Classification of High Energy Tracks Using CNNs

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Vipul Pawar

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

April, 2020

Supervisor: **Prof. Sourabh Dube**

Certificate

This is to certify that this dissertation entitled **Classification of High Energy Tracks Using CNNs** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by **Vipul Pawar** at Indian Institute of Science Education and Research under the supervision of Prof. Sourabh Dube, Prof. Arun Thalapillil , Department of Physics , during the academic year 2019-2020.

Prof. Sourabh Dube

Committee:

Prof. Sourabh Dube

Prof. Arun Thalapillil

This thesis is dedicated to Rekha, Dinesh and Abhishek

Declaration

I hereby declare that the matter embodied in the report entitled **Classification of High Energy Tracks Using CNNs**, are the results of the work carried out by **Vipul Pawar** at the Department of Physics, Indian Institute of Science Education and Research, Pune, under the supervision of **Prof. Sourabh Dube**, and the same has not been submitted elsewhere for any other degree.

Vipul Pawar

Acknowledgments

I would like to start by expressing my sincere gratitude to Prof. Sourabh Dube. I have learned many things after coming under his umbrella. I would like to thank him for encouraging me to think out of the box and helping me to be a better version of myself. Most importantly, he believed in me when nobody did. I am lucky to be a part of his group.

I would like to thank Anshul, Angira, And Arnab. First, I thought the fifth year is going to be 24/7 working in one place, but you people showed me that it is not at all like that. When you love your work, there is a driving force that makes you do it, and you enjoy your life too.

I would like to thank Sneha. We had each other's back all the time in IISER. You taught me many things and always motivated me to follow my heart. I still wish that we would have ended things differently.

Next, I would like to thank my friends. They have made this journey memorable. Thank Komal, Dimple, Minal, for supporting me and helping me in emotional times, always. Thank you Ritika, you are an angel. I would like to thank my love, Deepesh, for pulling me out of dark times. Life is more beautiful and exciting when you are with me.

Finally, I must express my profound gratitude towards my parents, Rekha and Dinesh, and my brother, Abhishek, for giving me continuous support and believing in me throughout my life. This accomplishment wouldn't have been possible without them. Thank you.

Abstract

Image recognition is a topic that focuses to find and identify various specified objects, classes of object, features in given input image or video frame, even it's no so clear to see. Deep Convolutional Neural Networks (CNN) is the-state-of-the-art technique for image recognition. These convolutional neural networks may require long training time. This training time is depended on number of classes, number of training images in those classes, computing power, and complexity of the neural network.

This master's thesis studies an application of convolutional neural network in tracking of high energy particles which were produced at CMS detector at CERN.

There are many applications for image recognition. These applications spans various domains, such as Face recognition, OCR, Manufacturing inspection and Quality Control, Medical diagnosis, and Autonomous vehicle. The application discussed in the thesis is to identifying particles which has "high" energy that the "low" energy particles using images of the hit pattern formed in tracker of CMS detector.

The importance of this application is that this image identification process is faster once the model is trained. In future, LHC will go into its phase 3, because of this there will huge change in data which will get generated per collision. Traditional methods will failed to read and keep data in time as the hardware's writing speed is limited. This thesis try to show the use of convolutional neural network at a trigger level by using images data of tracks of particles per event.

Two models have been created for analysis. One model is called toy model and this is simplified model. Second model is based on data of simulated hits taken from CMS tracker. Performance of both models have been tested by putting different conditions on CNN architecture and tracker geometry.

Contents

Chapter 1

Introduction

1.1 Machine Learning in HEP

There are two main objectives for experimental high energy physics (HEP). The first one is probing the Standard Model(SM) with high, increasing precision and secondly the search for the new physics. These processes demand recognition of rare signals in a vast amount of background. As Large Hadron Collider (LHC) will get upgraded to high luminosity LHC (HL-LHC), the data generated in HL-HLC will be 100 times the luminosity of current LHC. This huge amount of data will also come with new challenges such as complexity of data, size of the data, etc. The performance of existing algorithms and computational power will limit the scope of physics of the experiment. The fusion of machine learning (ML) with experimental high energy physics promises to tackle these difficulties with great force.

Machine learning methods are designed to work with huge data-sets. These methods reduce the complexity of data and look for new features. Boosted Decision Trees (BDTs) and Neural Networks (NN) are mostly used algorithms. Machine learning models are mostly classification models and regression models. Relevant variables to the physics problem are selected, and models are trained by making classes as signal and background. Most time and human power dominating step is training, whereas a testing phase is comparatively fast. For the classification of particles, BDT's and NNs are used. Its been a while that neural networks are playing a role in HEP. After the revolution of deep learning, training algorithms and computational power have improved drastically. This has a serious influence

on HEP. If a huge amount of data is present with sophisticated features along with non-linear dependencies linking input and output, deep learning methods are favorable than traditional methods. Fully-connected networks (FCN), Convolutional neural networks (CNN), k-nearest neighbors (KNN) These are the typical examples of deep learning methods. This thesis focus on the application of convolutional neural networks. There are some related works that has published whcih are as follows:

- Patrick T. Komiske, Eric M. Metodiev, Matthew D. Schwartz
  Deep learning in color: towards automated quark/gluon jet discrimination. [12]
- Baranov, Dmitriy, Sergey, Pavel (2018).
  The particle track reconstruction based on deep learning neural networks. [13]
- The HEP.TrkX Project: Deep Learning for Particle Tracking. [14]
- Ashutosh V. Kotwal : A fast method for particle tracking and triggering using small-radius silicon detectors. [15]

1.2 Aim of the Project

At CMS, groups of protons collide 40 million times per seconds. There is only 25 nanoseconds time gap between each bunch crossing. So detector has only 25 nanoseconds to store the information of an event.

To do this job, there is a system in place called trigger system. This system rapidly make a choice of which events to keep. This choice making is necessary because only small fraction of event data gets stored. Due to this limitations of data storage rates and capacity, trigger system is very crucial. Current CMS has 100 kHz output rate of L1 trigger whereas pp collision rate exceeds 1 GHz.

In LHC upgrade luminosity will go up drastically and these trigger systems may not be able to cope up with huge amount of input data. This is the part where machine learning methods plays important role. More information about trigger system can be found here [2]

This thesis work presents applications of convolutional neural network (CNN) in tracking of particles. An interesting challenge is to classify high energy tracks from low energy tracks.

High pt tracks are seldom created in nature that is why they make interesting point of study. Traditionally, to achieve this, full track reconstruction is done and momentum of tracks are estimated. This thesis study attempts an image based solution to address this challenge with the help of convolutional neural networks (CNNs). CNN transforms images into tensors to learn about their features. Machine learning methods have been proven to be fast and reliable, this application of CNN for tracking may also used in trigger system to tackle the problem of high influx of data.

1.3 Thesis Structure

This thesis report has 6 chapters. Chapter[2] talks about large hadron collider and CMS detector. Chapter[3] reviews machine learning methods. Stars with artificial neural networks (ANN) , Convolutional neural networks (CNN) and then goes to working of CNN. Chapter[4] explains toy tracker model along with results. In chapter[5], application of CNN to simulated data from CMS has explained along with results.Lastly, in chapter[6] the conclusion of the project and future directions are given.

Chapter 2

LHC

The Large Hadron Collider (LHC) is the largest and the most powerful collider ever built. It accelerates protons to nearly the speed of light and collide them at four different location along its ring. At these points, the energy of collisions gets transformed into mass, spraying particles in all possible directions. The Compact Muon Solenoid (CMS) detector is located at one of these four collision points. It is designed to detect particles resulting from the collision. CMS detector has cylindrical shape made up of cylindrical layers of different sub-detectors such as silicone tracker, ECAL, HCAL, muon chamber, refer figure [1]. These sub-detectors have different functions, we are interested in tracker. The tracker tracks the trajectory of charged particles. More information about HLC can be found here [5]

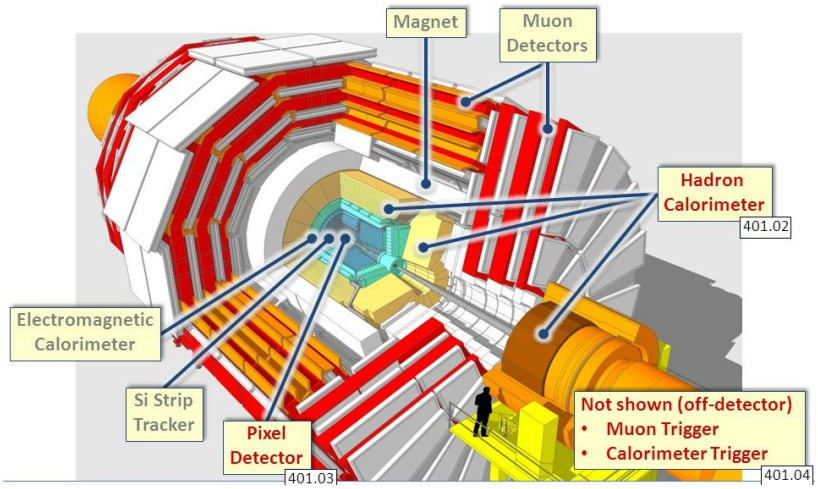

Figure 2.1: Internal Structure of CMS Detector

2.1 CMS Tracker

Figure [2.2] shows different components of the CMS detector. Electromagnetic Calorimeter is designed to measure the energy of particles that interact primarily through electromagnetic interaction. Hadron calorimeter is designed to detect particles that interact through strong nuclear force. Each tracker has strips of sensors. Each sensor is made up of a grid of pixels. When a particle travel through the tracker, these pixels record an electric signal. These signals get stored. The points of intersections are called hits. Ideally it is expected that when a particle passes through the tracker, only a pixel should lit up upon passing through a sensor but in the real scenario situation is very different. There is hardly a single pixel lit up upon crossing. Mostly it is the grid of pixels that lit up. This result in a region of hit rather than a single point. More information about tracker is given cited here [6]

In figure [2.2] trajectory of muon is shown. The hits are constructed using the hit clustering algorithm and from connecting these hits we can get the tracks. This process is called tracking. Momentum of any particle is a building block for picturing an event. In our report,we focus on predicting the transverse momentum (p_{T}) of a particle only by reconstructed hits with the help of machine learning techniques.

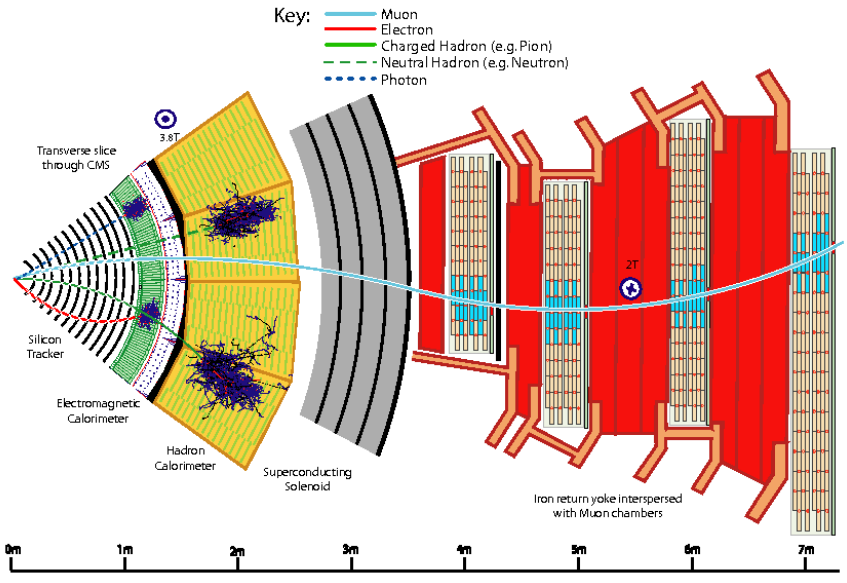

Figure 2.2: Trajectory of particle in CMS tracker

Figure [2.3] shows the upgrades phase 1 geometry of the CMS tracker in comparison to original geometry. The work shown in this thesis only consider phase 1 geometry. CMS started its working in phase 1 geometry in 2017. Tracker consist of five parts mainly, pixel, tracker inner barrel (TID), tracker inner dics (TID), tracker outer barrel (TOB), and tracker end cap region (TEC). Phase 1 geometry has 4 cylindrical layers and 3 tracker discs. These

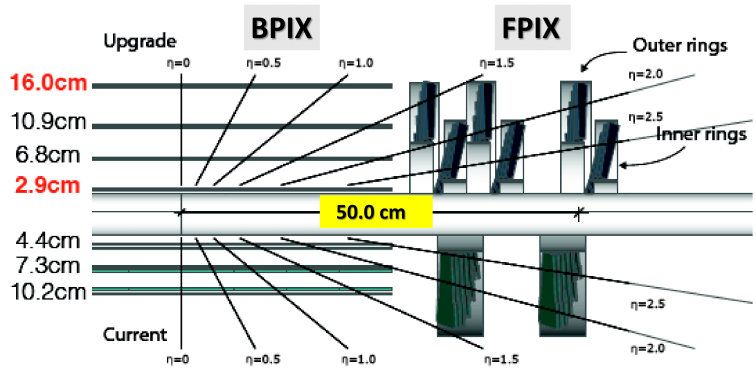

Figure 2.3: Phase 1 geometry of CMS tracker

four cylindrical layers placed at radii of 29, 68, 109, 160 mm and three disks in each of the forward regions placed at a distance from primary vertex at 291, 396 and 516 mm. This geometrical layout is optimized to give full 4-hit tracking coverage up to pseudorapidities of 2.5. More geometrical and material information is given in tracker technical design report (TDR) [7].

Along with pixel part, there are 4 cylidrical layers in TIB, 3 dics in TID, 6 cylidrical layers in TOB, and 9 dics in TEC. This information about number of layers in each section of tracker is important. During the analysis, using this information, it is easy to choose some layers and drop some layers. In this thesis work, analysis is done on different combination of barrel layers and end caps layers. Motivation behind it is that to find the optimum combination such that the input image will not be too crowded also not too sparse. This is to check the capability of network to understand the image data.

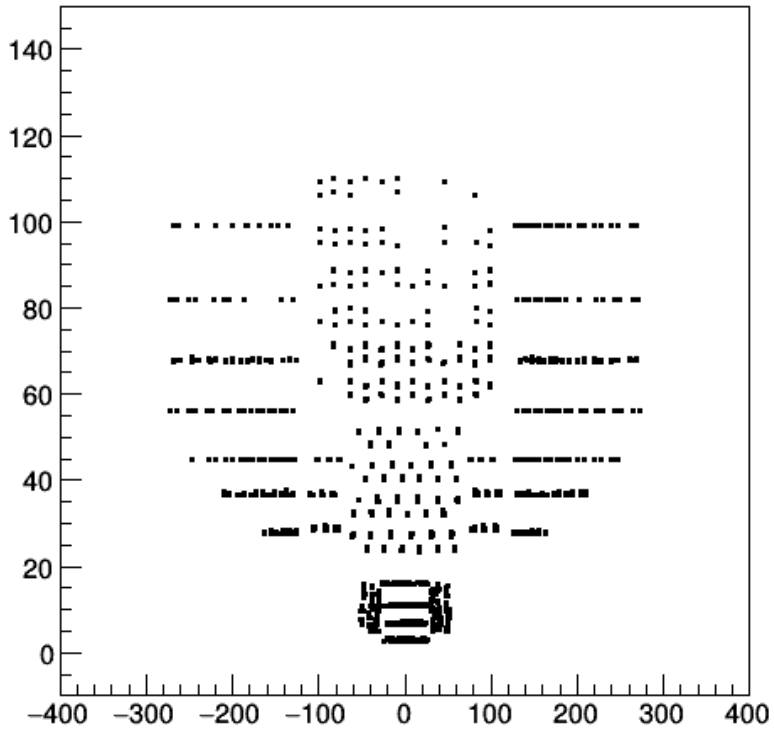

Figure 2.4: r-z plot from simulated data of CMS tracker

Figure [2.4] shows the real picture of how tracker layers gets populated after an event. It is a r-z plot of CMS tracker. The entries here are from just one $t\bar{t}$ event. This r-z plot is made up from the simulated data taken from CMS.

Chapter 3

Neural Networks

3.1 Artificial Neural Networks

Artificial neural networks(ANN) a.k.a neural networks are the small domain which comes under machine learning. These ANNs models itself after human brain by creating an artificial neural network, like in our brain, which works through algorithms that computer uses to learn new data.

There are many artificial intelligence algorithms are present now a days. Action of learning by the neural networks has been named as deep learning. Taking analogy from human brain, the basic unit is neuron. In artificial neural networks, the basic unit is called perceptron. Perceptron is capable of doing simple signal processing and bunch of such perceptron are connected together to make a large artificial neural network.

There are several applications of machine learning which includes classification that is classifying object into different classes, regression which do the prediction also others such as clustering, dimensionality reduction,and density estimation. Artificial neural networks analyses training samples to learn features about the data which have been labeled before analysis. For example, object recognition task. In object recognition, ANN has given a big amount of data of object which is under observation. After giving the data, ANN learn the repetitive patterns in given data and learns to categorize new data.

Neural networks can not be programmed direclty for their deep learning task. First ANNs should learn the information like humans developing brain. There are three methods of learning for ANN, These are as follows:

Supervised Learning: This one is a simple strategy. In this Labeled dataset presented to ANN. ANN goes through the data and algorithm modifies itself until it get the desirable result.

Unsupervised Learning: As name suggest, this strategy is used when there is no labeled data present for learning. ANN goes through the dataset and ANN's cost function determine how far off the network is from actual prediction. Using cost function's result neural network adjust to increase accuracy of the algorithm.

Reinforced learning: The neural network make a sequence of decision. For every right decision there is a reward and for every wrong decision there is a punishment. In this scenario, networks learn slowly overtime.

An Introduction to Statistical Learning : with Applications in R by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani is the best source to refer for basics neural networks. This book is cited here [8]

3.2 Convolutional Neural Networks

The convolutional neural network (CNN) is a specialized type of neural network. This convolutional neural network is specifically for data which comprises of images. Central part of the convolutional neural network is the convolutional layer because of which network got this name. Operation of convolution is performed by this layer.

In convolutional neural network, linear operation is performed by convolutional layer. In this linear operation input values get multiplied by set of weights. As images are two dimensional, this multiplication happened between two dimensional array of weights to array of input data via convolutional filter of kernel. A filter size is smaller than input data. Multiplication between filter and part of input array which is of size of filter is a dot product. Dot product is a element wise multiplication of arrays which later added to get single value. Because of this single value the operation is also referred as scalar product. These small

filters detects specific type of features present in the data. These filters spans the input data and learn about different features of image data which are present anywhere on the image. This property of the CNN is called translation in-variance.

In this thesis work, supervised learning method has been used. Neural network is presented with image data which labeled as "high p_{T} tracks" and "low p_{T} tracks" and performance of classification has shown in later chapters. The idea behind this was that the trajectory of high energy particles are straighter than that of low energy particles and CNN should use this feature to classify the new data. To learn more about CNN, refer to Keiron O'Shea, Ryan Nash: An Introduction to Convolutional Neural Networks [10].

3.3 Tensorflow

Tensorflow[16] is the machine learning framework created by Google's Brain team. It is an open source library. This library is used for numerical calculation and for large scale machine learning. Codes for tensorflow are written in Python.

Tensorflow commonly used for training and running of deep neural networks which do image recognition, handwritten digit classification, and speech recognition etc. Tensorflow's 1.10 version has been used in this thesis work for making the convolutional neural network model.

3.4 Working of Convolutional Neural Networks

Presently, convolutional neural network is most successful method for deep learning when image data is under consideration. Traditionally, features have to be extracted manually for learning but CNN learn features directly from the input images without any pre-processing of data. The structure of CNN consists of input layer, output layers, and several hidden layers between input and output layers. Examples of hidden layers are convolutional layers, max-pooling layers and fully connected layers. More information about working of CNN is cited here[10][11]

While making the CNN model, it's architecture varies. The number and type of layers are to be chosen specific to the problem. Many hidden layers get included while designing the CNN. Each layers is activated with a function. These functions are called activation functions.

1. **Convoutional Layer (CONV)**: These filters to make feature map.
2. **Rectified Linear Unit Layer (ReLU)**: This filter out negative values for faster training.
3. **Pooling Layer (POOL)**: It perform down-sampling and keeps only dominant parameters present in the data.
4. **Fully Connected Layers (FC)**: This layers computed probability score in form of vector. Size of the vector is the number of classes.

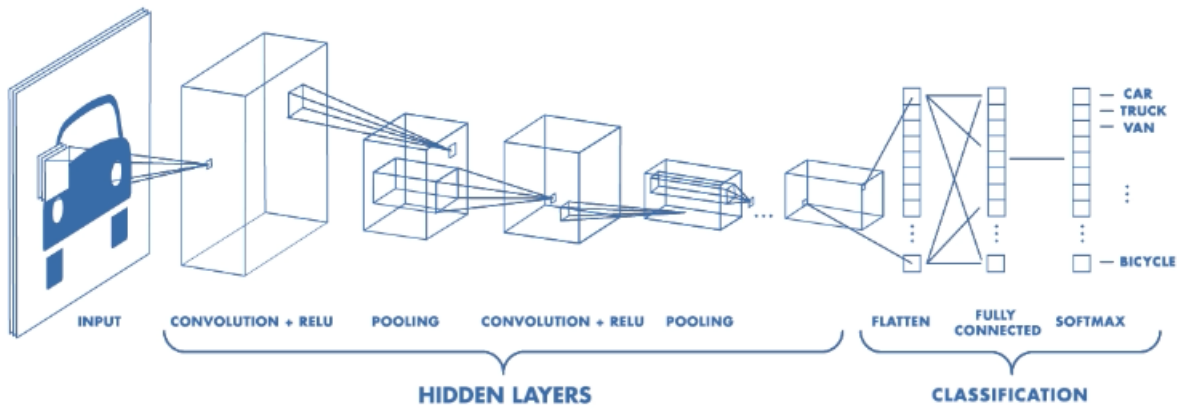

Figure 3.1: Example of convolutional neural network architecture. In this figure CNN is classifying input image in the classes of car, truck, van etc.

In figure[3.2], shown the CNN architecture that was being used in this study. It has 3 convolutional layers each followed by 1 max-pooling layer, and one fully connected layer. first convolutional layer that is conv1 has 32, 5x5 filters followed by 2x2 max-pooling layer. Second convolutional layer (conv2) has 64, 3x3, filters accompanied with 2x2 max-pooling layer. Third convolutional layer (conv3) has 64, 1x1 filters followed by 2x2 max-pooling

layer. At the end, there is a fully connected dense layer. This fully connected layer has 512 nodes. Every node from max-pooling layer from conv3 is connected to this 512 nodes of the dense layer. Probabilistic output is given by function softmax.

```
# Create model
def conv_net(x, n_classes, dropout, reuse, is_training):
    # Define a scope for reusing the variables
    with tf.variable_scope('ConvNet', reuse=reuse):

        # Convolution Layer with 32 filters and a kernel size of 5
        conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
        # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
        conv1 = tf.layers.max_pooling2d(conv1, 2, 2)

        # Convolution Layer with 32 filters and a kernel size of 5
        conv2 = tf.layers.conv2d(conv1, 64, 3, activation=tf.nn.relu)
        # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
        conv2 = tf.layers.max_pooling2d(conv2, 2, 2)

        # Convolution Layer with 32 filters and a kernel size of 5
        conv3 = tf.layers.conv2d(conv2, 64, 1, activation=tf.nn.relu)
        # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
        conv3 = tf.layers.max_pooling2d(conv3, 2, 2)

        # Flatten the data to a 1-D vector for the fully connected layer
        fc1 = tf.contrib.layers.flatten(conv3)

        # Fully connected layer (in contrib folder for now)
        fc1 = tf.layers.dense(fc1, 512)
        # Apply Dropout (if is_training is False, dropout is not applied)
        fc1 = tf.layers.dropout(fc1, rate=dropout, training=is_training)

        # Output layer, class prediction
        out = tf.layers.dense(fc1, n_classes)
        # Because 'softmax_cross_entropy_with_logits' already apply softmax,
        # we only apply softmax to testing network
        out = tf.nn.softmax(out) if not is_training else out

    return out
```

Figure 3.2: CNN architecture used in this thesis work.

Figure[3.2] shows the CNN architecture that has been used in this thesis. After making changes to all hyper-parameters, with trial and error, this was the architecture that was giving the higher performance.

3.4.1 CNN Training Phase

Data Collection and Labels

Data collection for the toy model and the CMS tracker was very different. For toy model there, C++ code script has written and data is govern via this script on local machine using Root[4] is a data analysis framework of CERN. In figure [3.3], image on the left shows the data from the toy model. This toy model image has 100 tracks with smearing ± 1. Image on the right is from CMS tracker. Data is generated from $t\bar{t}$ process. Image is of one $t\bar{t}$ event.

In the toy model[4], total images used for training are 20,000. There are 2 classes for classification task. First class is of events which contain high energy tracks. Calling this class as a signal. Second class is of events which contain no high energy tracks that is all low energy tracks, calling this class as a background. Definition of high $p_{\mathrm{T}}$and low $p_{\mathrm{T}}$is given in chapter[4].Each class contain 10,000 images. New image data sets have been generated after changing the data conditions.

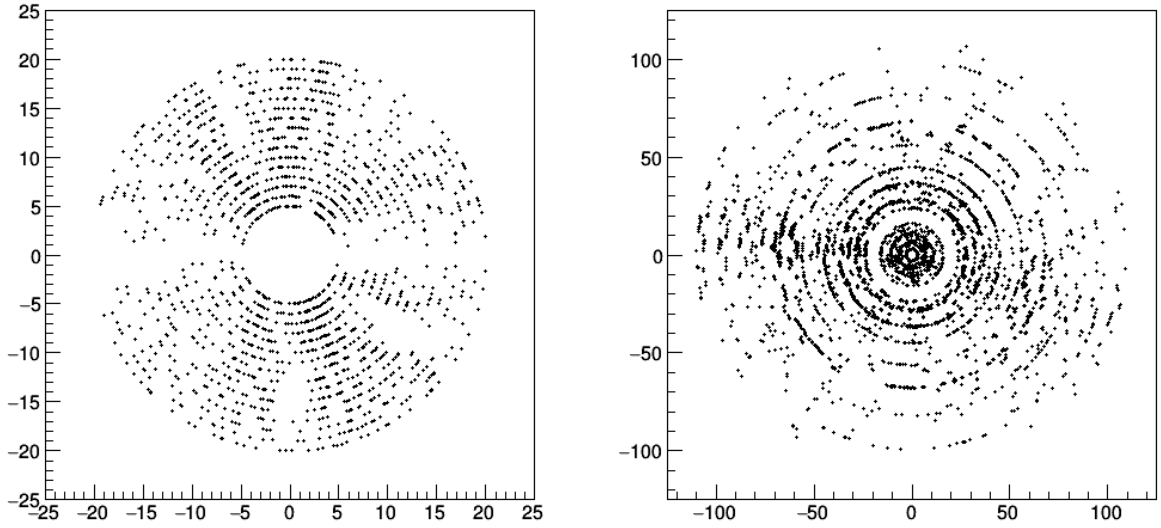

Figure 3.3: To the left is the input image data of the toy model and to the right is the input image data from the CMS tracker.

Training and Testing Phase of the CNN

In the training phase, all 10,000 images of each classes were shown to the CNN. CNN learns features from this data. Learning of CNN is depended on something called hyper-parameters. This hyper-parameters include number of epochs, learning rate, dropout rate, batch size, image size etc. There is no straightforward way to assign these parameters. To assign the value, one has to rely on trial and error method.

After training of CNN is done, it comes the testing phase. One important point here is that, in the training phase a parameter called loss function. This loss function corresponds to the quality of the CNN. After training, value is loss function is tends to zero then the

network is able to distinguish between given classes. Sometimes having loss function very close to zero is also an implication of over training.

Hyper-parameters

- **Batch size** : Batch size is the amount of data that goes into CNN in each iteration. Batch size is depended on processing power. High batch size demands high processing power.

- **Dropout** : This parameter has fractional value between [0,1]. After neurons are linked to each other, dropout parameter make some links inactive. For example, if dropout is 0.5 then 50 percent of links are dropped after training. This helps to fight over training of network.

- **Epochs** : One epoch means that network has seen all of the data once. Number of epochs are depended on loss function.

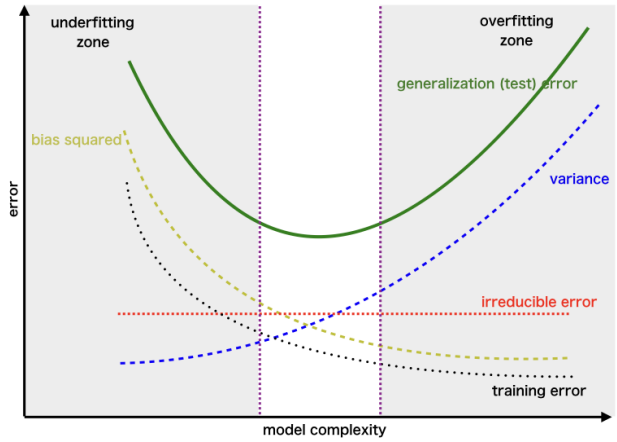

Figure 3.4: This shows the bias-variance trade off of neural network. If network has more biasness then it is over-trained whereas if network has more variance then it is under-trained. This is very important concept in the study of machine learning.

In the figure[3.4], how model complexity changes by changing hyper-parameters is shown. Hyper-parameters are responsible for loss function which corresponds to training and testing error. Best neural network is the network which gives low testing error. Additionally, from this bias-variance trade of we can estimate optimum number of epochs by plotting number

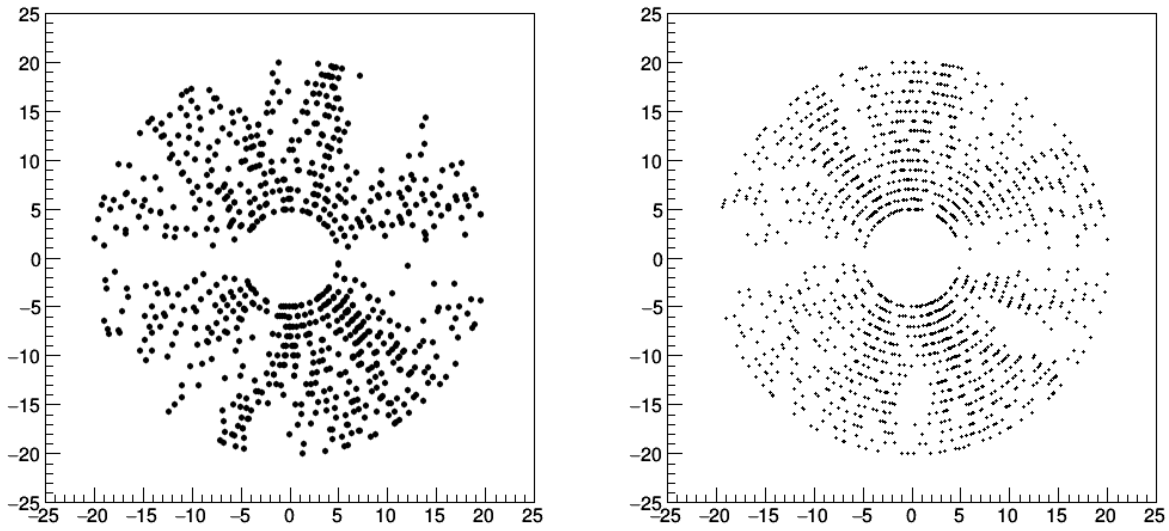

Figure 3.5: These images shows the different marker sizes used in plotting data. Uses of smaller and bigger marker size signifies the amount of data in the image as bigger dot covers more pixels.

of epochs versus testing error. Optimum number will be the one which has low testing error value.

In addition to the hyper-parameters describe above, there is a another parameter called marker size which has played essential role in learning of CNN in this thesis. For example, in figure[3.5], image on the left represent an event which has 50 tracks and image on the right represent an event with has 100 tracks. After trying various marker sizes for plotting, it was observe that 50 tracks case need bigger marker size as there is less data points on the canvas. This sparsity of data cause difficulty in learning for CNN. By making marker size bigger, CNN is working better. In 100 tracks case, data points are two times of 50 tracks case. This is making canvas crowded and due to bigger marker size points were overlapping. Using smaller marker size, this overlapping got less and CNN performance got better.

ROC curve

A Receiver operating characteristic (ROC) curve is a plot that represent capability of binary classifier. ROC curve is made by plotting false positive rate (FPR) and true positive rate

(TPR) at different thresholds. TPR is also called probability of identifying in machine learning.

3.4.2 CPU vs GPU

The major distinction between GPU and CPU is that CPU is more versatile and it is design to handle a wide-range of tasks rapidly. This rapidness of CPU is depended on the clock speed. Performance of CPU is also dependant on number of task running at the same time. Whereas GPU is specifically design for multimedia. GPU process images and video really fast concurrently. GPU has huge amount of parallel processing power. More details are given here [17]

This thesis work uses image data for analysis. Analysis time has significantly reduced after using GPU. For example, 100 training steps took approximately 24 seconds to finish by using CPU but same steps took nearly 4 seconds on GPU. System Specifications are given below :
CPU - Intel® Core™ i7-4710MQ CPU @ 2.50GHz 8
GPU - GeForce GTX 860M/PCIe/SSE2 2GB
RAM - 8 GB

Chapter 4

Toy Tracker Model

4.1 2D Toy Tracker Model Specifications

2D model specifications :
This model consists of total 16 concentric layers. These layers are unit distance apart from consecutive layers. The smallest radius is 5 and with the increment of one, it goes till 20. There is no thickness associated with the toy tracker layer. The name 2D comes from setting z component of momentum vector (p_z) value to zero. In this model, high p_{T}values are constrained in [100,150] GeV interval and low p_{T}values are from interval [10,50] GeV. p_{T}distribution is uniform. Following table [4.1] summaries 2D toy model :

$p_z = 0\ GeV$
16 tracker layers
High p_t - [100,150] GeV
Low p_t - [10,50] GeV
Uniform p_t distribution.
B = 1 T

Table 4.1: Specifications of 2D Model

4.2 Smearing of co-ordinates

What is a hit?

Imagine just a single particle is generated after proton-proton collision. This particle travels through tracker and interact with material of pixel, TOB and TIB sections. These part are made up of strip and pixel tracker[7]. Ideally when particle passes through a pixel, only that pixel should lit up but in reality a grid of pixel lit up. This happens due to many reason suck as residual charge present in pixel, malfunctioning of pixel, etc. There are various statistical techniques used to get approximately right hit point. A presentation was given by Jay Hauser Frank Hartmann on indico about tracking which was very helpful. Presentation is cited here[3].

Smearing of a hit

In toy tracker model, calculated intersection points are ideal. In CMS tracker, reconstructed hits are not exactly at intersection but around the exact point. Processes of smearing is basically adding uncertainty to intersection points. In this problem values of x are deviated by ± 0.02 that is by 2 percent such that value of y still lie on the detector layer.

$$x_s = x \pm 0.02$$

$$y_s = (r_d^2 - (x_s)^2)^{\frac{1}{2}}$$

Here on-wards only smeared co-ordinates are considered.For each p_x and p_y value there exist an unique track. Each track has 16 intersection points that is 32 co-ordinates (x1....x16;y1....y16). Simulation has generated about 1 million tracks, each having unique p_t.

A typical example of three tracks in toy model is shown in figure [4.1]. Figure on the left shown three tracks without smearing and figure on the right shows smearing of coordinates by value ± 1.

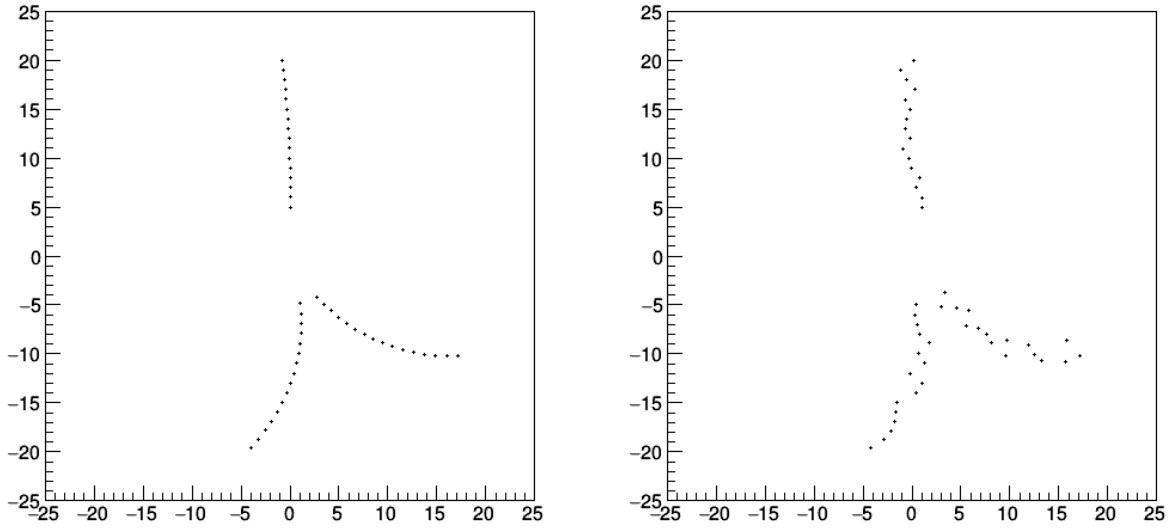

Figure 4.1: Left: Tracks with no Smearing Right: Track with smearing amount ± 1

4.3 Theory Behind Toy Model

According to electrodynamics, moving charged particles follow curved paths in a magnetic field. Positive charges bend in the counter-clockwise direction while negative charged particles bend in the clockwise direction. This behaviour is govern by following mathematical formalization: For a classical case: force experience by moving charged in magnetic field is given by,

$$F = q(VxB)$$

Particle is curving, hence there is a centripetal force,

$$F = \frac{mV^2}{r}$$

Comparing the above two equations, we get,

$$r = p_t/(qxB) = (p_x^2 + p_y^2 + p_z^2)^{\frac{1}{2}}/(qxB)$$

From the above equation, it is self-explanatory that more the momentum of the charged particle, less curve will its path be and vice versa.
To simulate the problem, only two dimensional case is considered. That is,

$$p_z = 0\, GeV \qquad B = 1\bar{z}$$

This constraints the particle to move in XY plane.
The circles centred at origin are the detector layers in our simulation. Intersection points of tra- jectory and layers represent hits. We are only considering a trajectory till the last detector layer because afterwards there is no magnetic field and particle will not bend. In the real detector, magnetic field is produced using a superconducting solenoid which is placed between HCAL and muon chamber that is why direction of magnetic field reverses later.As you see in figure [3] that the muon's trajectory changes after passing the super conducting solenoid. Below is the calculation for intersection points,
Equation of detector layer:

$$x^2 + y^2 = rd^2$$

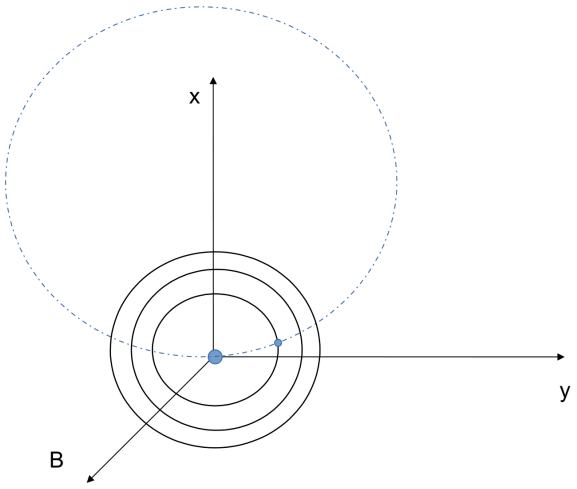

Figure 4.2: Arrangement of tracker laters in Toy Model

Equation of trajectory:

Legends	Correspondence
r_d	Radius of a detector layer
r	Radius of trajectory of a particle
a	x co-ordinate of the centre of the trajectory of a particle
b	y co-ordinate of the centre of the trajectory of a particle

Table 4.2: Legends - Toy Model

$$(x-a)^2 + (y-b)^2 = r^2$$

After solving this equation, intersection points are:

$$y = \frac{b*r_d^2}{2*r^2} \pm \frac{a*r_d*(4*r^2-r_d^2)^{\frac{1}{2}}}{2*r^2} \qquad x = \frac{r_d^2}{2*a} - \frac{b*y}{a}$$

Also by using geometry,

$$b = \pm p_x \qquad a = \mp p_y$$

In equation for y, there is a term in root $4r^2 - r_d^2$ and that is the limiting factor. This factor take into account the track that curls inside without reaching a particular value of r_d. Here tracks which satisfy $2r > r_d$ are considered.

4.4 Results

Two CNN classes are called signal and background. Signal class contains images of events with high p_{T} track. Number of high p_{T}tracks will be given wherever necessary. Background class contains images of events which has no high p_{T}track(s). Values of "high" and "low" are given in table [4.1]. Results are shown in form of ROC curve and classes distribution. For reference, definition of ROC curve and class distribution is given below:

• ROC curve - higher the area under ROC curve, better is the classifier. It gives accuracy of the classifier.

• Class distribution - Farther the two histograms, better the classification. This means that network is understanding the difference between two classes.

4.4.1 100 Tracks

Total Number of tracks	100
Signal	1 high $p_{\mathrm{T}}$and 99 low p_{T}
Background	100 low p_{T}
Smearing amount	± 1
Training images	10,000
Testing images	10,000
Marker size	0.4

Table 4.3: 100 tracks case specification table

In this case, there are total 100 tracks per event. Each hit point is smeared by value ± 1. Signal Class corresponds to events with total 100 tracks in which 1 track is high $p_{\mathrm{T}}$and other 99 tracks are low p_{T}. CNN performance check has done on these classes. Figure [4.3] is an example of input images which are given to the CNN. Total number images for training and testing both are 10,000. Marker size is 0.4 .

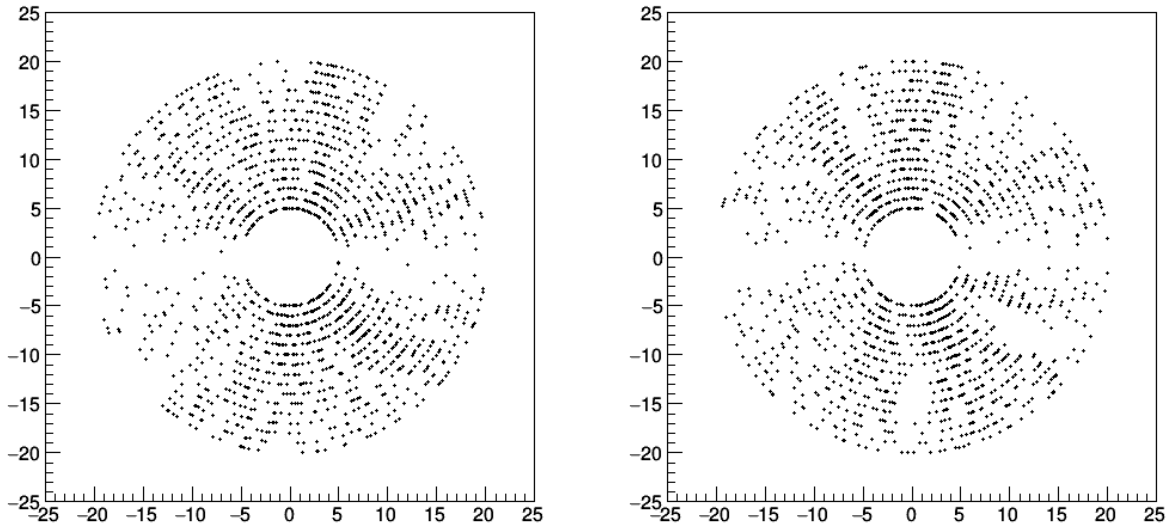

Figure 4.3: Image to the left has event with total 100 tracks which has 1 high p_{T}track and image to the right has all low p_{T}tracks. All hits are smeared by amount ± 1.

Figure[4.4] shows the performance of CNN model. Area of ROC curve is 0.97. It means that given 100 images to CNN, CNN can classify 97 out of 100 images correctly. This means 97% accuracy of the network. Class distribution of both training and testing are separated

from each other, this is also states that the CNN is learning the feature from the data. In other words, network is understanding, what is high $p_{\mathrm{T}}$and what is low p_{T}which has stated earlier.

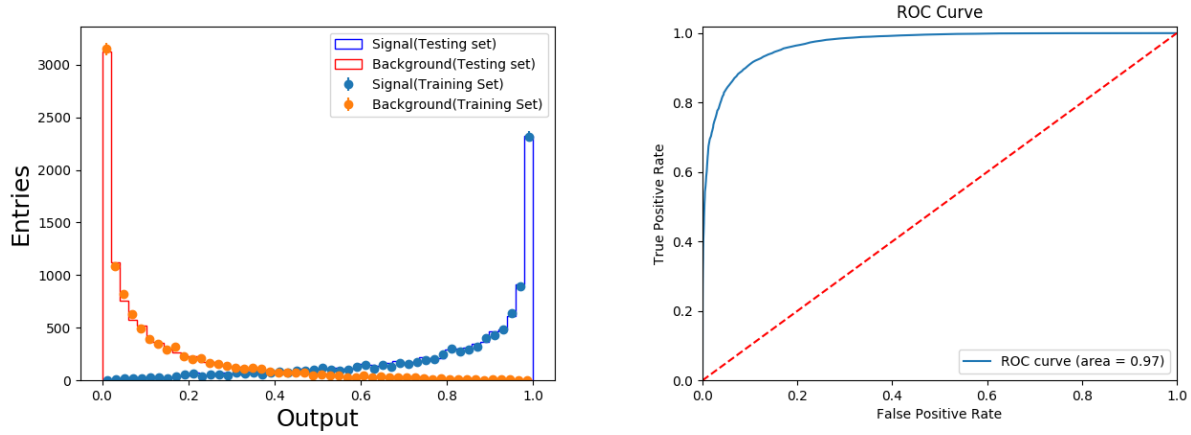

Figure 4.4: On the left class Distribution is shown and on the right there is a ROC curve.

4.4.2 200 Tracks

Total Number of tracks	200
Signal	1 high $p_{\mathrm{T}}$and 199 low p_{T}
Background	200 low p_{T}
Smearing amount	± 1
Training images	10,000
Testing images	10,000
Marker size	0.4

Table 4.4: 200 tracks case specification table

In this case, there are total 200 tracks per event. Signal class contains events with 199 low p_{T}tracks and 1 high p_{T}track and background class contain events with all 200 low p_{T}tracks. All hit positions are smeared by amount ± 1. Figure [4.5] shows the images that are given to CNN for training. Total number images for training and testing both are 10,000. Marker size is 0.4 .

Figure[4.6] shows the performance of CNN model. Area of ROC curve is 0.88. It means that given 100 images to CNN, CNN can classify 88 out of 100 images correctly. This means

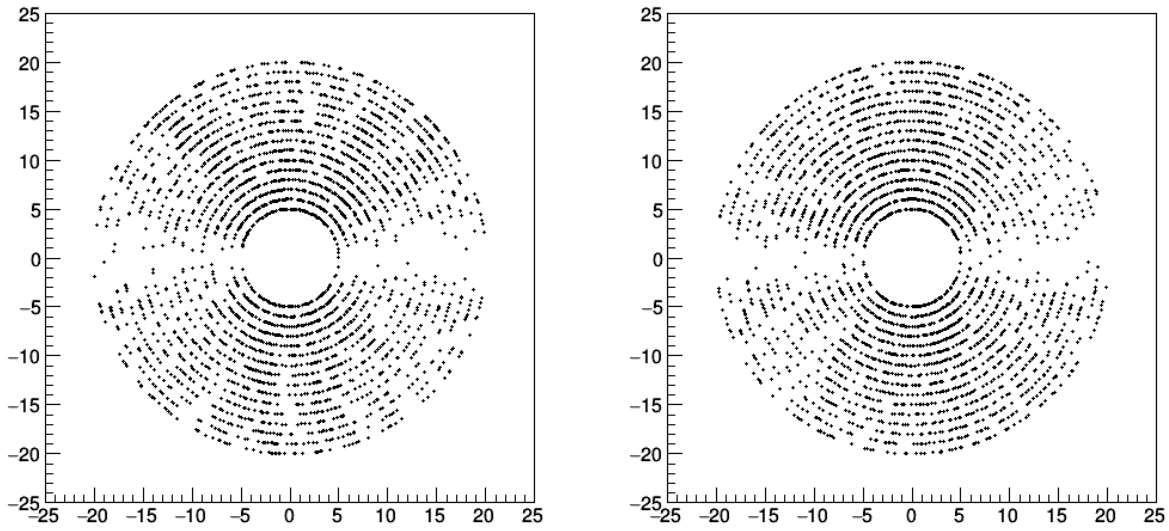

Figure 4.5: Image to the left has event with total 200 tracks which has 1 high p_{T}track and image to the right has all 200 low p_{T}tracks. All hits are smeared by amount ± 1.

88% accuracy of the network. Class distribution of both training and testing are not well separated from each other like in 100 track case. This is because images are too complex for CNN to understand the feature clearly. In other words, network is sort of understanding, what is high $p_{\mathrm{T}}$and what is low p_{T}which has stated earlier.

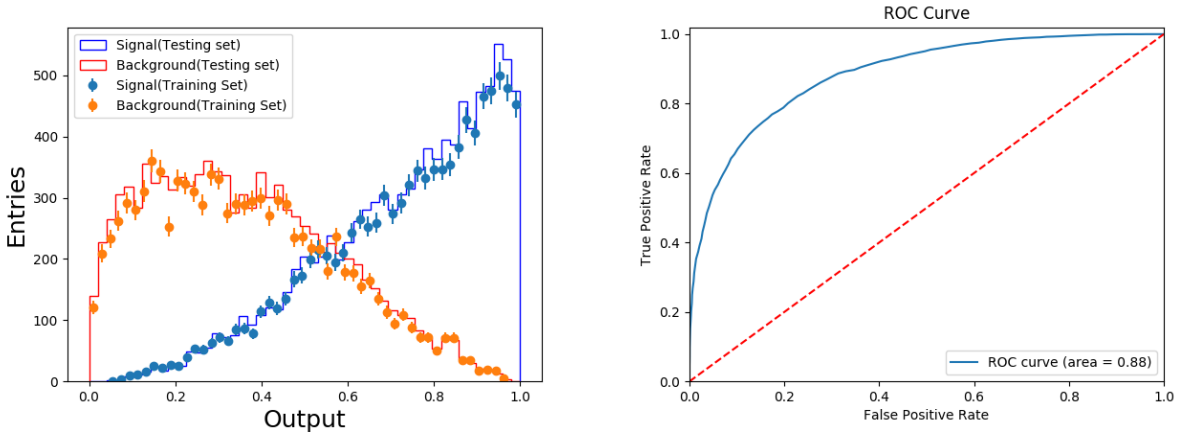

Figure 4.6: On the left class Distribution is shown and on the right there is a ROC curve.

4.4.3 50 Tracks

Total Number of tracks	50
Signal	1 high $p_{\mathrm{T}}$and 49 low p_{T}
Background	49 low p_{T}
Smearing amount	± 1
Training images	10,000
Testing images	10,000
Marker size	0.5

Table 4.5: 50 tracks case specification table

In this case, there are total 200 tracks per event. Signal class contains events with 49 low p_{T}tracks and 1 high p_{T}track and background class contain events with all 50 low p_{T}tracks. All hit positions are smeared by amount ± 1. Figure [4.7] shows the images that are given to CNN for training. Total number images for training and testing both are 10,000. Marker size is 0.5 . It is important here to see that the marker size has changed. This is because there is sparsity of data in the input images. Due to this factor, bigger marker size adds more information in images hence CNN can able learn the feature of the image data.

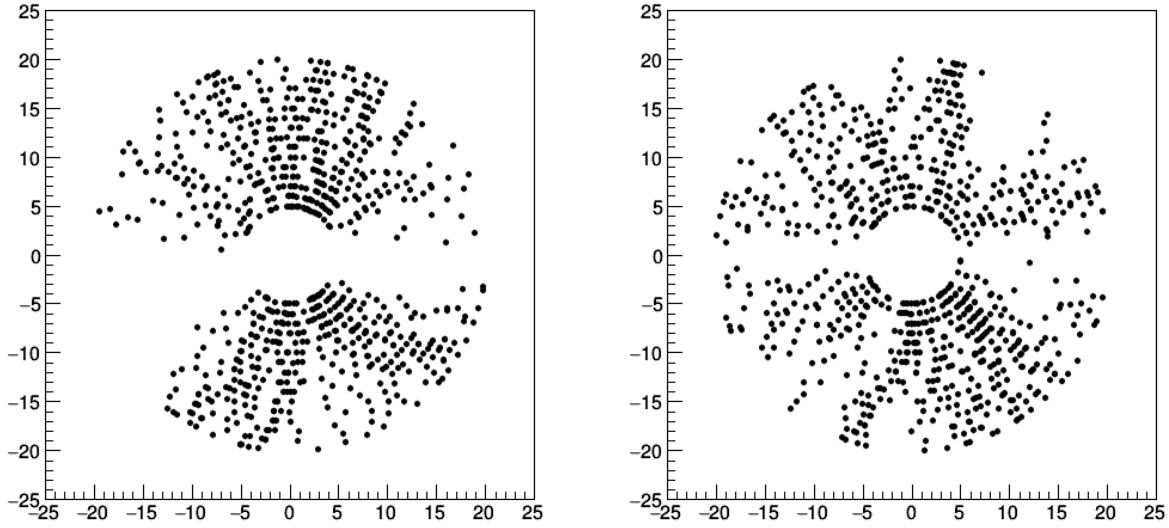

Figure 4.7: Image to the left has event with total 200 tracks which has 1 high p_{T}track and image to the right has all 200 low p_{T}tracks. All hits are smeared by amount ± 1.

Data is less complex as compared to the earlier 100 tracks and 200 tracks cases. It was

expected that CNN will work better in this 50 track case. Area of ROC curve is 0.99. Again, It means that given 100 images to CNN, CNN can classify 99 out of 100 images correctly. This means 99% accuracy of the network. Class distribution is very well separated. This means that CNN is understanding features of data and it is easily able to identify which image belong to signal class and which image belongs to background.

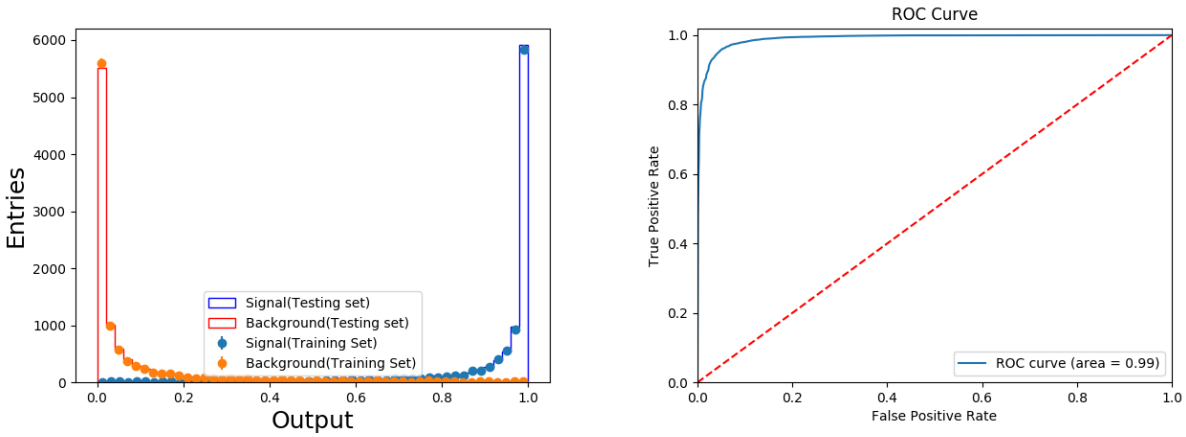

Figure 4.8: On the left class Distribution is shown and on the right there is a ROC curve.

4.5 3D Tracker

In this model, p_z value is not equal to zero. Now, due to p_z is not zero, there is an overlap between the (x,y) coordinate. In other words, some rechits may have same (x,y) coordinates but different Z coordinate. This is problematic because input images to CNN are two-dimensional projection of rechits on XY-plane. So in data there may be overlapping tracks that CNN will not learn about. In figure [4.9], XY overlap for 6 tracks are shown. In the left, there are total 6 tracks but only 3 tracks are visible. This is because other 3 tracks have same X and Y coordinates. This is shown in the right image of the figure [4.9]. Vertical axis is the Z axis. As it is shown in 3 dimensional image, every point is unique in space.

To tackle this XY overlap issue, geometry of the tracker has to be disturbed a little. By referring the illustration in figure [4.10], tracker layers are cylindrical in toy tracker model. Transforming these cylinder into a cone such that the (x,y) coordinates little bit so that no 2 point has same XY coordinates. After transforming cylindrical layers into cones next step

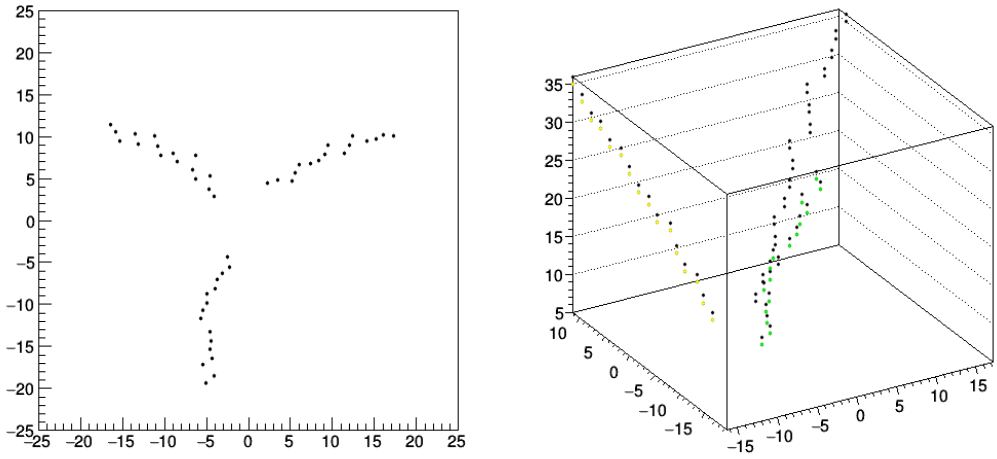

Figure 4.9: Left: 2D view Right: 3D view

is to take the projection of these transformed conical layers into xy plane. As the no point will have same coordinates, every point in the xy plane will be distinct. There will be no overlap of coordinates.

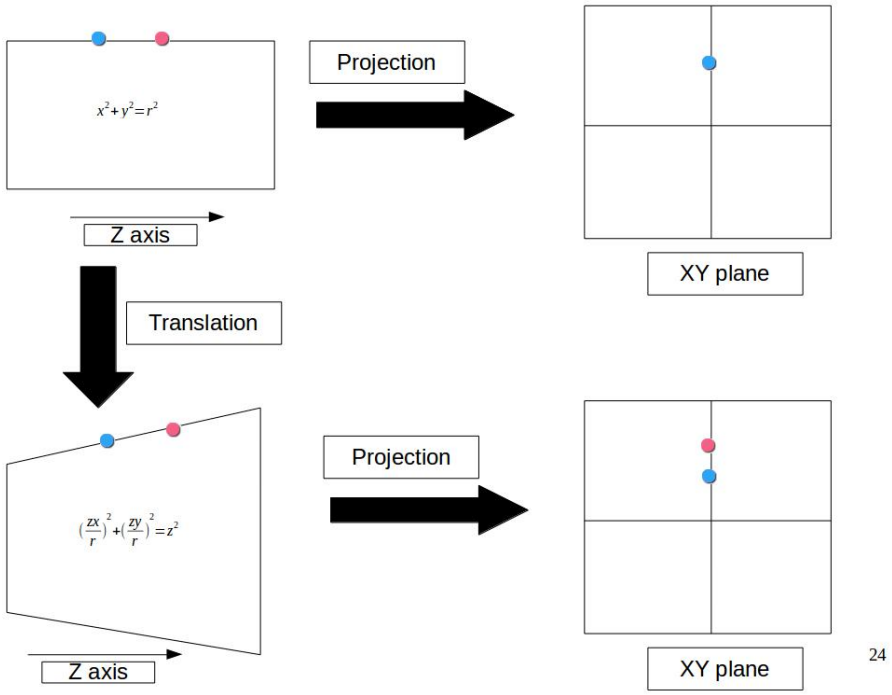

Figure 4.10: Transformation to lift xy overlap

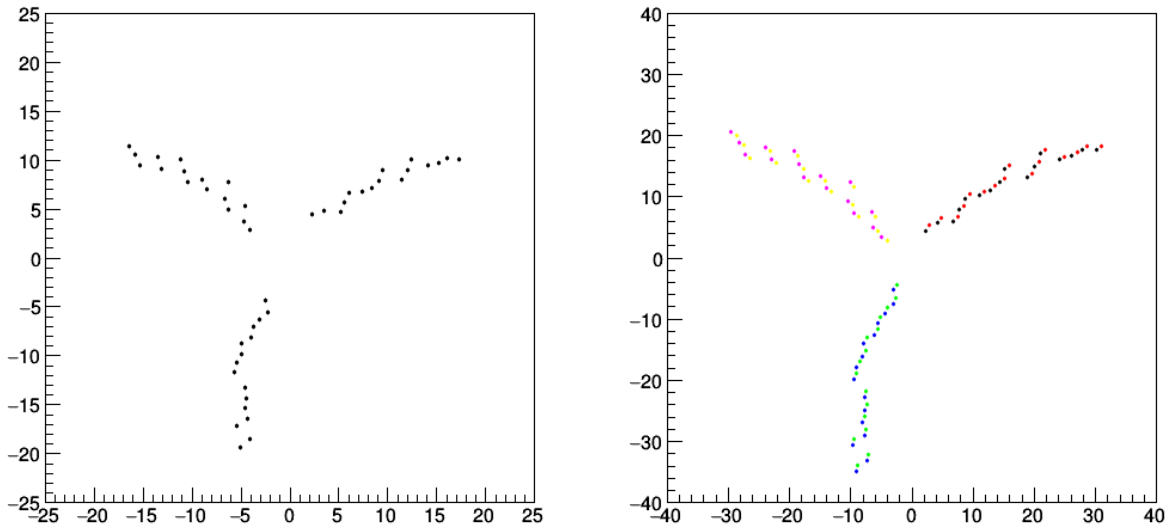

Figure 4.11: Left: Before transformation Right: After transformation

In figure [4.11], left images is the one without the coordinate transform. Right image show tracks after the coordinate transform. Different color in the right image shows the different tracks. Now, there are 6 tracks in the images. Also, one should see that the scale of the image has changed. Now, these are the images will be given to CNN to train on.

4.6 Result

In this section, performance of CNN after applying the transformation has studied. Signal and background classes are bit different that previous section. These classes are explicitly mention in each section below.

4.6.1 100 Tracks - Case 1

There are total 100 tracks per event. Figure [4.12], left image is from signal class. Signal class consist of 49 overlapping low p_{T}tracks along with 1 non-overlapping low p_{T}track and 1 high p_{T}track. Background class has 50 overlapping low p_{T}tracks. Total number of training and testing images are 10,000 and marker size is kept at 0.4 .

Total Number of tracks	100
Signal	1 high p_{T}track, 1 low p_{T}track and 49 on 49 low p_{T}tracks
Background	50 on 50 low p_{T}track
Smearing amount	± 1
Training images	10,000
Testing images	10,000
Marker size	0.4

Table 4.6: 100 tracks case 1 specification table for 3D toy tracker model

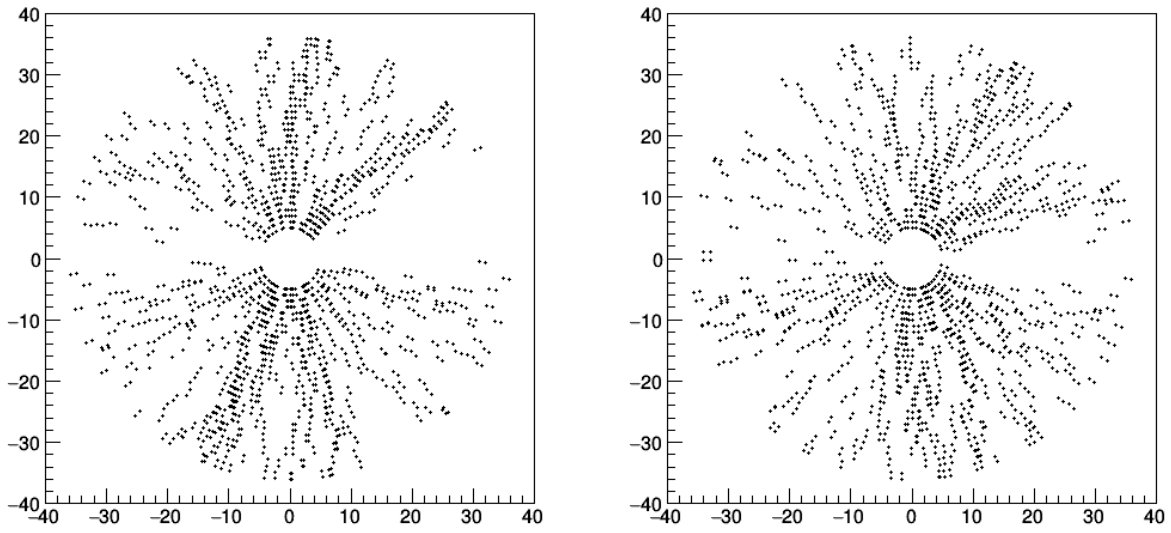

Figure 4.12: Image to the left has event with total 100 tracks which has 1 high p_{T}track, 1 low p_{T}track along with 49 on 49 low p_{T}tracks and image to the right has all 200 low p_{T}tracks with 50 on 50 low p_{T}tracks. All hits are smeared by amount ± 1.

In figure [4.13], area of ROC curve is 0.99 which means 99 % of accuracy. The accuracy for this case is high than the 100 track case in 2d toy tracker model is because the images here are less dense than the later case. This less density comes from the overlapping of coordinates. After removing the overlap using the conical transformation the range of xy-axes is also changing. First it was 25 cm and after transforming range is 40 cm. This factor also makes images less dense and that is why CNN is learning the features in the data.

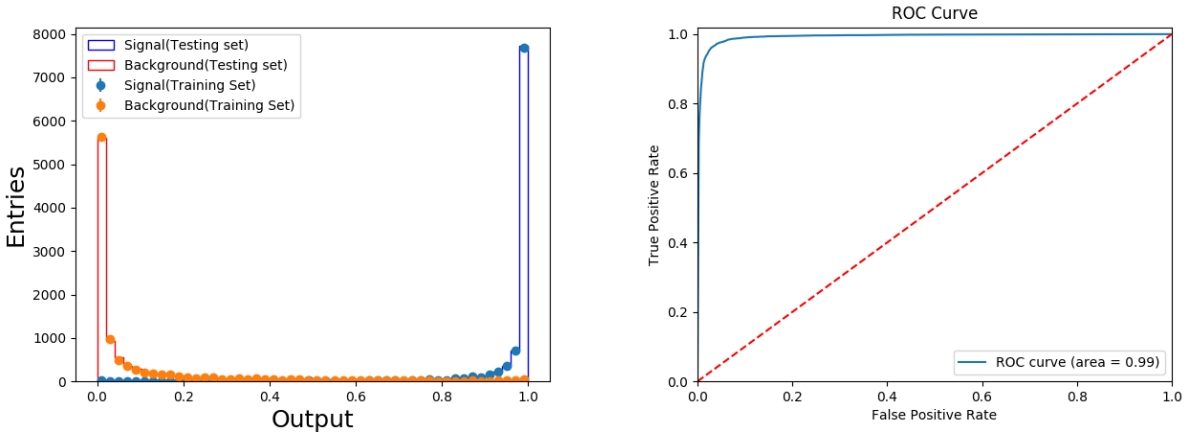

Figure 4.13: On the left class Distribution is shown and on the right, there is a ROC curve.

Total Number of tracks	100
Signal	1 high p_{T}track, 49 non-overlapping and 25 on 25 low p_{T}tracks
Background	50 non-overlapping low p_{T}tracks and 25 on 25 low p_{T}tracks
Smearing amount	± 1
Training images	10,000
Testing images	10,000
Marker size	0.4

Table 4.7: 100 tracks case 2 specification table for 3D toy tracker model

4.6.2 100 Tracks - Case 2

There are again total 100 tracks per event. Figure [4.14], left image is from signal class. Signal class consist of 49 non-overlapping low p_{T}tracks along with 25 on 25 overlapping low p_{T}track and 1 high p_{T}track. Background class has 50 non-overlapping low p_{T}tracks and 25 on 25 overlapping low p_{T}tracks. Total number of training and testing images are 10,000 and marker size is kept at 0.4 .

In figure [4.15], both training and testing classes are well separated and points are matching. ROC curve has area 0.99 which gives 99 % of accuracy. CNN is learning features of this dataset and able to classify testing samples into respective classes with 99 % of accuracy.

So far, convolutional neural network model is working on toy tracker model with high accuracy of classification. Accuracy had changed when the input images were made more complex than before but accuracy did not drop drastically. Taking this CNN model further,

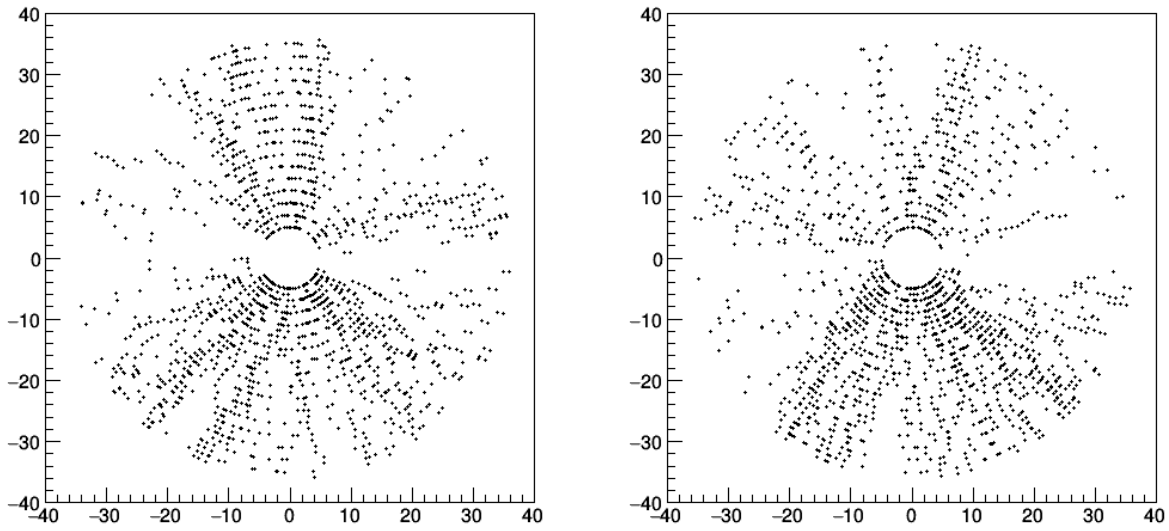

Figure 4.14: Image to the left has event with total 100 tracks which has 1 high p_{T}track, 49 non-overlapping low p_{T}track along with 25 on 25 low p_{T}tracks and image to the right has all 100 low p_{T}tracks with 50 non-overlapping tracks along with 25 on 25 overlapping low p_{T}tracks. All hits are smeared by amount ± 1.

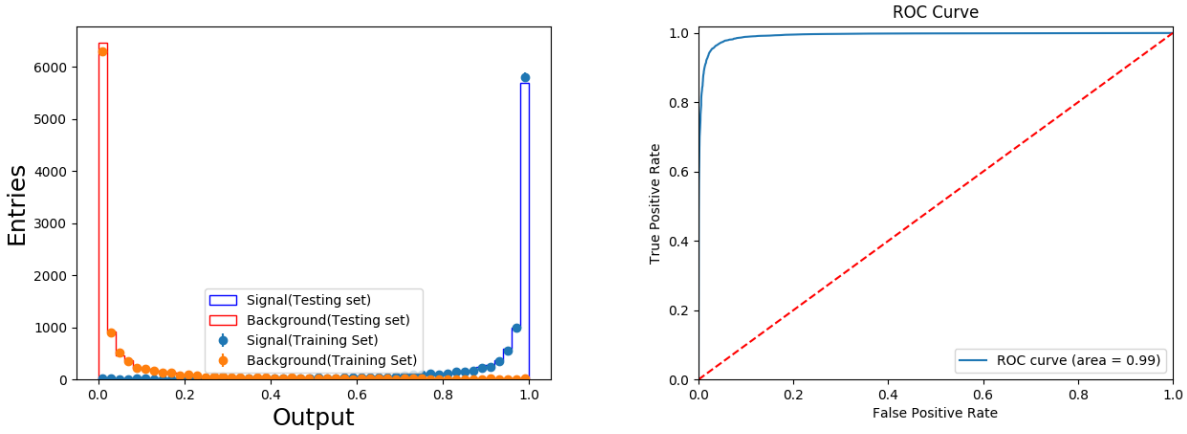

Figure 4.15: On the left class Distribution is shown and on the right, there is a ROC curve.

it is time to tackle the real problem.

Chapter 5

Applying to CMS Data

5.1 Setup

After application of CNN model to the toy detector, this chapter talks about performance of CNN when applied to the data of CMS tracker. Simulation of $t\bar{t}$ events is used to construct images. Figure[5.1] shows the hits distribution in CMS tracker after an event.

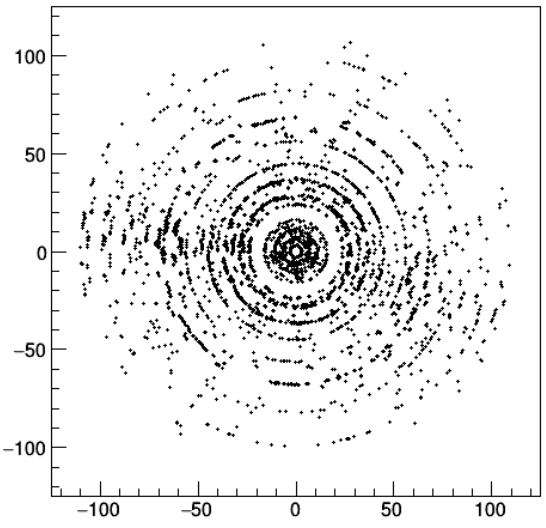

Figure 5.1: This is this plot of an $t\bar{t}$ event in CMS tracker.

Let's quickly summarize the CMS tracker geometry. Pixel region has total four inner barrel layers and three inner discs layers. In inner barrel region, there are four cylindrical layers. Tracker inner disc region has three discs. Tracker outer barrel region, there are six cylindrical layers ans tracker end-cap region has nine discs.

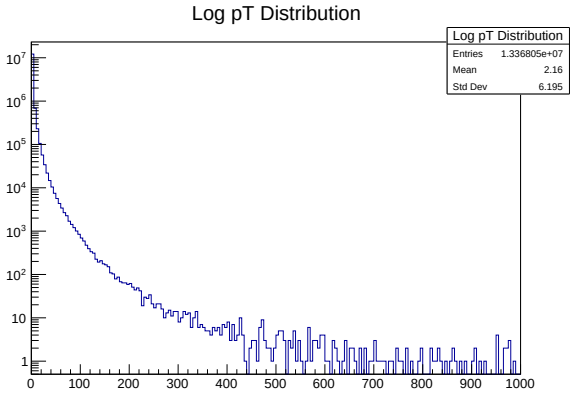

Figure 5.2: This is a log distribution of p_{T}in $t\bar{t}$ events over 50000 events. Each entry corresponds to individual particle.

Figure[5.2] show the logarithmic distribution of p_{T}in 50,000 $t\bar{t}$ events. As it is seen from the plot, majority of the values are approximately below 25 GeV. This plot is very helpful in further analysis because it tells that ration of number of high energy tracks to low energy tracks is small.

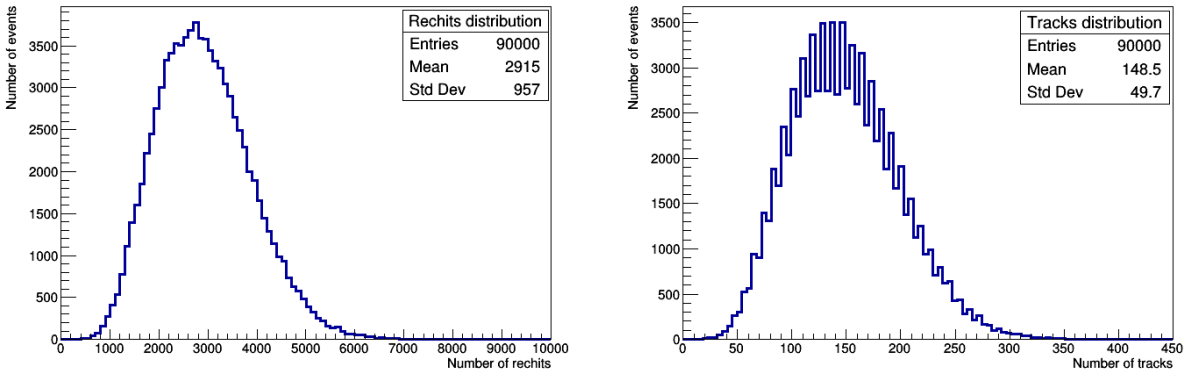

Figure 5.3: First plot from the left show the distribution of rec-hits and plot in the right shows the distribution of number of tracks in 90,000 $t\bar{t}$ events

In figure[5.3], plots of rec-hits distribution and number of tracks distribution are shown. Each plot is peaking at some value. These plots play an important role in choosing the

classification classes for CNN.

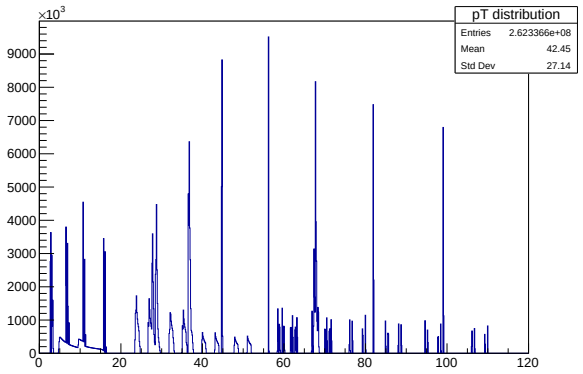

Figure 5.4: This plot shows the density of hits in tracker layer. This data is from 90,000 $t\bar{t}$ events.

Figure[5.4] shows the hit density in each layer of tracker. There are more entries in the region between 40 and 60 cm that in pixel part. This shows that particles are decaying after passing from pixel tracker.Taking into account all this information about simulated data, let's check the performance of CNN.

In toy model, there was a control on parameters such as number of hits, number of tracks and track p_{T}etc. This is not the case with simulated data. There is no control over number of tracks or number of hits per event. These parameters vary in each event as well as these parameters vary within the same class as well. This is the main difficulty with simulated data. For example, event with 50 track can also belongs to signal class as well as event with 120 track. Because of this, CNN may get confused, as there is no way to figure out exactly what CNN is learning, it may learn first than less rec-hits density in the input image is the feature but it may first high rec-hits density images in next batch. In the following section[5.2], intensive study has done on the tracker geometry to simplify the image data for CNN.

5.2 Result

Figure[5.5] shows plots corresponding to both each classes. In top right corner is plot of an event which has all track with p_{T}less than 70 GeV. This is the background. In the top right

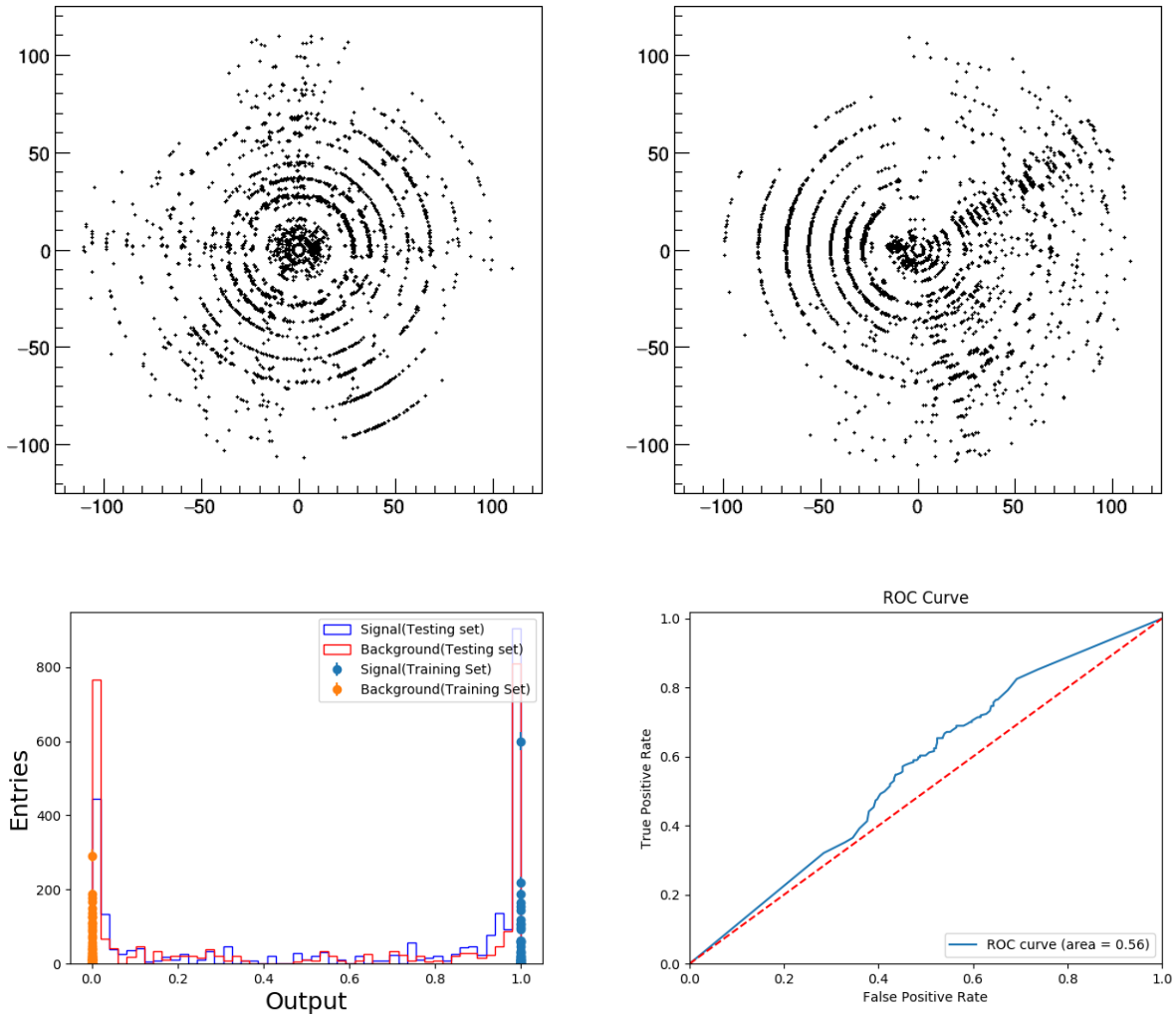

Figure 5.5: Plot on the left is of an event with all tracks having p_{T}less than 70 GeV and plot on the right is of event with one track having p_{T}more than 75 and all other tracks have p_{T}less than 70. Next row shows the result

corner is the plot of an event which has one track with p_{T}greater than 75 GeV and all other tracks have p_{T}less than 70 GeV an this is the signal class.

This was the brute-force attack. Just to check how exactly CNN is working for simulated data from CMS. As stats were low in first attempt, it was expected that the network will break-down and possibility of over training. From the class distribution histogram it is showing the over training. This attempt was just to see working of CNN.

5.2.1 Classification using number of Rec-hits

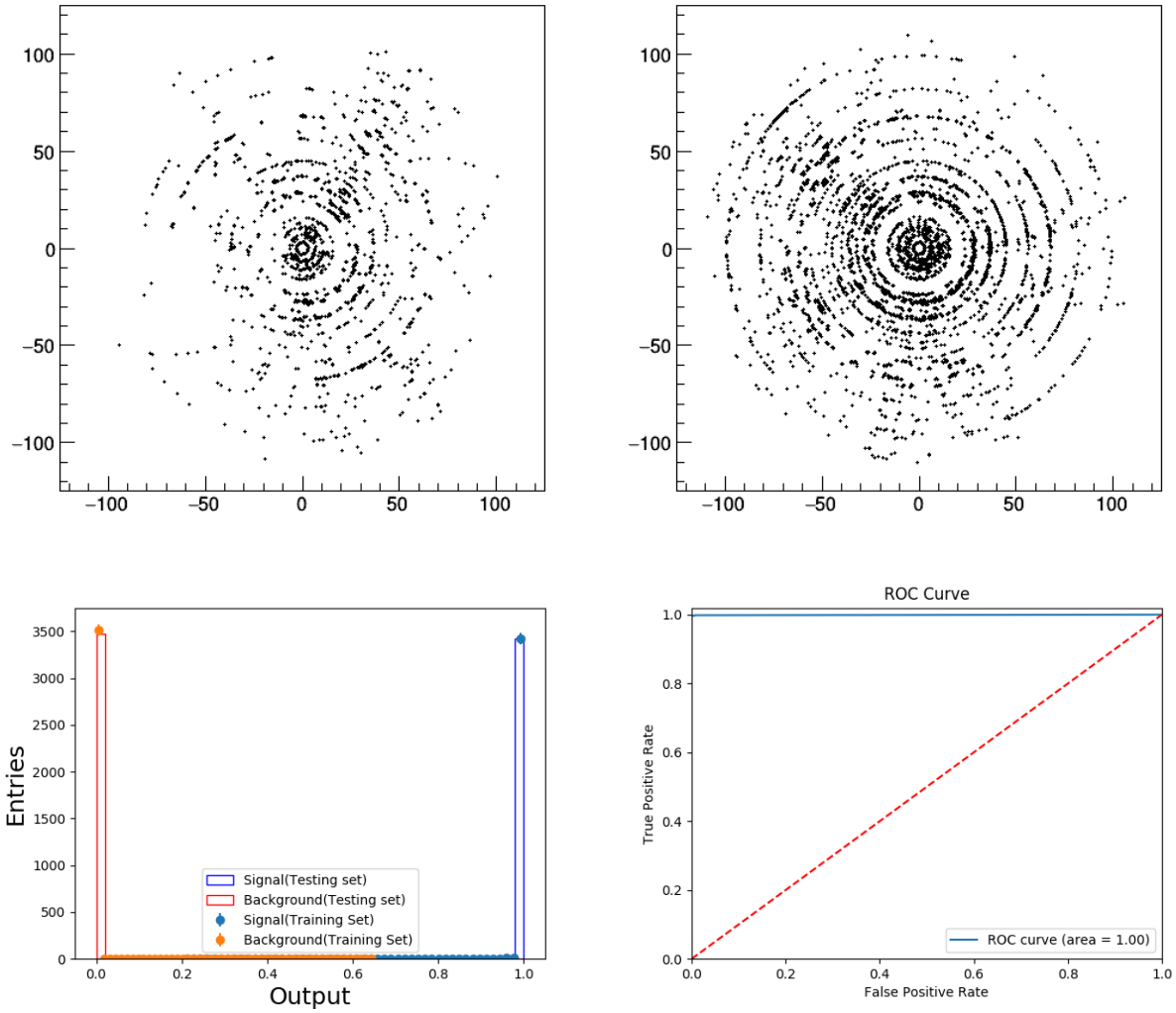

Figure 5.6: Plot on the left is of an event with number of rec-hits less than 2000. This is a background class. Plot on the right is of event with number of rechits more than 3500. This is signal class. CNN performance result has shown in next row.

In this exercise, referring to figure[5.3] background class consists of events with total number of rec-hits less than 2000 and signal class contains events with number of rec-hits greater than 3500. Both classes had 3500 images each. This part was done to see whether CNN is understanding the data or not. Classification is simple. CNN just has to classify dense images from less dense images with very high Accuracy. This was is shown from the ROC curve and class histogram plot. With 100% accuracy shown in figure[5.6]. This was

expected, CNN is classifying signal and background. This exercise gives confidence about CNN's process of reading the data.

Restricting p_{T} along with Rec-hits : Case 1

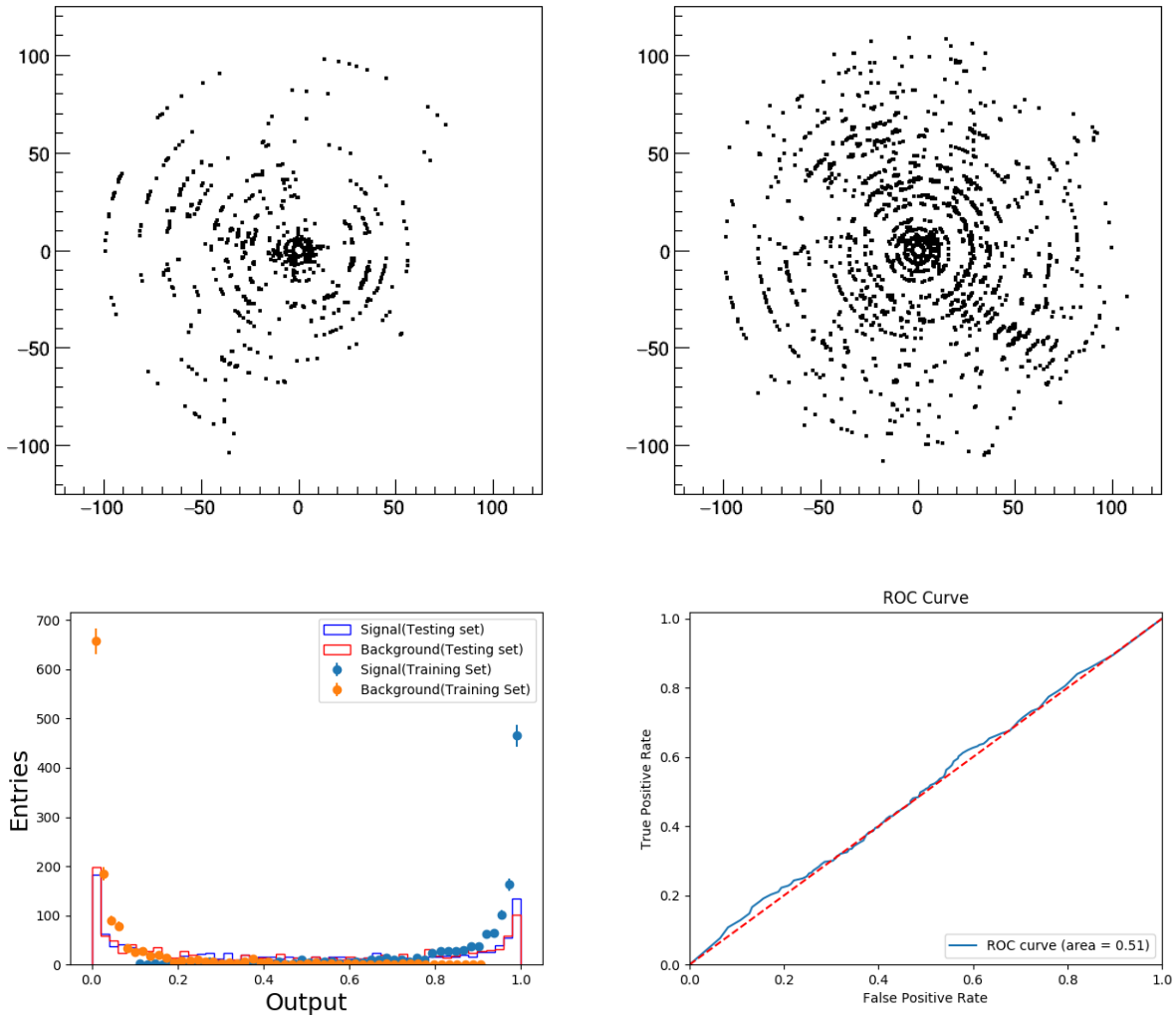

Figure 5.7: Plot on the left is of an event with number of rec-hits less than 2000 along with p_{T} of all tracks less than 70 GeV. This is background class. Plot on the right is of event with number of rechits more than 2000 along with p_{T} of one track more than 75 GeV and rest less than 70 GeV. This is the signal class. CNN performance result has shown in next row.

Motivation behind this exercise was to make image data less dense making feature of high

p_T more prominent. By doing this CNN may be able to understand the difference between two classes in testing phase.

Background class here is events with number of rec-hits less than 2000 along with p_T of all tracks less than 70 GeV. Signal class is events with number of rechits more than 3500 along with p_T of one track more than 75 GeV and rest less than 70 GeV. 3000 images were used for training.

Talking about CNN's performance, ROC curve just show 0.51 area under the curve which means CNN is not able to understand. This is also understood from class histogram plot.

Restricting p_T along with Rec-hits : Case 2

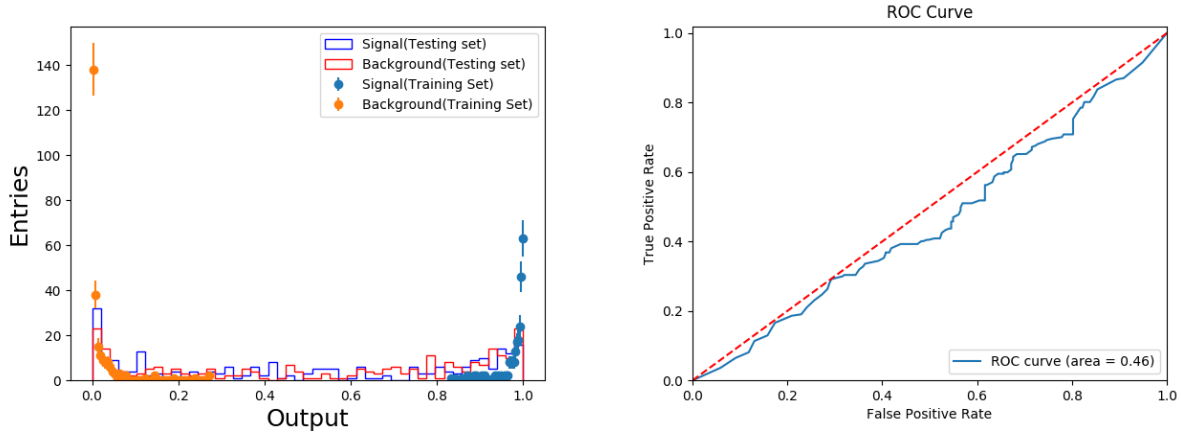

Figure 5.8: CNN performance has shown in this plots.

Applying logic of making images data less dense to make feature prominent, number of rec-hits were reduced to 1500. Event with number of rec-hits less than 1500 along with p_T of all tracks less than 70 GeV. This is background class. Event with number of rechits more than 1500 along with p_T of one track more than 75 GeV and rest less than 70 GeV. This is the signal class. CNN again has low accuracy for classification.

5.2.2 Dropping of Alternate Tracker Layers

Let's examine the performance of CNN by excluding alternate layers of tracker. again the Intuition behind this to make input image simplified enough so that CNN may learn the

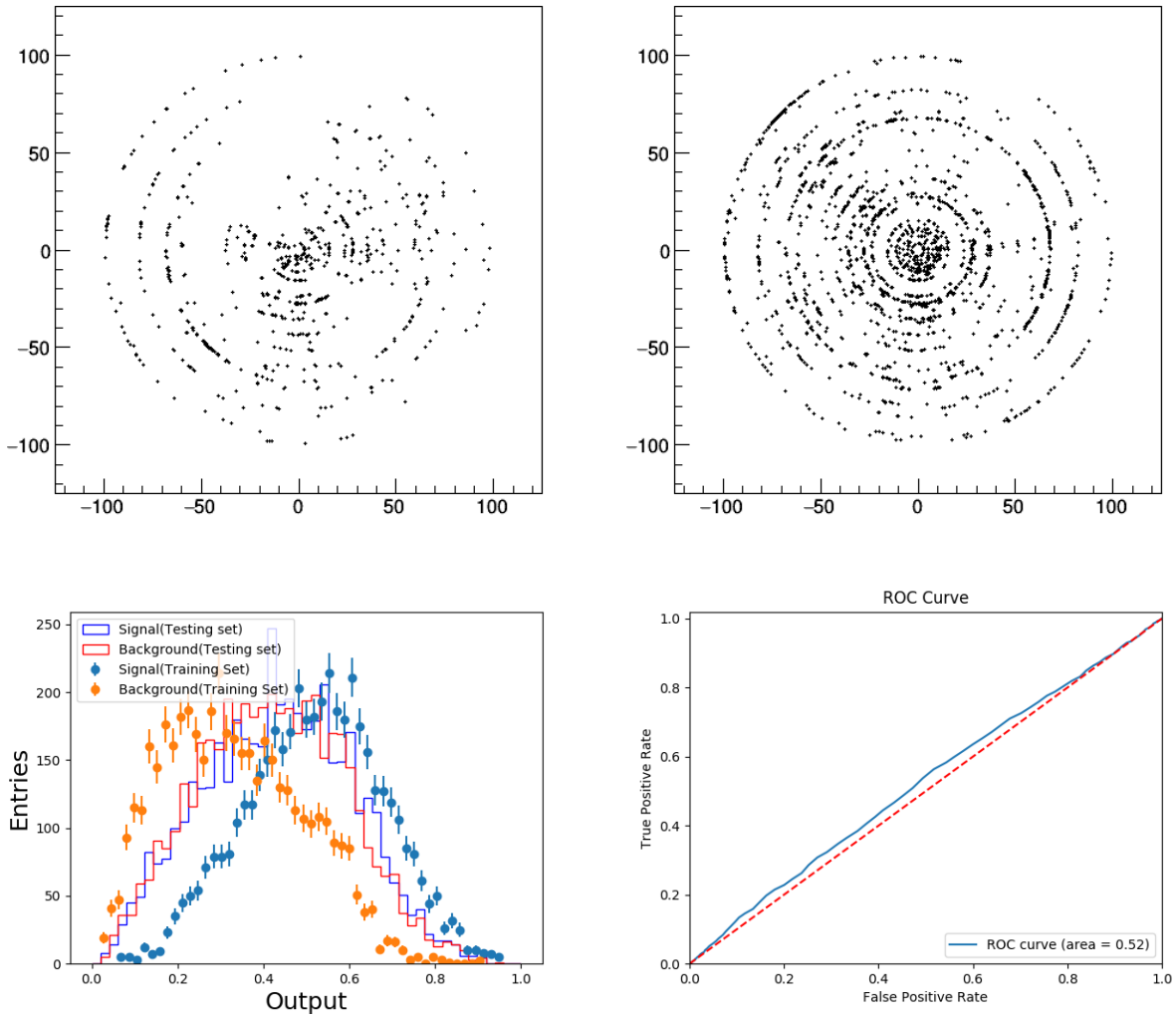

Figure 5.9: In this plot, even number of layers are dropped. Top left images is of background class and top right image is of background class. Signal class corresponds to event which has one track with p_{T} greater than 80 GeV and rest less than 30 GeV. Background class consists of tracks with p_{T} less than 30 GeV

feature.

ROC has 0.52 area under the curve. By looking at the class histogram, there is very little separation between classes while training as well as testing. CNN is performing poor. To see if making images less dense able to aid CNN's performance, same exercise had done by restricting the number of rec-hits, as it was done in previous section.

In figure[5.10], ROC again has 0.52 area under the curve. By looking at class distribution

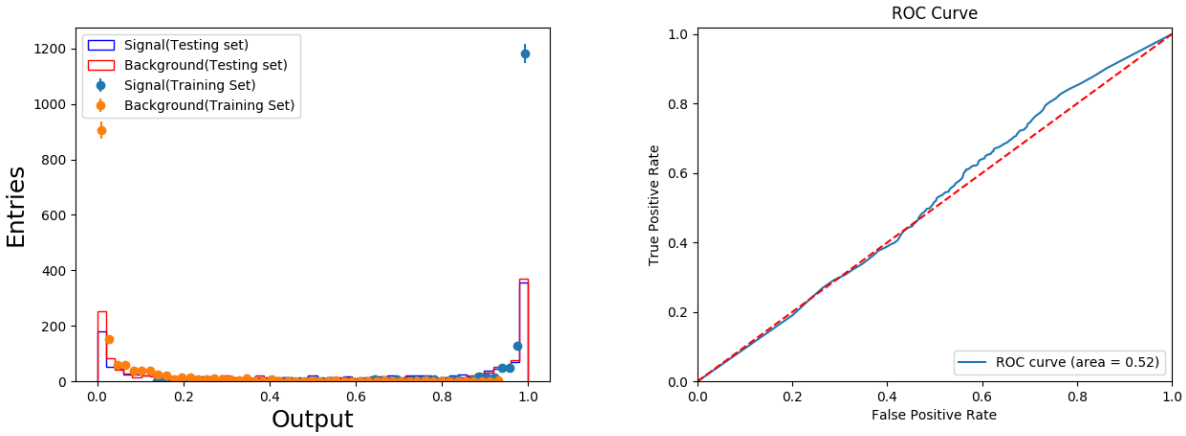

Figure 5.10: CNN performance has shown in this plots by restricting number of rec-hits histogram, CNN shows poor performance.

5.2.3 Systematic Dropping of Tracker Layers

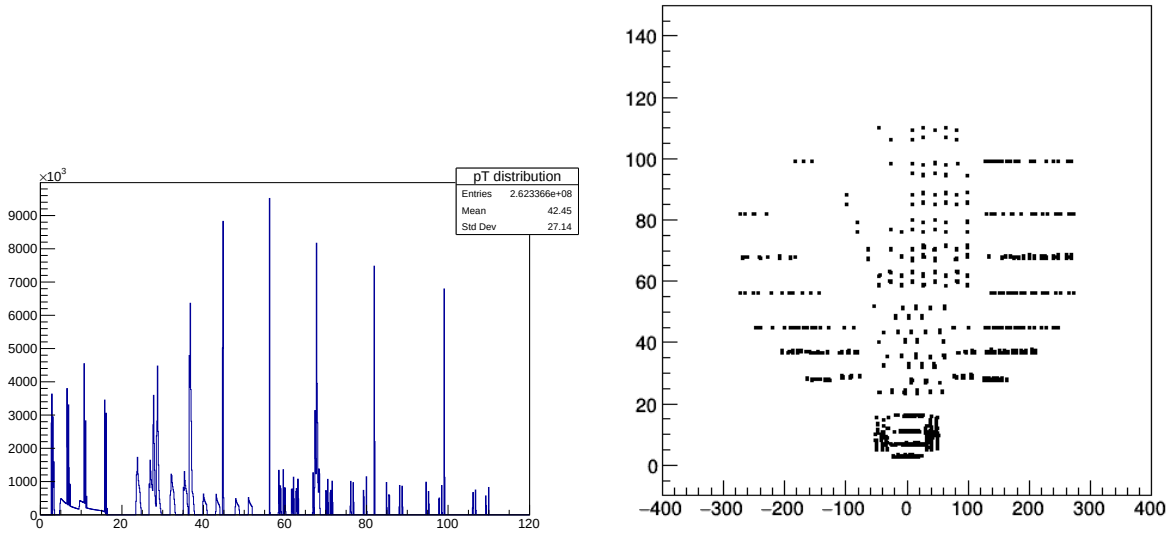

Figure 5.11: Plot on the Left shows the density of hits in each radial value. Plot on the right is the r-z plot. This plot shows the real picture of how hits are populating the tracker layers.

Figure[5.11] gives the information about population of hits in tracker layer. In the plot on the left, high peaks are between 40 to 80 cm. Advantage of this plot is that it is easy

to exclude layer, by taking radial value from the plot. The purpose of dropping layers systematically is if an tracker layer is very dense then it may not contribute to CNN's learning. High hit count in the layers may confuse CNN as CNN could take this high hit count as a feature. Although this high hit count in certain layer may present in both classes hence this high density of hits may act as a noise in the image data.

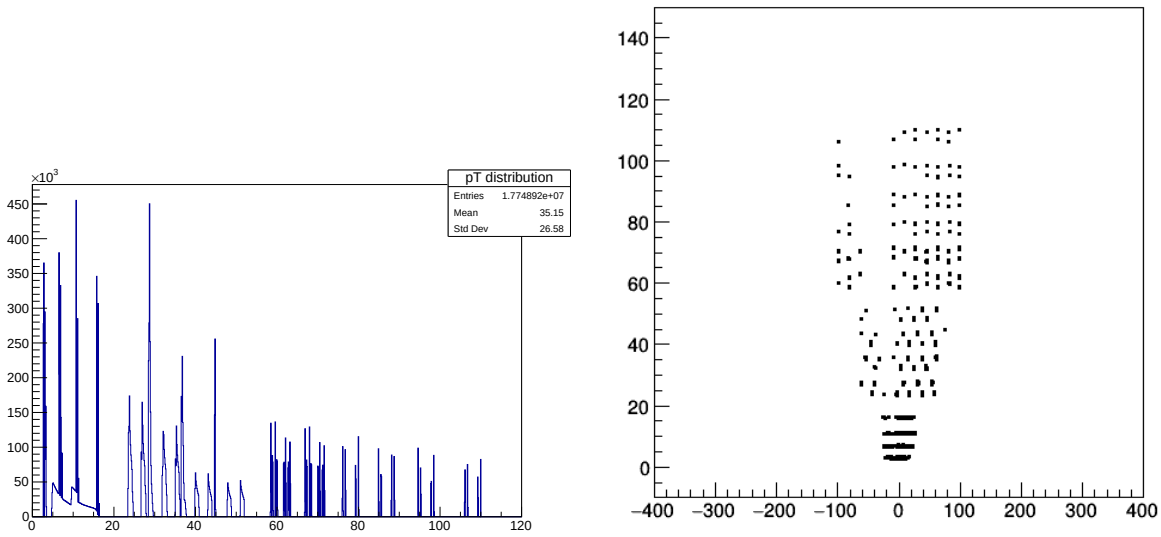

Figure 5.12: Plot on the right shows no high peaks as end-cap region is removed. Plot on the right also shows that end-caps as well as pixel discs are excluded by putting restriction on radius.

After observing data carefully, is was seen that those high peaks in figure[5.11] were from end-cap region. From the right plot in figure[5.11], end-cap region starts from 120 cm on both sides of z-coordinates. Taking this information, after putting restriction on z-coordinates, it was clear from figure[5.12] that those peak were indeed from end-caps region. This was shown in plot on the right on figure[5.12]. This plot also shows that pixel discs layers are also excluded. Motivation behind excluding pixel disc layers was to keep just cylindrical barrel layers as toy model only has cylindrical layers.

Application of CNN on r-z Restricted Data

Figure[5.13] shows images with restriction on r-z. Images look less crowded as compare to the figure[5.5]. Signal class corresponds to events which has one track of p_{T} 100 GeV and

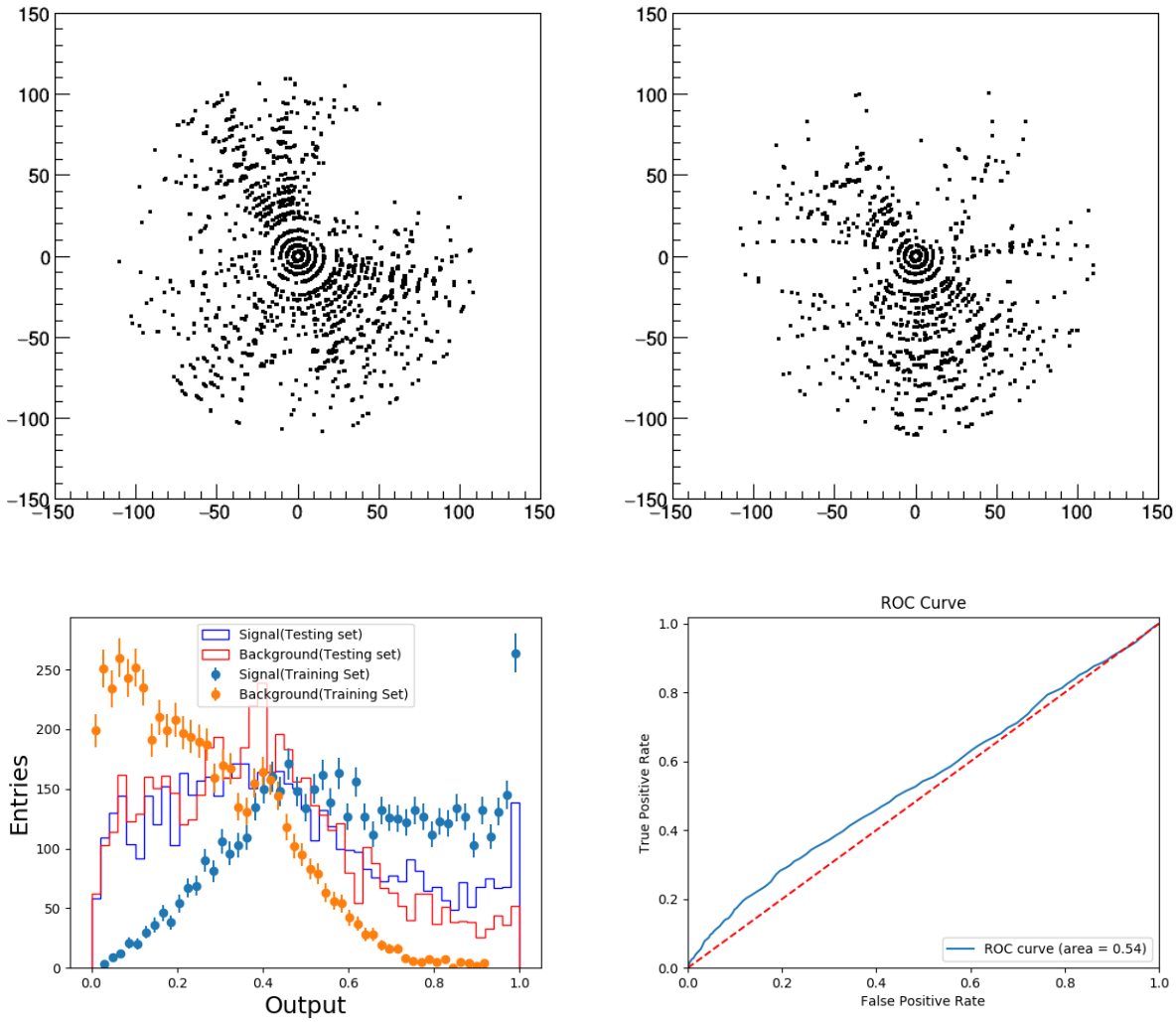

Figure 5.13: These are the plots of r-z restricted data. Top left images is of background class and top right image is of background class. Signal class corresponds to event which has one track with p_{T} greater than 100 GeV and rest less than 20 GeV. Background class consists of tracks with p_{T} less than 20 GeV

rest tracks have p_{T} less than 20 GeV. Whereas background class contain all tracks with p_{T} less than 20 GeV.

Relatively speaking CNN is kind of understanding the data now than other cases stated before but this performance far from what is expected. ROC currve has area of 0.54. This gives light to the direction of the work. As well as class distribution plot also started showing separation.

Dropping Inner Barrel Layers

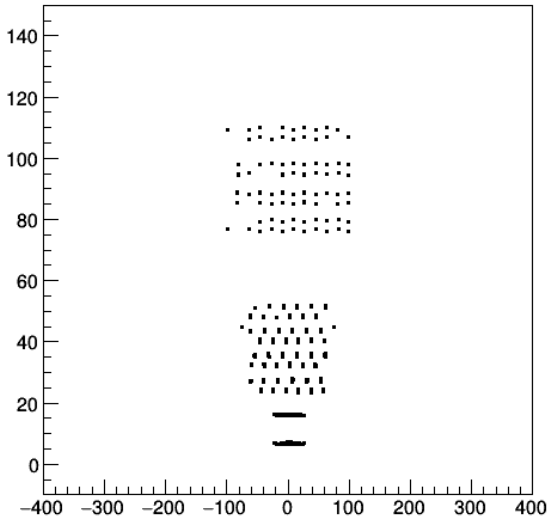

Figure 5.14: This is a r-z plot. Plot shows removal of inner barrel layers and first and third pixel cylindrical layer from data. Idea behind this removal was to make images less dense and easy to understand.

Removal end-cap region and pixel disc region have made it easy for CNN to learn, relatively speaking. This can be seen from ROC curve. Now to make images more simple, inner barrel region has dropped. This is seen in the figure[5.15] along with first and third pixel barrel layer. Following shows the performance of CNN on this data.

In figure[5.15], top right plot is of signal class. Signal class corresponds to events which has one track of p_{T} 100 GeV and rest tracks have p_{T} less than 20 GeV. Whereas background class contain all tracks with p_{T} less than 20 GeV.

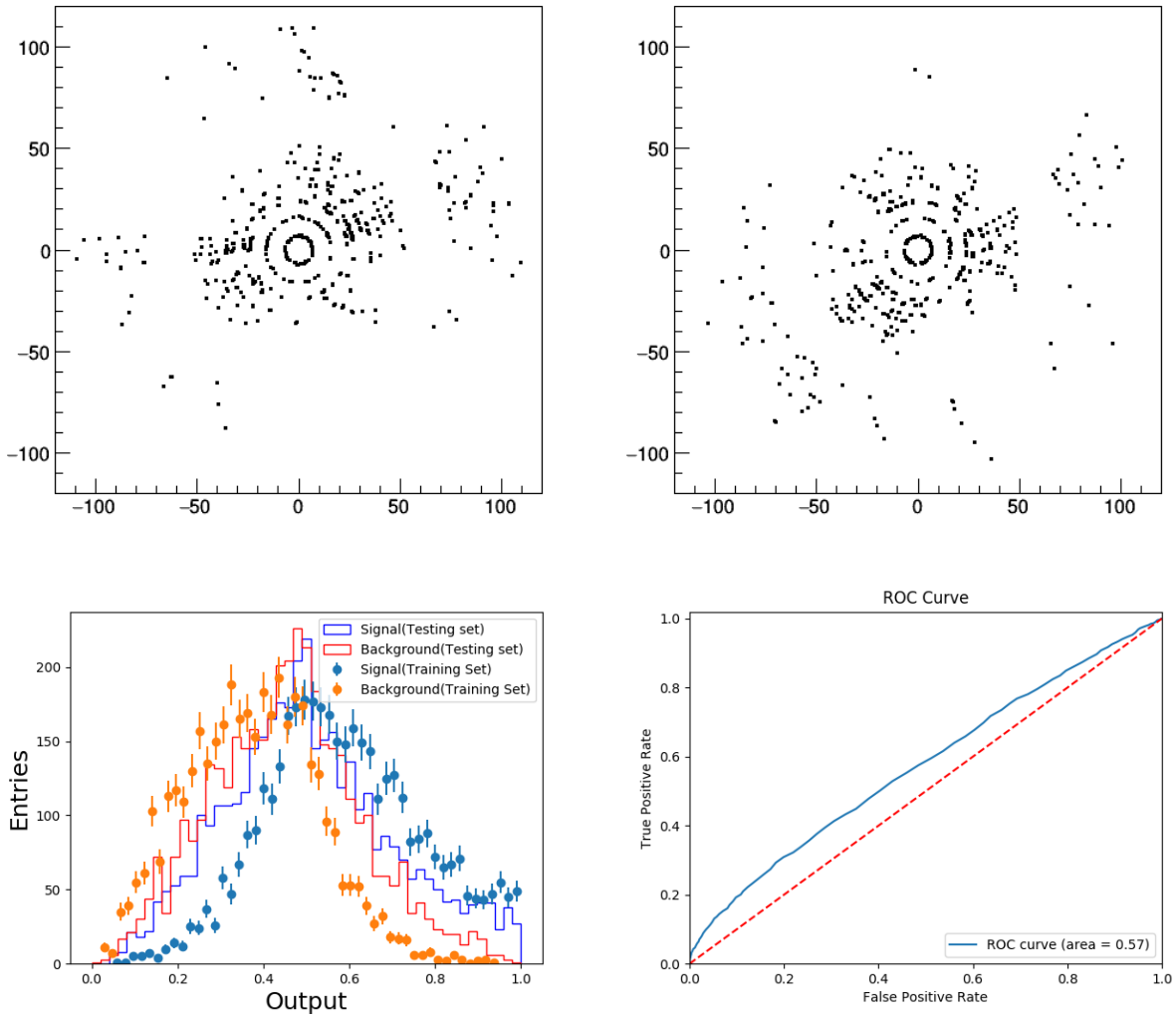

Figure 5.15: These are the plots of r-z restricted data along with inner barrel region and first and third pixel barrel layers exclusion. Top left images is of background class and top right image is of background class. Signal class corresponds to event which has one track with p_{T} greater than 100 GeV and rest less than 20 GeV. Background class consists of tracks with p_{T} less than 20 GeV

After removing end-caps, inner barrel region, and first and third pixel barrel region, images don't look crowded with data points. After putting these images through CNN, ROC gives area of 0.57. This is an improvement but again this value is far from expectation. Though, Class distribution plot does not show clear separation. More study is needed on this to achieve high accuracy of CNN.

5.2.4 Playing Around With Tracker Layers

This section studies about importance of pixel, TOB and TIB layers corresponding to CNN. Using figure[5.11], some layers are ignored. This section only has barrel layers to study only cylindrical part of tracker. Muon end-cap and double dics regions are not considered into this study.

Case 1 :

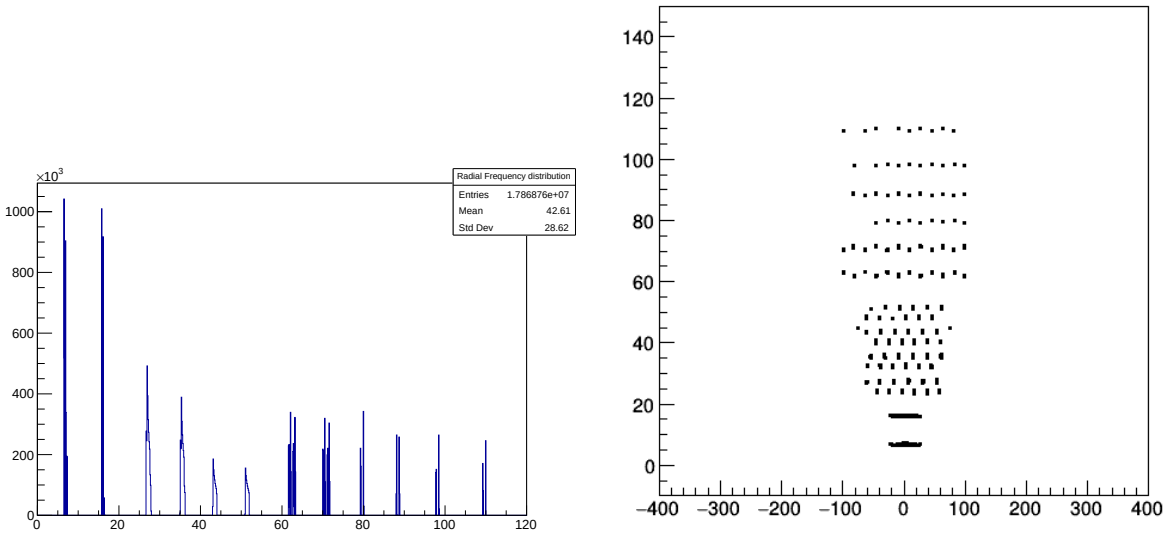

Figure 5.16: This is a r-z plot. Some layers from pixel, TOB, TIB sections are removed. Details of exclusion of layers have given in the table[5.1]

Figure[5.1] is a r-z plot. Some layers from pixel part, TIB and TOB part are not taken into consideration. Layers are excluded arbitrarily. In Table[5.2.4], exclusion values of z-coordinate and radius are given.

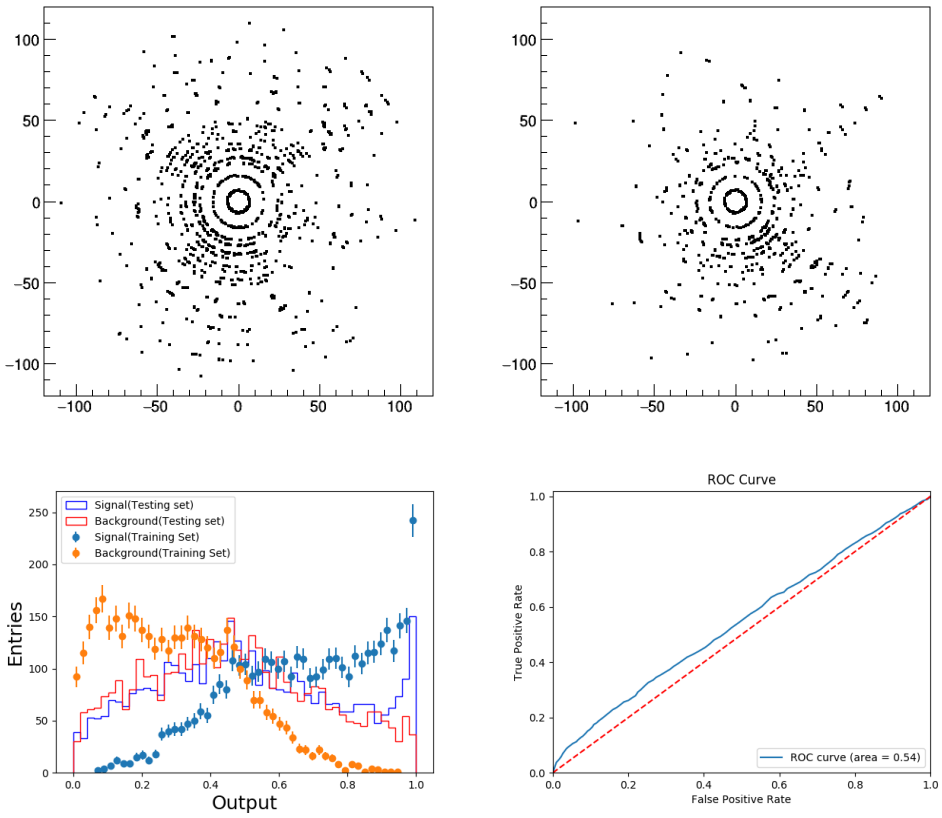

Figure 5.17: These are the plots of r-z restricted data. Top left images is of background class and top right image is of background class. Signal class corresponds to event which has one track with p_{T} greater than 100 GeV and rest less than 20 GeV. Background class consists of tracks with p_{T} less than 20 GeV

Shown in figure[5.17]. This is how image data looks when end-caps, inner barrel region, and first and third pixel barrel region are removed. After putting these images through CNN, ROC gives area of 0.54. Class distribution plot does show some separation for in training but still not separating testing classes. Let's see what happens if by excluding more tracker layers from data.

absolute z-coordinate	Excluded radial region in **open interval**
z>120	Every radial value
z<29	(0,20)
z<75	(22,52)
z<40	(7.5,11.5)
z<40	(2,4)
z<150	(58,61)
z<150	(66,69)
z<150	(75,78)
z<150	(84,86)
z<150	(94,96)
z<150	(105,108)

Table 5.1: Given the regions above in open intervals, these regions were excluded while plotting the data. Motivation behind this was to remove noise by reducing density so CNN may pick up the right feature.

Case 2 :

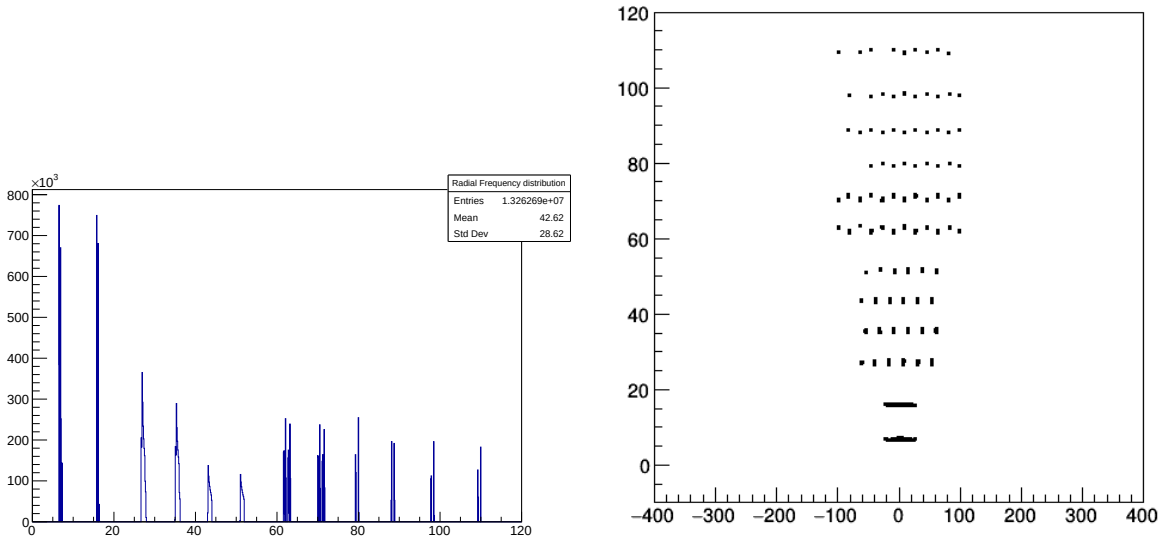

Figure 5.18: This is a r-z plot. Some layers from pixel, TOB, TIB sections are removed. Details of exclusion of layers have given in the table[5.2.4]

In this case, more layers from TIB section are excluded. It is shown in figure[5.19] also in table[5.2], exclusion ranges are given.

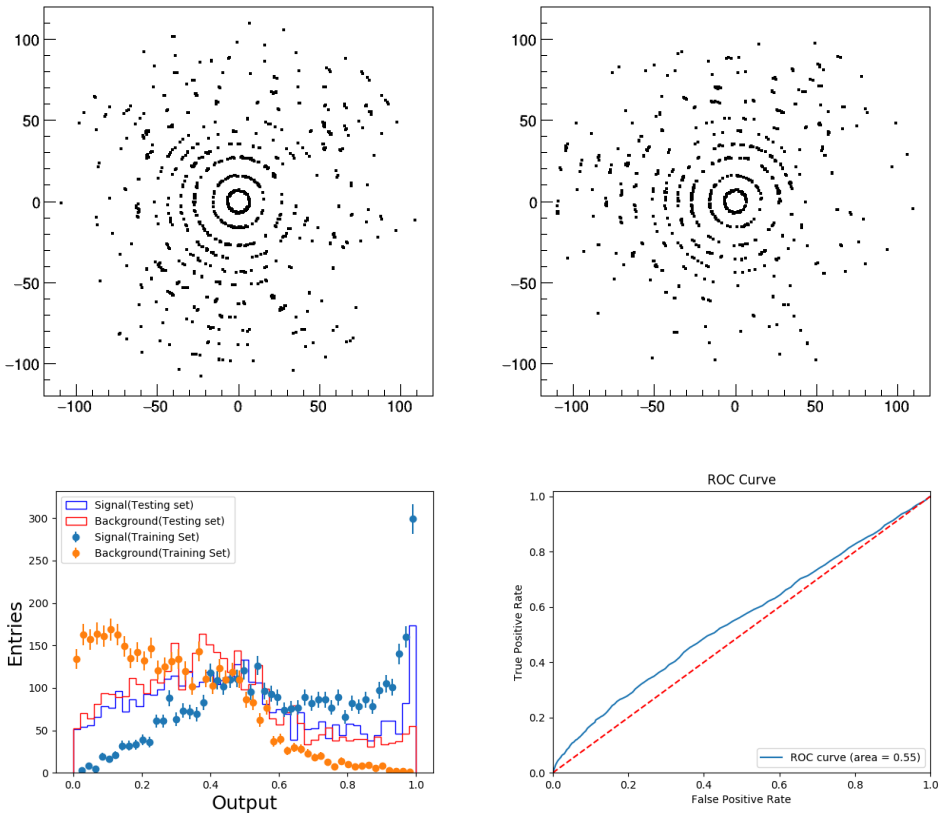

Figure 5.19: These are the plots of r-z restricted data. Top left images is of background class and top right image is of background class. Signal class corresponds to event which has one track with p_{T} greater than 100 GeV and rest less than 20 GeV. Background class consists of tracks with p_{T} less than 20 GeV

Shown in figure[5.17]. This is how image data looks when end-caps, inner barrel region, and first and third pixel barrel region are removed. In output of CNN, ROC gives area of 0.55. Again Class distribution plot does show some separation for in training but still not separating testing classes. Class distribution looks similar to case 1.

absolute z-coordinate	Excluded radial region in **open interval**
z>120	Every radial value
z<29	(0,20)
z<75	(22,52)
z<40	(7.5,11.5)
z<40	(2,4)
z<150	(23,25)
z<150	(30,34)
z<150	(39,42)
z<150	(44.5,45.3)
z<150	(47,50)
z<150	(58,61)
z<150	(66,69)
z<150	(75,78)
z<150	(84,86)
z<150	(94,96)
z<150	(105,108)

Table 5.2: Given the regions above in open intervals, these regions were excluded while plotting the data.

Case 3

In this case, whole pixel part is not into consideration. Table[5.3] summarises r and z values. It was just to see whether pixel part make any contribution or not as pixel region is mostly fully occupied. From the figure[5.20], it is non-conclusive as ROC curve still shows the area of 0.54. Important point here is class distribution. In class distribution, training ans testing classes curves are peaking at central value,signifying CNN has not understood the data of these two classes. It is treating them same.

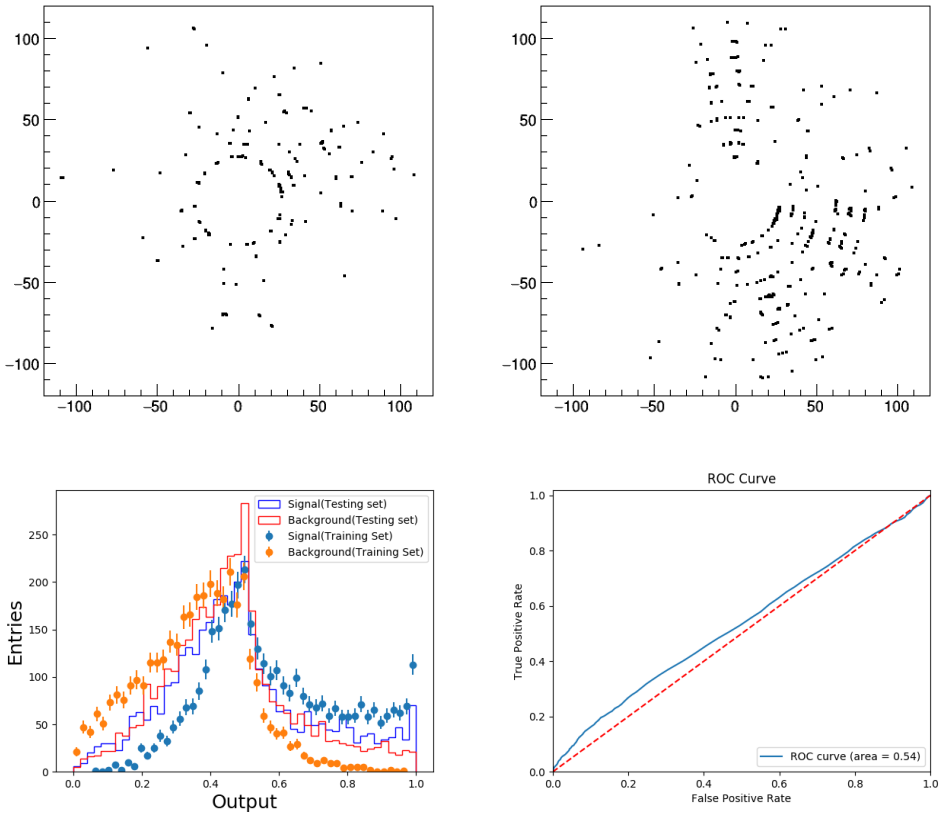

Figure 5.20: These are the plots of r-z restricted data. Top left images is of background class and top right image is of background class. Signal class corresponds to event which has one track with p_{T} greater than 100 GeV and rest less than 20 GeV. Background class consists of tracks with p_{T} less than 20 GeV

Absolute z-coordinate	Excluded radial region in **open interval**
z>120	Every radial value
z<29	(0,20)
z<75	(22,52)
z<40	(2,4)
z<40	(6,8)
z<40	(7.5,11.5)
z<40	(14,18)
z<150	(23,25)
z<150	(30,34)
z<150	(39,42)
z<150	(44.5,45.3)
z<150	(47,50)
z<150	(58,61)
z<150	(66,69)
z<150	(75,78)
z<150	(84,86)
z<150	(94,96)
z<150	(105,108)

Table 5.3: Given the regions above in open intervals, these regions were excluded while plotting the data.

Chapter 6

Conclusion and Discussion

6.1 Summary and Conclusion

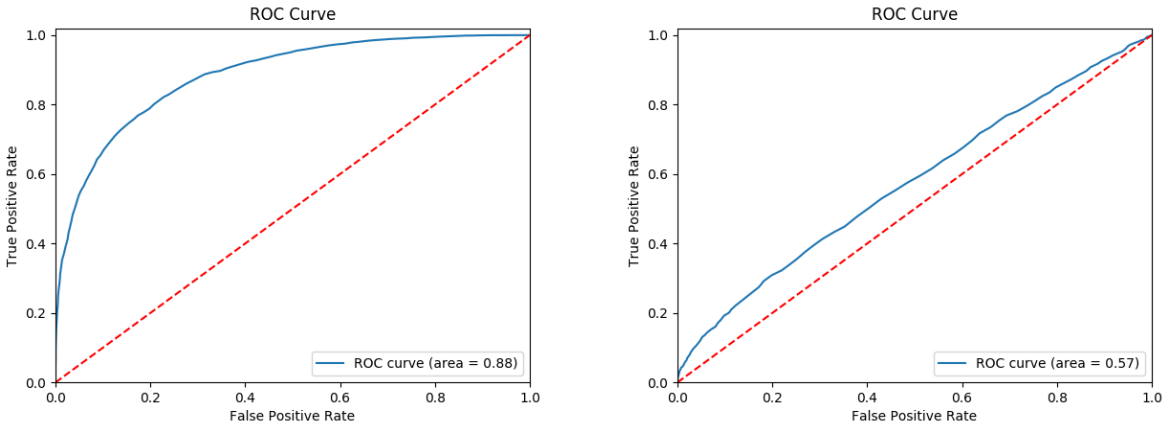

Figure 6.1: Plot on the left shows the accuracy of the CNN for highest complexity of toy model data whereas plot on the left shows the highest accuracy of the CNN for simulated data from CMS.

Figure[6.1] shows the contrasting behavior of CNN for toy tracker model and simulated data from CMS. ROC on the toy tracker model has area 0.88 where as for simulated data from CMS, ROC has area of 0.57. There are several reasons for this anomaly. In toy tracker model many parameters were under control, mainly number of tracks, p_{T} of tracks. These parameters stayed constant in signal and background classes. The case was different in

simulated data. Events were generated from $t\bar{t}$ process, there were no control over number of tracks and p_{T}. These parameters had random values in signal and background classes and these parameters acted as a noise in data.In toy tracker model it was easy to find the feature from data because there was no noise, that is why CNN was able to learn and perform classification, on the other hand, in simulated data CNN was learning many more features which were corresponding to both classes. This is why the loss of performance observe. There was another important factor in simulated data is stats. Data was populated with low energy particle, as expected but in 5000 events approximately 500 events had high energy particles. Data generation was happening on CMS cluster, there was a limit to how long one job should run. This took a lot of time. Accuracy may go up if large number of statistics made available.

On the other hand, usage of CNN at trigger levels is promising. Training used to take around 30 minutes on lab's GPU, referring to section[3.4.2]. This time will reduced to nothing as CERN has GPU farm. More data and new data harvesting techniques may boost CNN's performance making triggering system more faster.

Bibliography

[1] The CMS Collaboration :
Description and performance of track and primary-vertex reconstruction with the CMS tracker.

[2] https://cms.cern/detector/triggering-and-data-acquisition
The CMS trigger system : CMS Collaboration arXiv:1609.02366

[3] https://indico.cern.ch/event/317063/contributions/732633/attachments/609275/838401/Induction.T

[4] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/

[5] https://home.cern/science/accelerators/large-hadron-collider

[6] http://cms.web.cern.ch/news/tracker-detector

[7] Karimäki, V. The CMS tracker system projectTechnical Design Report. CERN/LHCC-98-6, CMS-TDR-005, 1997.

[8] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.

[9] Ashutosh V. Kotwal : A fast method for finding charged-particle trajectories using unsupervised machine learning embedded in field-programmable gate arrays arXiv:1910.14149 [physics.ins-det]

[10] Keiron O'Shea, Ryan Nash: An Introduction to Convolutional Neural Networks arXiv:1511.08458 [cs.NE]

[11] Dan Cireşan, Ueli Meier, Juergen Schmidhuber: Multi-column Deep Neural Networks for Image Classification arXiv:1202.2745 [cs.CV]

[12] Patrick T. Komiske, a Eric M. Metodiev, a and Matthew D. Schwartz b :
Deep learning in color: towards automated quark/gluon jet discrimination.

[13] Dmitriy Baranov 1 , Sergey Mitsyn 1 , Pavel Goncharov 2,* and Gennady Ososkov 1 :
The particle track reconstruction based on neural networks

[14] Aristeidis Tsaris 1,a , Dustin Anderson 3 , Josh Bendavid 3 , Paolo Calafiura 2 , Giuseppe Cerati 1 , Julien Esseiva 2,4 , Steven Farrell 2 , Lindsey Gray 1 , Keshav Kapoor 1 , Jim Kowalkowski 1 , Mayur Mudigonda 2 , Prabhat 2 , Panagiotis Spentzouris 1 , Maria Spiropoulou 3 , Jean-Roch Vlimant 3 , Stephan Zheng 3 , Daniel Zurawski 1 :
The HEP.TrkX Project: Deep Learning for Particle Tracking

[15] Dan Guest, 1 Kyle Cranmer, 2 and Daniel Whiteson 1 :
Deep Learning and Its Application to LHC Physics

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[17] https://www.omnisci.com/technical-glossary/cpu-vs-gpu