived by the real data distribution. Alternatively, using the idea introduced by another researcher Salimans, and used by the AnoGAN authors, to compute the feature matching loss, extracting features from a discriminator layer f in order to take into account if the generated sample has similar features as that of the input one.

**Pros**

* The implementation shows GANs can be used for Anomaly Detection.
* Provided a new mapping methodology to map latent space into input data.
* Used the same mapping scheme to define an anomaly score.

**Cons**

* + This implementation requires time optimization steps for every new input and also having bad test-time performance.
  + The GAN objective has not been modified to take into account the need for inverse mapping learning.
  + The Anomaly Score is difficult to interpret, not being in the probability range.

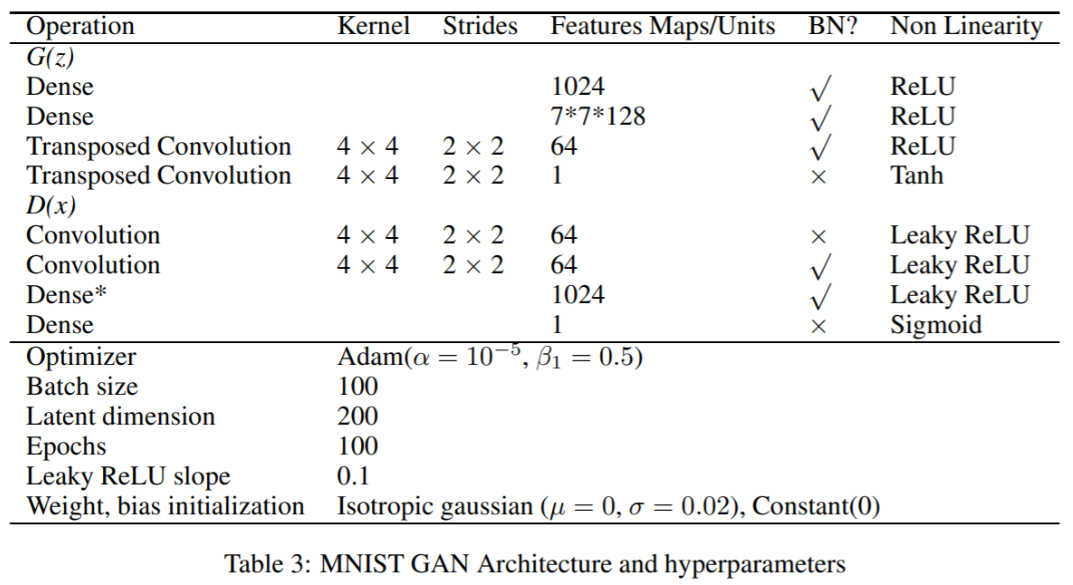


Fig. 4.3.5.2 Stepwise selection of hyper-parameters

**2] Efficient GAN Based Anomaly Detection(EGBAD) Approach:**

There are two main components of the Generative Adversarial Networks(GANs). One is the Generator, and the other is the Discriminator. These two models play an important role. The generator learns to create the realistic synthetic images from noise(i.e. latent space representation), and the discriminator learns to distinguish real images from the generated images. BiGAN architecture extends the GAN architecture by adding the third component: The Encoder Model. The function of the Encoder model is to learn to map the input data into the latent representation. The objective of the generator in BiGAN architecture remains the same while the objective of the discriminator is to classify between a real sample and a synthetic sample and additionally between a real encoding, i.e., given by the encoder, and a synthetic encoding, i.e. a sample from the latent space z. When the training phase is completed, the generator has learned the mapping from latent space representation (z) to realistic (normal) images (x). Now, there are two latent space representations: one is the original latent space z, and other is the latent representation from Encoder E given the input x as shown in the figure. Similarly, the Generator model will take input z and given the output G(z), which will also be similar to the x. The degree of similarity of x and G(z) depends on to which extent the normal image follows the data distribution that was used for training of the generator. To find the best z, we start with randomly sampling z1 from the latent space distribution Z and feed it into the trained generator to get a generated image G(z1). Based on the generated image G(z1), we define a loss function, which provides gradients for the update of the coefficients of z1, resulting in an updated position in the latent space, z2. In order to find the most similar image G(zΓ ), the location of z in the latent space Z is optimized in an iterative process via γ = 1, 2, . . . , Γ back propagation steps. In continuation to this, we define a loss function which maps new images to the latent space that comprises two components, a residual loss and a discrimination loss. The residual loss focuses on the visual similarity between the generated image G(zγ) and query image x. The discrimination loss focuses on the generated image G(zγ). Therefore, both components of the trained GAN, the discriminator D and the generator G, are utilized to adapt the coefficients of z via backpropagation. In the following, we give a detailed description of both components of the loss function.

**Residual Loss**- The residual loss measures the dissimilarity between normal image x and generated image G(zγ) in the image space representation. Under the assumption of a perfect generator G and a perfect mapping to latent space z, for an ideal normal case, images x and G(zγ) are identical. In this case, the residual loss is zero.

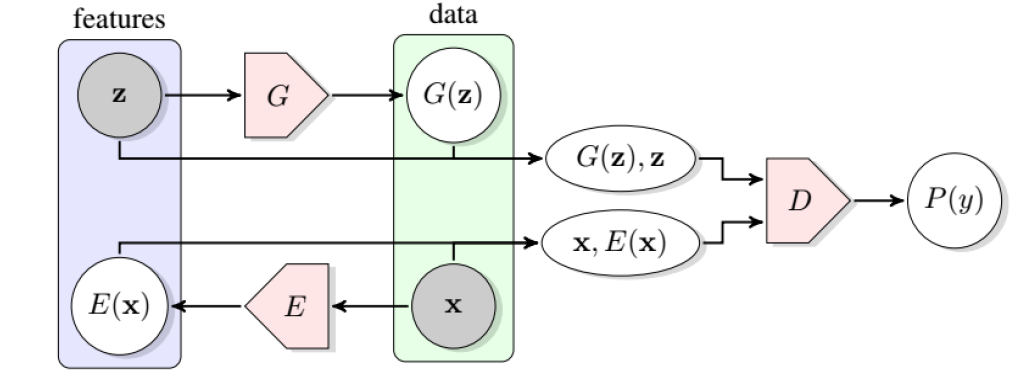


Fig. 4.3.5.3 Efficient GAN Based Anomaly Detection(EGBAD) Architecture operating on BiGAN network

**Discrimination Loss**- Firstly, we will feed the generated image G(zγ) into the discriminator D and also applying the sigmoid cross-entropy, which defined the discriminator loss of real images during training. In contrast to the work of the researcher Yeh, where zγ is updated to fool Discriminator D, we define an alternative discrimination loss LD(zγ), where zγ is updated to match G(zγ) with the learned distribution of normal images. This is inspired by the recently proposed feature matching technique. Feature matching addresses the instability of GANs due to over-training on the discriminator response. In the feature matching technique, the objective function for optimizing the generator is adapted to improve GAN training. Instead of optimizing the parameters of the generator via maximizing the discriminator’s output on generated examples, the generator is forced to generate data that has similar statistics as the training data, i.e. whose intermediate feature representation is similar to those of real images. Another researcher, Salimans, found that feature matching is especially helpful when classification is the target task. Since we do not use any labelled data during training, we do not aim for learning class-specific discriminative features, but we aim for learning good representations. Thus, we do not adapt the training objective of the generator during training the model, but instead, use the idea of feature matching to improve the mapping to the latent space. Instead of using the scalar output of the discriminator for computing the discrimination loss, we propose to use a richer intermediate feature representation of the discriminator where the output of an intermediate layer f(·) of the discriminator is used to specify the statistics of an input image. Based on this new loss term, the adaptation of z does not only rely on a hard decision of the trained discriminator, whether or not a generated image G(zγ) fits the learned distribution of normal images but instead takes the rich information of the feature representation, which is learned by the discriminator during adversarial training, into account. In this sense, our approach utilizes the trained discriminator not as a classifier but as a feature extractor.

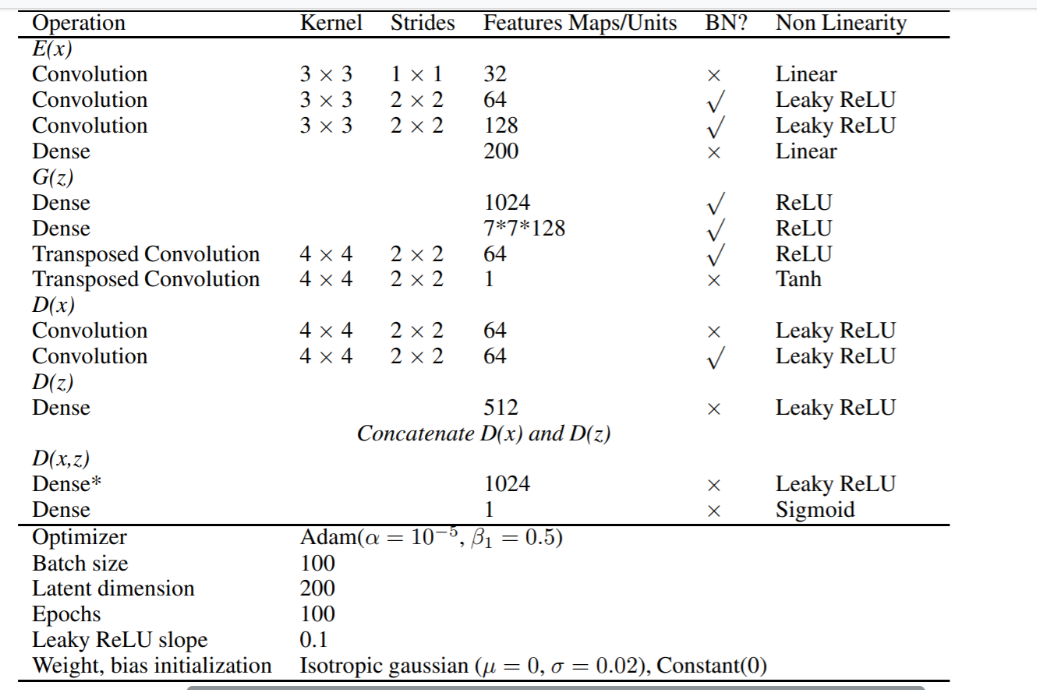


Fig. 4.3.5.4 Stepwise Hyper-parameter selection

**3] GANomaly Approach:**

Generator network The generator network consists of three elements in series, an encoder GE a decoder GD (both assembling an autoencoder structure) and another encoder E. The architecture of the two encoders is the same. GE takes in input an image x and outputs an encoded version z of it. Hence, z is the input of GD that outputs ^x, the reconstructed version of x. Finally, ^x is given as an input to the encoder E that produces ^z.

There are two main contributions to this architecture. First, the operating principle of the anomaly detection of this work lies in the autoencoder structure. Given that we learn to encode normal (non-anomalous) data (producing z) and given that we learn to generate normal data (^x) starting from the encoded representation z, when the input data x is an anomaly its reconstruction will be normal. Because the generator will always produce a non-anomalous image, the visual difference between the input x and the produced ^x will be high and in particular, will spatially highlight where the anomalies are located. Second, the encoder E at the end of the generator structure helps, during the training phase, to learn to encode the images in order to have the best possible representation of x that could lead to its reconstruction ^x.

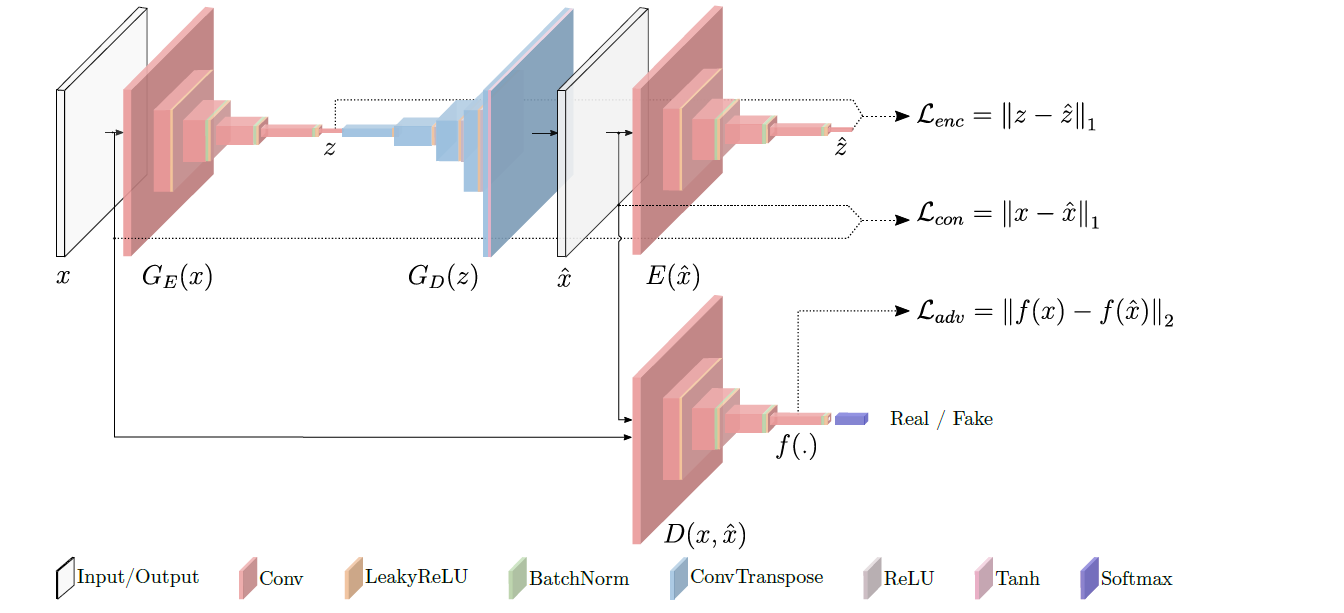


Fig. 4.3.5.5 GANomaly Architecture

The discriminator network D is the other part of the whole architecture, and it is, with the generator part, the other building block of the standard GAN architecture. The discriminator, in the standard adversarial training, is trained to discern between real and generated data. When it is not able to discern among them, it means that the generator produces realistic images.

Besides these two networks, the other main contribution of GANomaly is the introduction of the generator loss as the sum of three different losses; the discriminator loss is the classical discriminator GAN Architecture loss. The objective function is formulated by combining three loss functions, each of which optimizes a different part of the whole architecture.

**A] Adversarial Loss-** Following the current trend within the new anomaly detection approaches, we also use feature matching loss for adversarial learning. Proposed by one of the researchers Salimans, feature matching is shown to reduce the instability of GAN training. Formally, let f be a function that outputs an intermediate layer of the discriminator D for a given input x drawn from the input data distribution, feature matching computes the L2 distance between the feature representation of the original and the generated images, respectively.

**B] Contextual Loss-** The adversarial loss is adequate to fool the discriminator D with generated samples. However, with only an adversarial loss, the generator is not optimized towards learning contextual information about the input data. It calculates the loss by bifurcating between the original image samples and the generated image samples.

**C] Encoder Loss-** The two losses introduced above can enforce the generator to produce images that are not only realistic but also contextually sound. Moreover, we employ an additional encoder loss to minimize the distance between the bottleneck features of the input and the encoded features of the generated image. Basically, it calculates the difference between the Latent space representation from the first encoder and from the second encoder as well.

**4.4 PARAMETERS FOR TRAINING A NETWORK:**

**Hyperparameters** are the **variables which determine the network structure**(Eg: Number of Hidden Units) and the **variables which determine how the network is trained**(Eg: Learning Rate). **Hyperparameters** are **set before training**(before optimizing the weights and bias).

**Hyperparameters related to training Algorithm:**

**1] Learning Rate:** The learning rate defines how quickly a network updates its parameters. **Low learning rate** slows down the learning process but converges smoothly. **Larger learning rate** speeds up the learning but may not converge.

**2] Momentum:** Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations. A typical choice of momentum is between 0.5 to 0.9.

**3] Number of epochs:** Number of epochs is the number of times the whole training data is shown to the network while training. Increase the number of epochs until the validation accuracy starts decreasing even when training accuracy is increasing(overfitting).

**4] Batch size:** Mini batch size is the number of sub samples given to the network after which parameter update happens.

There are many Hyperparameters other than this that has some specific value and the values are shown in the table below:

|  |  |
| --- | --- |
| **HYPERPARAMETERS** | **VALUE OF HYPERPARAMETER** |
| **Learning rate** | 0.0001 |
| **Momentum** | 0.5 |
| **Decay** | 1e-6 |
| **Optimizer** | Adam |
| **Loss function** | binary\_crossentropy |
| **Batch size** | 128 or 256 (for large data) |
| **Number of Epoch** | 70 |

Table 4.4.1 Values of Hyperparameters in implementation code.

There are three types of Optimization Algorithms:

1] Grid Search Algorithm

2] Random Search Algorithm

3] Bayesian Optimization Algorithm

**4.5 RESULTS AFTER THE IMPLEMENTATION:**

We have implemented the code for the GANomaly architecture on Google Colab as well as on the Cluster also. The implementation of the Cluster makes it faster. Basically, we are implementing it by using background and signal dataset. Training of the model is done with the help of a background dataset so that the model can learn to produce similar images like background when it is tested. Now, the test dataset consists of background samples as well as signal samples, signal samples being the anomaly. So, during the testing phase, when the background images will enter the network, it will process like the background set, and by comparing the output and input, we will get the minimal difference. However, when the signal images enter the network as input, they will be processed like background images because the model is trained with background images. So, while comparing the output with that of input, we get the maximal difference between the images. That's how the anomalous image in our dataset is recognized.

By setting a suitable environment, the implementation of GANomaly architecture had done, and lastly, the output had observed. The output consists of Plotting the ROC(Receiver Operating Characteristic) and also observing the Losses of generator and discriminator.

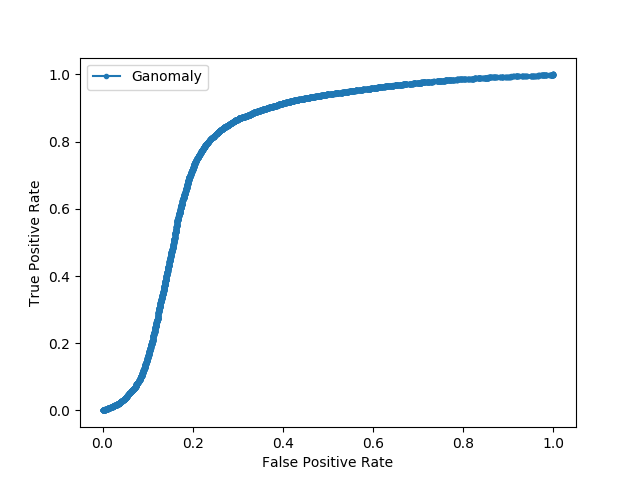


Fig. 4.5.1 ROC curve Plot

This is the example of the receiver operating characteristic (ROC) curve, which is showing the AUC (Area Under the Curve) of 0.8014, which is nearly 80.14% accurate.

**Scattered plot:**

The following Scatter plot suggests the signal (pink) and Background (blue) images. The background images, when tested, give the output similar to the input itself, so the Anomaly score turns out to be less as compared to that of Signal(pink) images. The signal(pink) images when inserted to this trained model, it will give an output which looks similar to the background images, and when the difference is calculated between the images, it will be large. So, the Anomaly score is large as compared to that of background images.

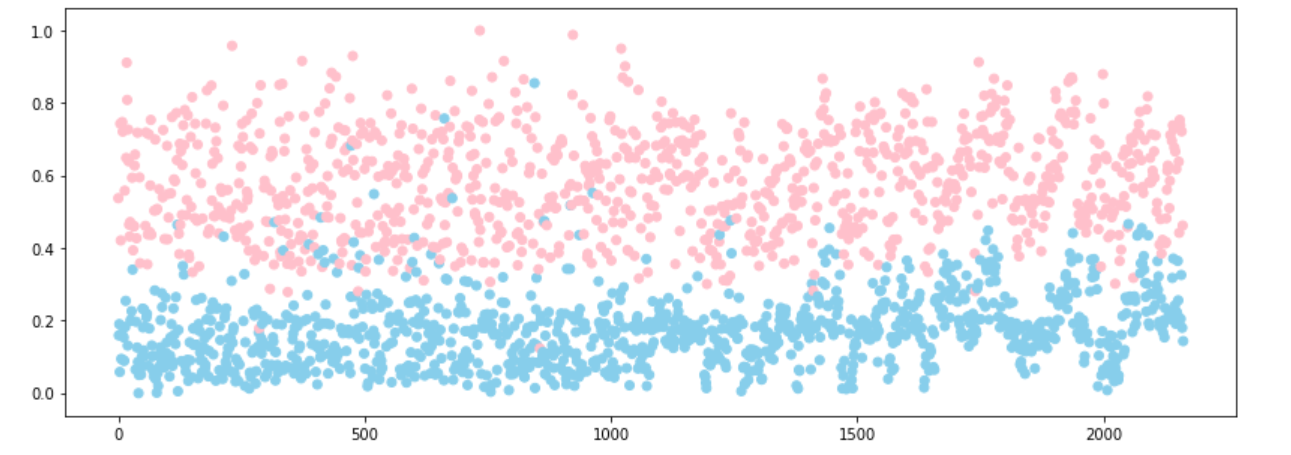


Fig. 4.5.3 Plot of Anomaly score between signal and background images.

**4.5.1 Comparison of the obtained results:**

The research papers which are referred to implement the GANomaly architecture had also implemented this architecture for Anomaly Detection. But all these research papers have used MNIST dataset and CIFAR10 dataset for anomaly detection. So, here we are going to compare the ROC curve plot with ours.

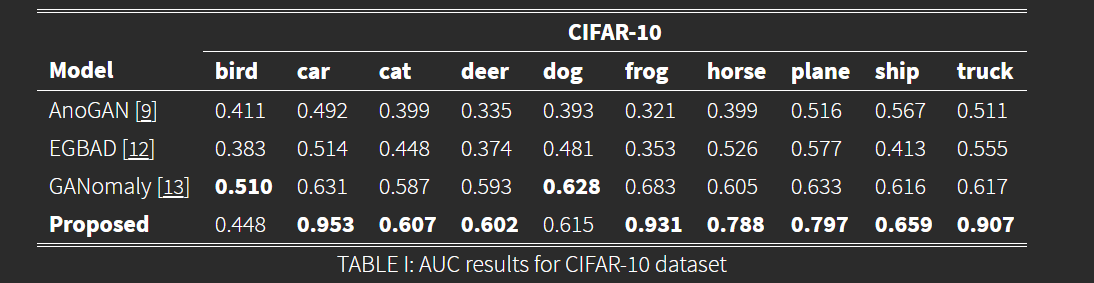


Fig. 4.5.1.1 ROC for all three architectures on CIFAR 10 dataset.

The results of the implementation of the architecture suggest that GANomaly architecture shows more AUC as compared to the other two approaches. These three approaches have been implemented with the CIFAR-10 dataset. For all the classes of the dataset GANomaly approach shows the best results. Moreover, Anomaly detection of jet images has also been implemented using the CNN technique. Below is the ROC curve plotting using the CNN model.

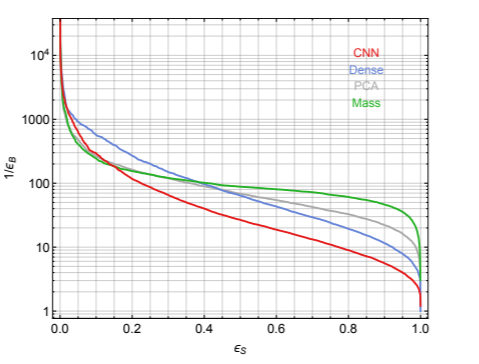


Fig. 4.5.1.1 ROC curve plotting using CNN architecture.

|  |  |  |
| --- | --- | --- |
| **RESULTS** | **CNN ARCHITECTURE** | **GAN ARCHITECTURE** |
| **AUC value** | 0.77 | 0.8014 |
| **Number of layers** | 3 | 4 |
| **Number of input filters** | 128 | 32 |
| **Activation Function** | ReLU | LeakyReLU |
| **Loss Function** | Mean squared error | binary\_crossentropy |
| **Image input shape** | 40 x 40 x 1 | 32 x 32 x 3 |
| **Accuracy of the model** | 0.90 | 0.88 |

Table 4.5.1.1 Comparison of GAN and CNN models

As shown from the table, CNN architecture shows the accuracy of 90% with the dataset of background and signal images. They have implemented the whole code in Keras and in CNN, they used Conv2d, MaxPooling2D, Flatten, Dense layers in their architecture. The dimensionality reduction is shown by the usual max-pooling layers. After a series of max-pooling and Convolutional layers, the output is fed to one or two dense layers, resulting in the latent space representation. The whole process is inverted (with 2D upsampling layers instead of the max-pooling layers) to retrieve an image back with the same dimensions. Since the data is large, they kept a large batch size that takes more image samples to process at a time. Since they have used some significant layers, the noise in the images was not present. They showed that after using the mean square error loss function, the loss was determined minimally in the space of linear projections. Thus they have also used PCA (Principal Component Analysis) for dimensionality reduction and compared the PCA ROC curve with that of jet mass ROC curve. Moreover, their architectural implementation was completely fine.

Source: <https://arxiv.org/pdf/1808.08992.pdf>

**4.6 PROBLEMS FACED DURING IMPLEMENTATION:**

While compiling the model, faced an issue of image size conversion. So, to solve this error, reshape() function was used, and the image was reshaped into an appropriate shape.

**Error in number of filters:** According to the size of the image, number of filters are adjusted. For example, if the image size is a multiple of 3, then we can set a filter size as a multiple of 3.

**Scaling Error:** In order to bring features to the same scale, we could decide to use either normalization or standardization of features. If the feature tends to be uniformly distributed, then we may use normalization (MinMaxScaler).

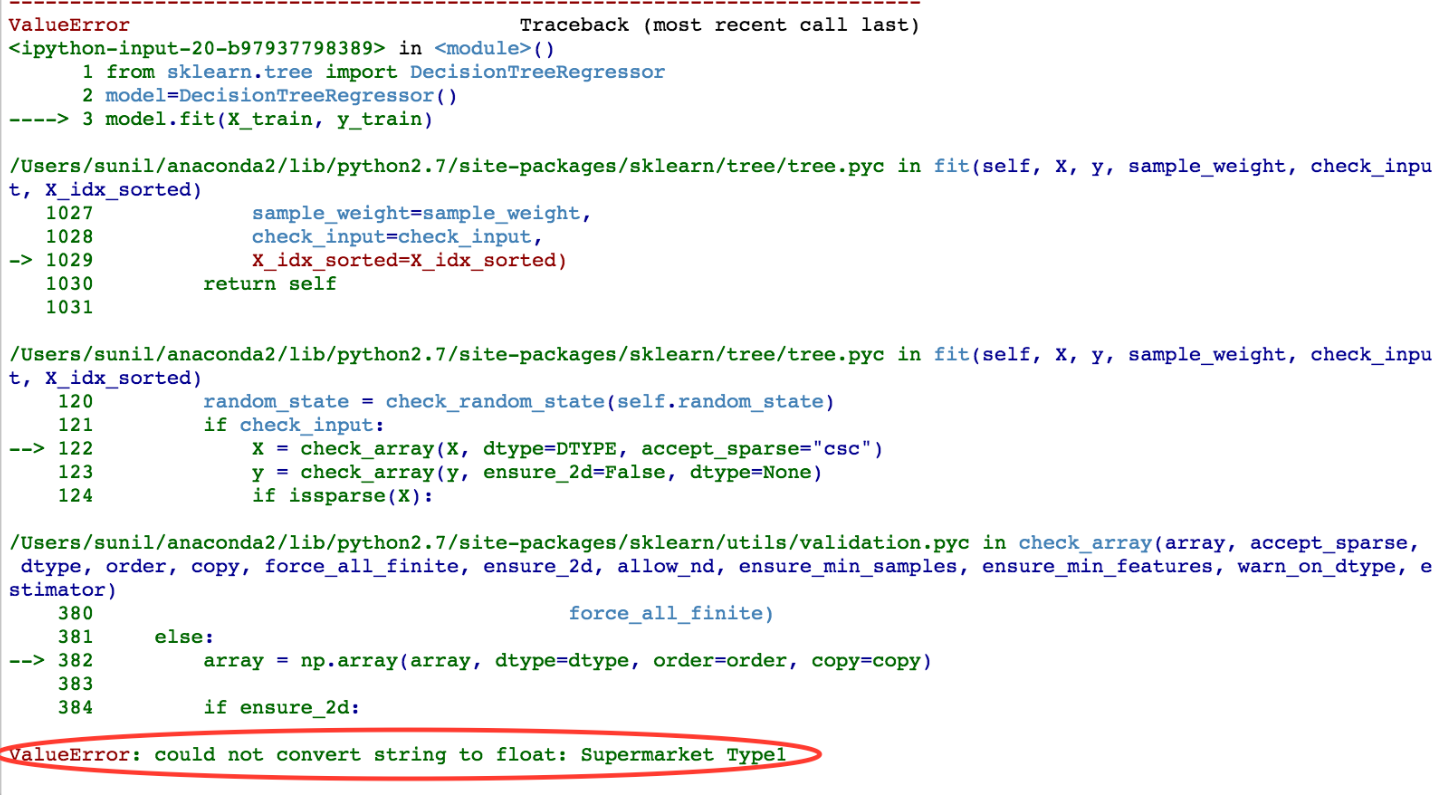


Fig. 4.6.1 Scaling error.

* **Variance Error:** This occurs when too many features are used in training the model so that the model captures both real and random effects. Generally, a model trained on a very high dimensional dataset is too complex and difficult to interpret. It is always good to find the right balance between Bias Error (under fitted) and Variance Error (overfitted).

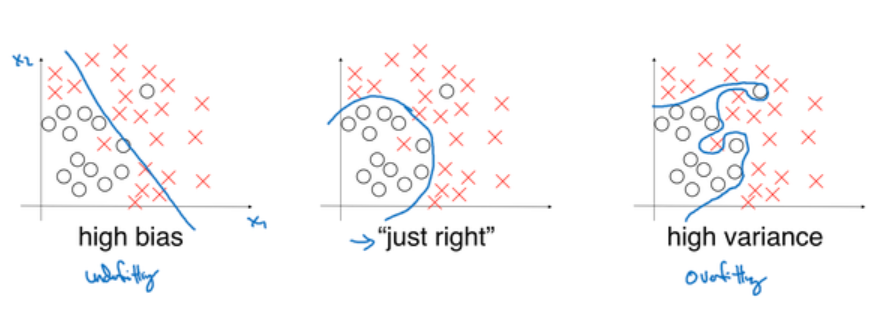


Fig. 4.6.2 Overfitting and Underfitting Error.

* **Value Error:** Could not broadcast input array of shape (256, 1) to (256). Use the Reshape function to change the shape of the array.

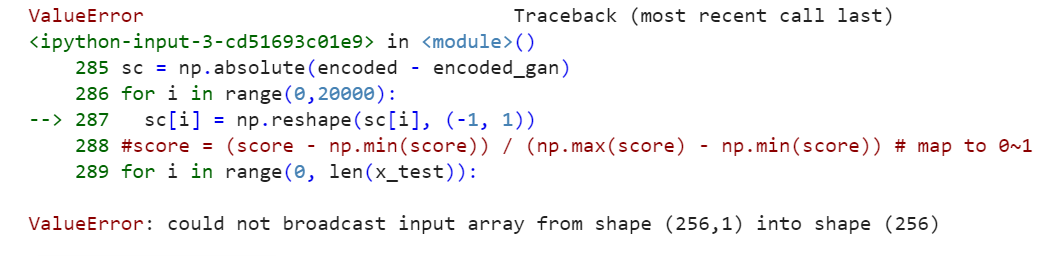


Fig. 4.6.3 Value Error.

**5: PROPOSED WORK**

**5.1 LOADING THE DATASET**

For the image data set, we used the pickle format where lots of data can be stored in the form of the NumPy array. The file was in the Dictionary form, and it was a need to extract that data for passing these image data to a particular Network. So, we have implemented the code to load the pickle dataset. There are two ways to load a pickle Dataset: one, by converting the NumPy array into images and then loading the image data to a neural network, and by, directly loading the NumPy array to a Network. The second method works faster than the previous one because it takes less time for pre-processing the data. Pre-processing data includes storing images in NumPy array form into some variable as well as normalizing the pixel data.

**5.2 IMPLEMENTING THE ARCHITECTURE**

Our dataset consists of two classes, namely the background and signal. Background images are the images formed by the quark/gluon particles, while signal images are the images formed by top-quark particles. Further, these top-quark particles decompose into w-bozon and bottom-quark. Following this, w-bozon particles decompose into (u and d) quarks. These particles constitute to form actual signal images. We have approximately 1,08,887 samples of background images and around 90,000 samples of signal images. We were training on background images to find the anomaly(signal) images. To do so, we used about 80,000 in training data and 10,000 samples into testing data. Since we are detecting the anomaly(signal) images, only signal images are considered in test data. We use 10,000 samples of signal images in test data. So, in total, we now have 20,000 images of signal and background images. We further trained the following images on the corresponding Architecture, and once the network is trained, We have executed it for the test dataset. Since we have trained the network for background images, while testing images, it will generate images similar to the images in the training dataset. So, if there was an anomalous image in the input, then it should be detected. Based on this logic, the anomaly detection using GAN is done.

**5.2.1 Python Libraries Used:**

**1] Keras:** It is an open-source library used in many neural-networks. Keras is written in Python. It was designed with an aim to enable fast processing with deep neural networks. It is also user-friendly, modular and extensible.

**2] sklearn:** It is a free and open-source machine learning library. It is used in the Python programming language. It includes various classification, regression and clustering algorithms with an added algorithm of support vector machine.

**3] cv2:** It is a Pythonlibrary that is designed to solve computer vision problems.

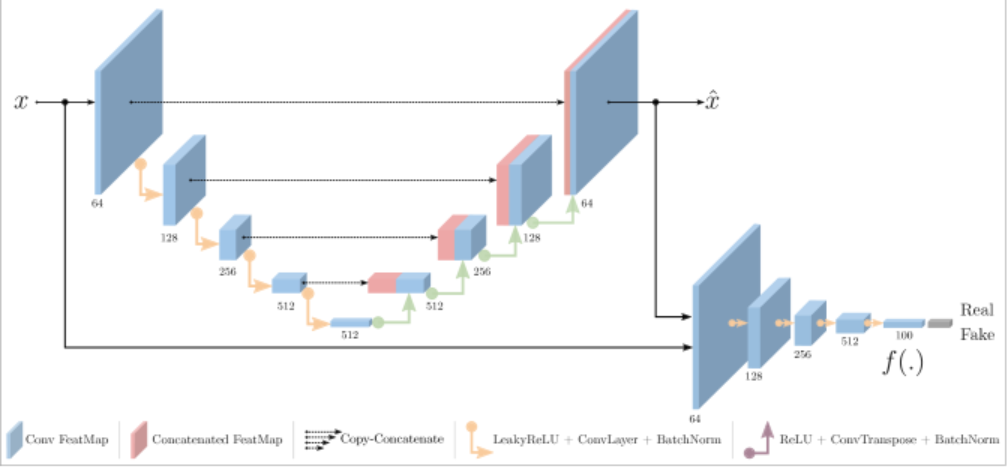
**4] Numpy: Numpy** is an array-processing package used in a general sense. It involves the processing of multi-dimensional array objects that are high-performance. It also provides tools for processing these arrays. It is the most basic and important package for scientific computing in **Python**.

**5] Pickle:** It is the module that is used for serializing and de-serializing a **Python** object and its structure. **Pickling** is a method for converting a **python** object (list, dict, etc.) into a character stream data.

**6] Matplotlib: Matplotlib** is a plotting library. It is used in the Python programming language. It has also benefited in numerical mathematics extension Numpy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

**5.2.2 Logic used for implementation:**

As we all know by now that GANomaly Architecture works well as compared to all other architectures, so the implementation of GANomaly architecture is carried out forward. Starting with the libraries, after importing all the libraries Construction of each and every model in this neural network had been done. In this GANomaly architecture, there are a total of three models, namely: generator(Autoencoder), Encoder and Discriminator.

Fig. 5.2.2.1 Logic implementation.

Each of these models contains 4 layers, and these layers have to be Convolutional layers for more image processing.

**Building a Model:** 3 Models generator(which is also called Autoencoder), Simple Encoder and Discriminator is constructed as shown in the above figure. Each of these models has different layers according to their functionality, and an additional model called the feature extractor model is also made which particularly extracts the features from the images and passes that features into the discriminator model. This feature extractor layer is known by the symbol "f" in the figure.

**Compilation of the generator model:** After formulating each and every model, we have to compile the model using the compile() function which basically takes three main parameters namely: loss function, Optimizer and metrics(for accuracy display).The compilation is an efficiency step. It transforms the simple sequence of layers that we defined into a highly efficient series of matrix transforms in a format intended to be executed on your GPU or CPU, depending on how Keras is configured.

**Preparing the Dataset:** The Dataset of jet images consist of a training set and the testing set. The training set consists of 90,000 images of background samples. The test set consists of 10,000 images of background samples and 10,000 images of signal samples. In this case, the signal is the anomalous data, and the background images are the real image data. The Dataset was prepared on a simulated tool used by the members of the Physical Research Laboratory(PRL).

**Data Preprocessing:** The preprocessing step includes the normalization of the data because the pixel data is a big number. So we have to normalize it into 0-1 or 0-255. This can be done by Min-Max Normalization(0-1) and Z-score Normalization(0-255). We can use either of the techniques for normalization depending upon the data given to us.

**Model Training:** Training is the most important step in our case because all the testing outcomes are dependent on the training. The generator and the discriminator have different training processes. GAN training proceeds in alternating periods:

1. The discriminator trains for one or more epochs.
2. The generator trains for one or more epochs.

We keep the generator constant during the discriminator training phase. As discriminator training tries to figure out how to distinguish real data from fake, it has to learn how to recognize the generator's flaws. That's a different problem for a thoroughly trained generator than it is for an untrained generator that produces random output.

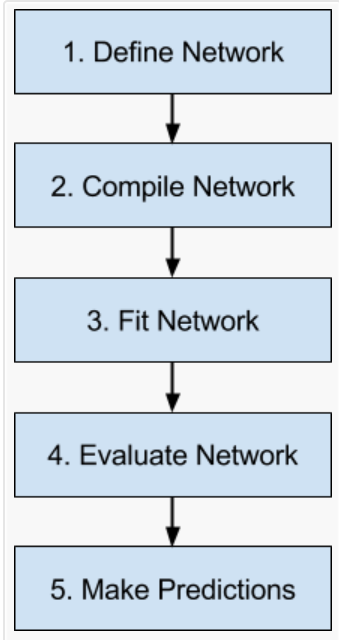


Fig. 5.2.2.1 Neural Network Flowchart.

**Evaluate Network:** The network can be evaluated on the training data, but this will not provide a useful indication of the performance of the network as a predictive model, as it has seen all of this data before. We can evaluate the performance of the network on a separate dataset, unseen during testing. The model evaluates the loss across all of the test patterns, as well as any other metrics specified when the model was compiled, like classification accuracy.

**Make Predictions:** Finally, once we are satisfied with the performance of our fit model, we can use it to make predictions on new data. The predictions will be returned in the format provided by the output layer of the network. In the case of a regression problem, these predictions may be in the format of the problem directly, provided by a linear activation function. For a binary classification problem, the predictions may be an array of probabilities for the first class that can be converted to a 1 or 0 by rounding.

After the implementation of the neural networks, we used to plot the ROC curve and calculate the AUC(Area Under the Curve) for each and every network. For ANOGAN Architecture, the AUC value was coming to be 0.69, whereas for the EGBAD approach it was to be 0.72. Finally, the GANomaly architecture was showing me the highest AUC value, and that was to be 0.75 approximately. Hence, among these three architectures, the GANomaly approach was giving me the best performance. We can also plot the Precision-recall curve, but here the test dataset consists of the equal number of images for background and signal, we plot the ROC curve. In the case of unequal data, we always have to plot the precision-recall curve.

**6: CONCLUSION**

Observing and analyzing the Figures of the ROC curve, we can get different results for an AUC plot. Those results were giving the AUC value. There were three values that were noted, and the values showed an increase in AUC. These values were 16.84% initially, further increased by 68.71% and finally, they indicated a surge in the value to 80.14%, i.e. our model is able to classify 80% of the images between signal and background dataset. All of the implementations were done in python. Additionally, the analysis and implementation of the algorithms allowed us to verify the effectiveness of the GANs-based algorithms for anomaly detection problems. This GAN-based approach indicates the differences between the MNIST dataset and our jet image dataset. Besides this, to test the feasibility of the architectures mentioned above, this work intends to provide some significant solution to the Anomaly Detection of images. We demonstrated that recent GAN models could be used to achieve state-of-the-art performance for anomaly detection on high-dimensional, complex datasets whilst being efficient at test time.

After the implementation of the GANomaly approach, we introduce a novel encoder-decoder-encoder architectural model for general anomaly detection enabled by this training framework. Experimentation has been done for various datasets of different level of complexity, and the implementation of GAN-based architectures has shown us the approaches of generalization ability to any anomaly detection task. Overall, it can be commented that out of these three architectures GANomaly approach was the best suited for anomaly detection using the jet particle images. Furthermore, it can also be stated that for the generation of images, Generative Adversarial Networks(GAN) can be used efficiently and effectively.

**6.1 REASONS FOR NOT GETTING THE BEST RESULTS:**

The data that was given to me was almost similar. By naked eye, we cannot classify the images of signal and background. Though the model was required to classify, due to its complex structure, the model was not giving us very close to 100% accuracy.

I was using the Generative Adversarial Models(GANs). So, there are exactly two models inside GAN to implement. One is the generator model, and the other is the discriminator model. We have to train as well as test both these models. Therefore, complexity may be one of the reasons for not getting the best results.

Techniques that are implemented to evaluate the GAN architecture were not working properly. For Plotting ROC curve, the general method is to predict the probability using predict\_proba() function, but in Functional Model API, this function does not work. Hence, we have to use some other function to predict probability which is not giving accurate results.

The values of the hyperparameters, such as learning rate, batch size, etc. are not the perfect ones. Hyperparameter tuning is required for getting the perfect values of all these hyperparameters. This can be done by various software that are open source in the market (e.g. AUPTIMIZER).

**Acknowledgements**

The data used here are courtesy to Madgraph and Pythia simulations for the particle level realistic simulation of Large Hadron Collider data along with detector effects. This work is supported by the Physical Research Laboratory (PRL), Department of Space, Government of India. Computations were performed using the HPC resources (Vikram-100 HPC) and TDP project at PRL.

**REFERENCES**

* CMS Collaboration. Simulated dataset dyjetstoll tunez2 m-50 7tev-madgraph-tauola in aodsim format for 2011 collision data (sm inclusive), 2016. DOI: 10.7483/opendata.cms.txt4.4rrp.
* CMS Collaboration. Simulated dataset wjetstolnu tunez2 7tev-madgraph-tauola in aodsim format for 2011 collision data (sm inclusive), 2016.DOI: 10.7483/opendata.cms.u7p6.ckvb.
* CMS Collaboration. Simulated dataset ttjets tunez2 7tev-madgraph-tauola in aodsim format for 2011 collision data (sm inclusive), 2016.DOI: 10.7483/opendata.cms.zbgf.h543.
* 2017 Introducing DNN primitives in Intel Math Kernel Library https://software.intel.com/en-us/articles/introducing-dnn-primitives-in-intelr-mkl
* 2017 Intel distribution of Ca↵e\* https://github.com/intel/caffe
* 2017 Intel Machine Learning Scaling Library for Linux OS https://github.com/01org/MLSL
* Kurth T et al. 2017 Accepted for SC17, arXiv:1708.05256 URL <http://arxiv.org/abs/1708.05256>
* <https://www.mathworks.com/discovery/convolutional-neural-network.html>
* <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>
* <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>
* <https://blog.datawow.io/interns-explain-cnn-8a669d053f8b>.
* <https://arxiv.org/abs/1805.06725>
* <https://arxiv.org/abs/1605.09782>
* <https://arxiv.org/abs/1605.09782>
* <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>
* <https://arxiv.org/abs/1611.07004>
* <http://www.cs.toronto.edu/~kriz/cifar.html>
* <http://yann.lecun.com/exdb/mnist/>
* <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
* <https://arxiv.org/abs/1411.1784>
* <https://dblp.uni-trier.de/db/journals/corr/corr1708.html#abs-1708-07747>
* <https://arxiv.org/abs/1607.07539>
* <https://arxiv.org/abs/1802.06222>
* <https://openreview.net/forum?id=SJQHjzZ0->
* <https://openreview.net/forum?id=HJ1HFlZAb>
* <https://openreview.net/forum?id=Sy1f0e-R->
* <https://arxiv.org/ps/1802.06222v2>
* <https://arxiv.org/ps/1802.06222v2>
* <https://arxiv.org/ps/1802.06222v2>
* https://arxiv.org/ps/1802.06222v2

**EXPERIENCE**

* Familiarity with machine learning is one of the greatest achievements in my life. I am working with different models such as CNNs, GANs, ANNs, etc. The training dataset and testing dataset makes these networks much efficient and faster for execution.
* We have implemented it using the Keras library of python. Keras makes it all simple and easily understandable to anyone. Implementation in Keras is easier than in PyTorch, Theano, etc. Among all the other python libraries, Keras is the simplest to implement.
* Working in this project helped me to gain more knowledge about Neural Networks and its applications. I also got to know about dense layer, flatten layer, Batch Normalization, activation function, Sequential model in Keras as well as functional API model, optimizer, learning rate, a compilation of the model, batch gradient descent, stochastic gradient descent, momentum, sample interval, regularization, dropout, etc.
* It has also helped me to gain knowledge about various types of GANs and its architectures. Implementation is also a crucial part of the project. All the codes are implemented in Keras.
* Working at PRL has also been fruitful to me. It has developed a work culture inside me that might help me in future purposes. Working Environment has also been good.

**RESEARCH PUBLICATION**

* Akcay, S., Abarghouei, A. A., and Breckon, T. P. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. abs/1805.06725, 2018. URL <http://arxiv.org/abs/1805.06725>.
* Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. abs/1708.07747, 2017. URL http://dblp.uni-trier.de/db/journals/ corr/corr1708.html#abs-1708-07747.
* Yeh, R. A., Chen, C., Lim, T.-Y., Hasegawa-Johnson, M., and Do, M. N. Semantic Image Inpainting with Perceptual and Contextual Losses. abs/1607.07539, 2016. URL http://arxiv.org/abs/1607.07539.
* Zenati, H., Foo, C. S., Lecouat, B., Manek, G., and Chandrasekhar, V. R. Efficient GAN-Based Anomaly Detection. abs/1802.06222, 2018. URL http://arxiv. org/abs/1802.06222.
* D. J. Im, A. H. Ma, G. W. Taylor, K. Branson, Quantitatively evaluating gans with divergences proposed for training, International Conference on Learning RepresentationsAccepted as poster. URL <https://openreview.net/forum?id=SJQHjzZ0->
* T. Lesort, F. Bordes, J.-F. Goudou, D. Filliat, Evaluation of generative networks through their data augmentation capacity (2018). URL <https://openreview.net/forum?id=HJ1HFlZAb>
* Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, Y. Yu, Activation maximization generative adversarial nets, International Conference on Learning Representations. URL <https://openreview.net/forum?id=HyyP33gAZ>
* G. Huang, Y. Yuan, Q. Xu, C. Guo, Y. Sun, F. Wu, K. Weinberger, An empirical study on evaluation metrics of generative adversarial networks, International Conference on Learning RepresentationsRejected. URL <https://openreview.net/forum?id=Sy1f0e-R->
* Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015
* Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. arXiv preprint arXiv:1602.05110, 2016

**CURRICULUM VITAE**

**NAME:** PATEL PARTH NITINBHAI

**ADDRESS:** 94 Narnarayan Bunglows Behind Mipco Street 6 Bharuch 392015 Gujarat.

**MOBILE:** +91-74869-05745

**EMAIL:** parthp0728@gmail.com

**CAREER OBJECTIVE**

To pursue a highly rewarding career, seeking for a job in a challenging and healthy work environment where I can utilize my skills and knowledge for organizational growth and can give a remarkable contribution to the information technology World. I want to explore my knowledge in this field of interest.

**ACADEMIC PROFILE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **Institute** | **Year of Passing** | **CPI(Percentage)** |
| **B.Tech. (Information technology)** | Dharmsinh Desai Institute Of Technology, Nadiad. | Pursuing (2016-2020) | 8.33 |
| **(H.S.C.)** | Amity High School, Bharuch. | March-2016 | 81.67% |
| **(S.S.C.)** | Amity High School, Bharuch. | March-2012 | 83.67% |

**KEY STRENGTH**

* Ability to work with full efforts and dedication in any condition.
* Helpful, co-operative, self-motivated

**HOBBIES AND INTERESTS**

* Playing cricket
* Watching movies
* Listening to music, Traveling

**EXTRACURRICULAR ACTIVITIES**

* Attended Hack x-void(Cyber Security& Ethical Hacking)Seminar.
* Successfully completed digital marketing training on internshala.
* Successfully completed advanced Excel training on internshala.
* Successfully completed Business communication skills training on internshala
* Attempted NCAT examination successfully.
* Completed deep learning course on courser.com

**TECHNICAL SKILLS**

* Basics of C,C++,Java,Python
* HTML,CSS
* ASP.NET
* SQL,DBMS
* data structures andAlgorithms
* Basics of AJAX, JQuery

**PROJECTS**

July 2018 – Oct 2018

**Gym Management System**

Technologies used: MYSQl, Xampp server, php, html, css

The aim of our project is to develop a system that is meant to provide easy maintaining and to handle gym facilities such as BMI calculation, database Updation and Deletion, Renew entries, subscription of member, functional checks for invalid entries and automatic primary key generation

21-22 feb.,2019

**Web Application for mall management system**

Technologies used: Android studio, Firebase database

The main objective of this project is to develop a system which includes front end and back end part of billing in malls. Also, languages like PHP and javascript for development and for backend development database management systems are used.

July 2019 - December 2019

**On-demand Music System**

Technologies used: Python - sci-kit learn library, Pandas, SVM classifier

The main objective of this project is to develop a system which includes features of Music which nowadays all apps are not providing to us. There will be Client and Admin apps which help the user to vote his/her music, and according to the votes, the corresponding franchise owner will play the song.

**WORKING EXPERIENCE**

PHYSICAL RESEARCH LABORATORY, AHMEDABAD, GUJARAT. (Jan 2020 – April 2020)

High Energy Physics(HEP) data analysis on jet images using GAN

Technologies used: Python - Keras library, Pandas, deep-learning, Autoencoder

This project involves Anomaly detection techniques on a particular dataset.

**LANGUAGE KNOWN**

* English
* Hindi

Gujarati

* I hereby declare that the above information is correct to the best of my knowledge and belief.

Yours Faithfully,

PATEL PARTH NITINBHAI