# Continuum Suppression in Belle II

**Jagmeet Singh**

*A dissertation submitted for the partial fulfilment of*

*BS-MS dual degree in Science*

**Indian Institute of Science Education and Research Mohali**

**April 2019**

*Dedicated to my Family*

# Certificate of Examination

This is to certify that the dissertation titled **Continuum Suppression in Belle II** submitted by **Jagmeet Singh** (Reg. No. MS14006) for the partial fulfillment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Prof. Jasjeet Singh Bagla        Dr. Anosh Joseph        Dr. Vishal Bhardwaj
(Supervisor)

Dated: April 24, 2019

# Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Vishal Bhardwaj at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Jagmeet Singh
(Candidate)

Dated: April 24, 2019

In my capacity as the supervisor of the candidates project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Vishal Bhardwaj
(Supervisor)

# Acknowledgment

Firstly, I would like to express my sincere gratitude towards my supervisor Dr. Vishal Bhardwaj for his kind support, guidance, motivation and the immense knowledge. Without his help and guidance, the project would not have been possible. I am very thankful to him for giving me an opportunity to work under his guidance.

Besides my advisor, I would like to thank the rest of my thesis committee members Prof. Jasjeet Singh Bagla, and Dr. Anosh Joseph for their insightful comments and encouragement.

I sincerely thank Belle II Collaboration for providing me necessary computing tools and resources.

I would like to thank all my group members (Rajesh, Sourav, Renu) for guidance and maintaining the cheerful and peaceful atmosphere in the group. I would also like to thank my friends Keshav, Virinder, Pawan, Aman, Sanjay, Sukhwinder, Ashish, Jaskaran, Gurdeep who always supported and motivated me.

I acknowledge IISER Mohali for providing me best Infrastructures and environment for carrying out this project. I am also thankful to the "Department of Science and Technology," Govt. of India for supporting me with institute fellowship during the past five years.

To conclude above all, it was almighty and my family which have been there all these years strongly supporting and guiding me in life.

# List of Figures

# List of Tables

# Notation and Abbreviations

| | |
|---|---|
| $\theta_{i,j}$ | Weight matrix |
| $b_j$ | Biased term for each layer |
| $F_i$ | Activation function of $i^{th}$ layer |
| ROE | Rest of Events |
| MVA | Multivariate Analysis |
| BDT | Boosted Decision Tree |
| DNN | Deep Neural Network |
| TMVA | Toolkit for Multivariate Analysis |
| ROC | Receiver Operating Curve |
| GPU | Graphical Processing Unit |

# Contents

# Abstract

$B$-factories (Belle I/II and Babar) are experiments that study the $B$ mesons decay in electron-positron annihilation at the energy of the $\Upsilon$(4S) resonance (10.58 GeV). Belle II is a successor of the Belle experiment and is expected to collect 50 times more data than Belle. Besides the desired collision containing a $B$ meson pair (signal), Belle II also generates other events $(u, d, s, c)$. These *udsc* events (known as continuum events) acts as background if one is studying $B$ decays. If one has to search for rare decays such as $B \to K^* \mu^- e^-$ (Lepton Flavor Violation) in search for New Physics, continuum needs to be suppressed in order to achieve sufficient sensitivity. Multivariate analysis methods are used to separate the signal and continuum events. Machine learning algorithms particularly neural networks have been quite successful in various classification problems. In this thesis, different machine learning algorithms are tested and compared for the continuum suppression using various libraries.

# Chapter 1

# Introduction

The Standard Model (SM) describes the basic building blocks of the universe called fundamental particles and explains about the three of the four fundamental forces which govern them. Emerged in the 1970s, it has revealed almost all the phenomenon with outstanding accuracy. However, the SM has limitation. Moreover, to answer the existence of dark matter and other phenomena in astrophysics and cosmology, there is a need for physics beyond the Standard Model.

The collider experiments like Belle and BaBar, in which $B$ mesons are produced and detected are termed as $B$-factories. The experiments involve the collision of electron-positron particles at the center of mass energy of $\Upsilon$(4S) resonance peak which decays to form two $B$-mesons. Thus, new rare decays can be analyzed which can give hint of new physics beyond the Standard Model. The first large signals of CP violation were observed by $B$-factories and thus supported the theory [Kobayashi 73] of *Makoto Kobayashi* and *Toshihide Maskawa*. They were awarded the Nobel Prize in Physics in the year 2008. One of the origins of the dominance of matter over anti-matter is assumed of CP violation.

The CP violation measured by the experiments was not sufficient to explain the defined asymmetry. Therefore, a more profound understanding is required for the phenomenon. Belle II is an upgradation of Belle experiment in which the instantaneous luminosity will increase by a factor of 40 to get luminosity of $8 \times 10^{35}\text{cm}^{-2}s^{-1}$ [Aushev 10]. It will collect 50 times more data than the Belle experiment and $B$-mesons decays can be analyzed more accurately. Apart from $B$ mesons formation after annihilation, many other events are also generated which do not contain the $B$

meson pair, thus do not contribute for the analysis part and are termed as continuum events. These events need to be removed or suppressed from the data to ensure the analysis of the proper data set.

In this thesis, the problem of 'continuum suppression' is analyzed using machine learning algorithms and frameworks. The Belle II data is simulated and reconstructed for signal and continuum events. The primary focus is to apply the neural network deep learning techniques using various open source libraries and to get an optimal model for the separation of continuum and signal. Some new features are added, and their importance is also measured to suppress the background.

Chapter 2 provides a brief introduction to the Belle II experiment. Different detectors, their importance and the software supplied by KEK are discussed. Chapter 3 gives an overview of the continuum suppression and the features that are being used to analyze this problem. Different multivariate analysis methods and their algorithms are discussed in Chapter 4. The implementation of the neural network for classification problem and its parameters are also discussed. Chapter 5 emphasis on the data that has been used for continuum suppression and the ways by which it has been simulated and reconstructed. The results using different methods and frameworks are given in Chapter 6.

# Chapter 2

# The Belle II Experiment

Belle II experiment is an upgradation of Belle experiment to study the decay of $B$ mesons formed by the collision of electron and positron at the center of mass energy of $\Upsilon - [4S]$. It is located in Tsukuba, Japan at the High Energy Accelerator Research Organisation (KEK) and is designed to have the highest luminosity in the world. The first collision took place on 26 April 2018 and events are recorded from the electron-positron collision. The design and build team of Belle II detector at SuperKEKB comprises of the international collaboration of about 750 researchers from 25 countries. The experiment will accumulate data for around ten years and about 50 billion $B\overline{B}$ meson pair production is expected within this period. The experiment aims to search the physics beyond Standard Model in rare decays and also to examine the exotic states.

This Chapter gives a brief introduction of the Belle II accelerator and detectors, Section 2.1 describes the accelerator at SuperKEKB. Section 2.3 provides an overview of basf2 software which is provided by KEK server for the computation and also about gbasf2 which provides an interface between the local server and GRID server.

## 2.1 Accelerator

The accelerator design at SuperKEKB is “nano-beam” [Kurokawa 03] scheme which was high current version in Belle. The cross section of the beam is decreased and thus increasing the luminosity by a factor of 20. The number of particles per beam is also increased to get an overall luminosity rise by a factor of 40. The same tunnel is used as that of KEKB. The beam energies are 4.0 GeV in the low energy ring (LER) and 7.0 GeV in the high energy ring (HER) for the positron and electron particles respectively. The collision takes place at 10.58 GeV which is center of mass energy of $\Upsilon - [4S]$. More dipole, quadrupole, and sextupole magnets are added to the accelerator.

## 2.2 Detectors

The Belle II detector is given in the Figure 2.1 with various detectors within it. A brief introduction of the different detectors is given in the following sections.

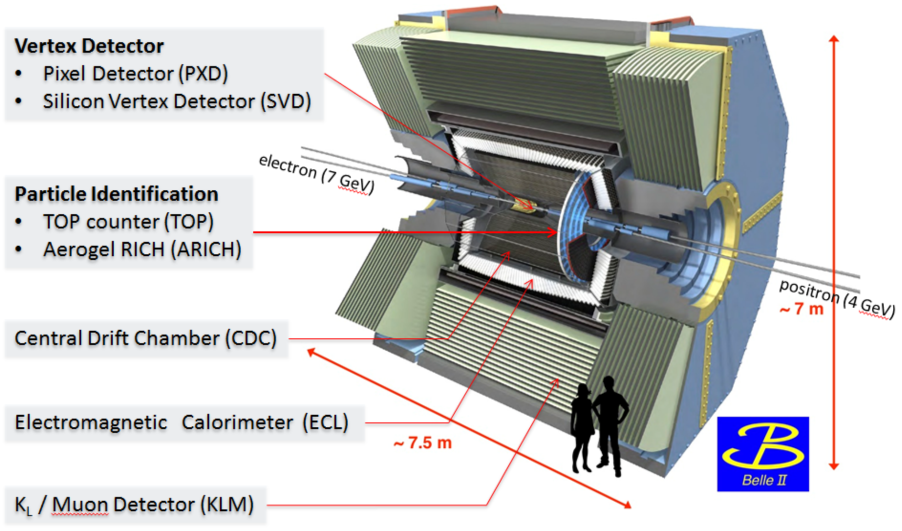


Figure 2.1: Belle II detector

### 2.2.1 Vertex Detector (VXD)

Vertex detectors consists of two detectors, one is silicon pixel detector (PXD), and the other is silicon strip detector (SVD) [Adachi 18]. The main purpose of the vertex detectors is tracking of the particles.

The PXD detector is two layers with layer 1 (L1) at radius 14 mm and layer 2 (L2) with radius 22mm. It is based on Depleted p-channel Field Effect Transistor (DEPFET) technology. The SVD detector is rectangular and trapezoidal and constitutes the four outer layers of VXD. It has four layers with double-sided silicon microstrip detectors (DSSDs). Thus the VXD has total of six layers around the beam pipe. The total power consumption of the vertex detector is very high, and it is supported with proper cooling channels using $-20°$ liquid $CO_2$.

### 2.2.2 Central Drift Chamber (CDC)

The purpose of Central Drift Chamber (CDC) is to reconstruct the trajectories of charged particles and thus to determine the momenta and particle identification in the low momentum region [Adachi 18]. It consists of a large volume gas drift chamber filled with Helium-Methane mixture with drift cells. The CDC cell is 1200 mm wide and 250 mm thick and consists of a lot of anode and cathode wires.

### 2.2.3 Particle Identification system (TOP and ARICH)

The time-of-propagation (TOP) counter is used for particle identification in the outer region of the CDC, and it uses the Cerenkov principle for the identification. This detector provides a separation between kaons and pions. In the end-cap region, a Cherenkov imaging detector ARICH with aerosol as a radiator is used to identify charged particles [Adachi 18].

### 2.2.4 Electromagnetic Calorimeter (ECL)

The purpose of electromagnetic calorimeter (ECL) is to detect $\gamma$ rays (photons) along with identification of electrons. It consists of a heavily-segmented array of Thallium-doped Cesium Iodide CsI(Tl) crystals and is located at the top end region and barrel region.

### 2.2.5 $K_L$ and Muon Detector (KLM)

The $K_L$ and muon detector (KLM) is located outside the superconducting solenoid and consists of an alternating sandwich of 4.7 cm thick iron plates and detector layers. It is used to provide a separation between hadrons and muons.

### 2.2.6 Other Components

The axial magnetic field is 1.5 T and is provided by superconducting solenoid [Adachi 18]. The trigger system rejects the large background events coming from beam scattering and Bhabha scattering. The data is transferred from the detector using the data acquisition system (DAQ). FPGA's and high-speed fiber optics are used to transfer the data, and the final data is then moved to the KEK computing center (KEKCC) and GRID for processing.

## 2.3 basf2 and gbasf2

basf2 is a software written for Belle II experiment analysis [Kuhr 18] . It replaces the old basf which is provided for the computation in Belle experiment. The event generation, simulation, and reconstruction are done using this software package, and different modules are available to do these tasks easily. It comes up with a python interface for users. The software is provided on the KEK computing work server. Different releases are available and are updated from time to time. It also allows submitting jobs at the Belle cluster where multiple jobs can be submitted at the same time.

gbasf2 provides an interface between local work server and the GRID server. It is designed as an extension of basf2 to distributed computing system [Miyake 15]. The grid server is used for reconstruction of background events. The grid data is accessed by giving the unique guid of each root file available at the grid. It includes many sites and jobs are submitted by setting up gbasf2. It uses the DIRAC Distributed Computing Framework to submit jobs to the best available site. It takes the same steering file as used by basf2. Jobs can be submitted by giving a unique job name, the release to be used and a set of parameters.

# Chapter 3

# Continuum Suppression

Each data analysis problem should be done on the proper data and it becomes more important when dealing with huge amount of data coming from experiments like Belle II. Data that do not contribute to the analysis needs to be removed. To check whether a data is important or not, different input features and methods are used. This chapter provides an overview of background data for the Belle II analysis and also the corresponding features which separate them from the signal data.

## 3.1 Continuum

The $e^+e^-$ collision at SuperKEK takes place at the energy of 10.58 GeV which corresponds to the invariant mass of $\Upsilon$-[4S]. The signal events at Belle are the events which result in the formation of $B$ meson pair on each collision. This is the desired event for a collision so that $B$ meson decay can be analyzed. Apart from the desired event, the $e^+e^-$ collision results in many other events that do not result in $B$ meson pair formation. These events are termed as continuum or background events. The continuum events need to be suppressed/removed for further analysis [Weyland 02].

The major contribution of background events comes from the events that leads to formation of $u\overline{u}$, $c\overline{c}$, $s\overline{s}$ and $d\overline{d}$. The relative cross-section for different events is given in Figure 3.1. The signal events occurs only around 22.2%, thus it becomes very significant to remove the continuum events.

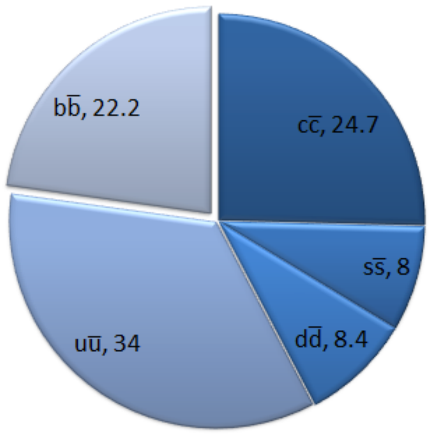


Figure 3.1: Relative cross-section of different events in $e^+e^-$ collision at $\Upsilon$-(4S)

## 3.2 Event Topology

The event topology for signal and continuum events is given in Figure 3.2. For signal events, the $B$ mesons have 0 spins and in the center of the mass frame, they decay almost isotropically in all directions giving a spherical shape for the decay products. For the background events, the light quark particles have high kinetic energy because of which the decayed particles do not deviate much from the beam axis and thus giving a jet-like structure for the final particles [Fox 78].

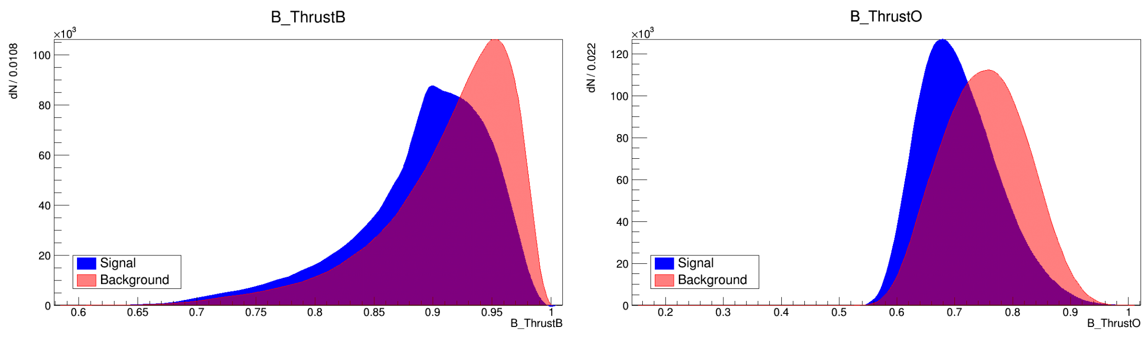
(a) Continuum events (b) Signal events

Figure 3.2: Event topology (Taken from [Li 15])

Each signal event results in the formation of an entangled pair of $B$ mesons. The final decayed particles detected at the detector are used to calculate different feature of an event which is given in the sections below. The particles are combined to get the $B$ meson which corresponds to one of the pair while the rest of tracks (other $B$) are termed Rest of Events (ROE).

### 3.2.1 Thrust Features

The axis that maximizes the sum of the longitudinal momenta of particles is defined as the thrust axis. For each event two thrust axis are defined, one for the reconstructed $B$ meson and other for the Rest of Events (ROE). Two different thrust feature is used which corresponds to the magnitude of the thrust axis. The magnitude of the thrust axis is higher for the background events because of their high kinetic energy while decaying. The distributions of thrust axis magnitude for signal and background events are given in Figure 3.3.



(a) $B$ thrust axis

(b) Other $B$ (ROE) thrust axis

Figure 3.3: Distributions of the magnitude of thrust axis

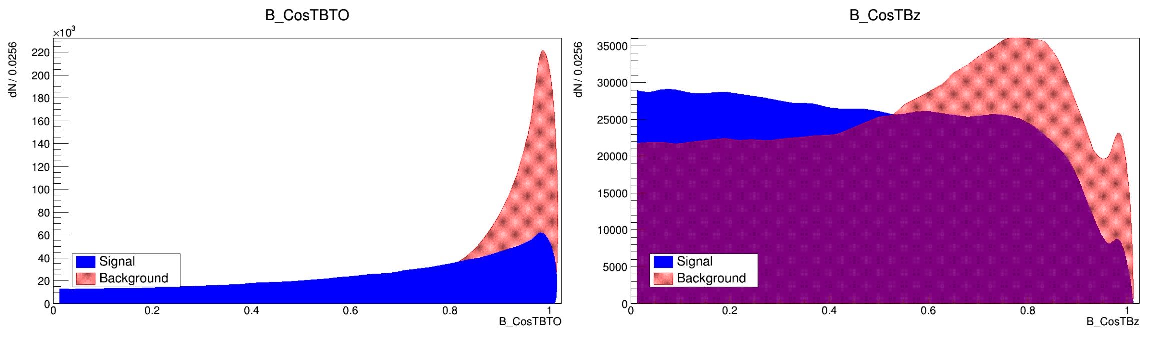
### 3.2.2 Angular Features

The two angular features which differentiate signal and background events are the:

1. $\cos(TBTO)$: Cosine of the angle between thrust axis of $B$ and thrust axis of the Rest of Events (ROE).

2. $\cos(TBz)$: Cosine of the angle between the thrust axis of $B$ and the z-axis.

The distribution plots for both the angular features are given in Figure 3.4. The $\cos(TBTO)$ for the background events shows a peak near 1.0 showing the angle between the $B$ and ROEs is almost 180° while the other angles correspond to the events in which the products do not decay in the direction of the beam axis. The $\cos(TBTO)$ distribution for the signal events, is almost equal for all the angles showing the isotropic nature of the events. The $\cos(TBz)$ for the background events show a peak at near about 1.0 which corresponds to the final products in the direction of the beam axis. The jet formation is not always in the beam axis that is why there are contributions from other angles also in the distribution plot. The signal distribution plot shows the isotropic nature of the decay with almost equal contribution from all the angles.

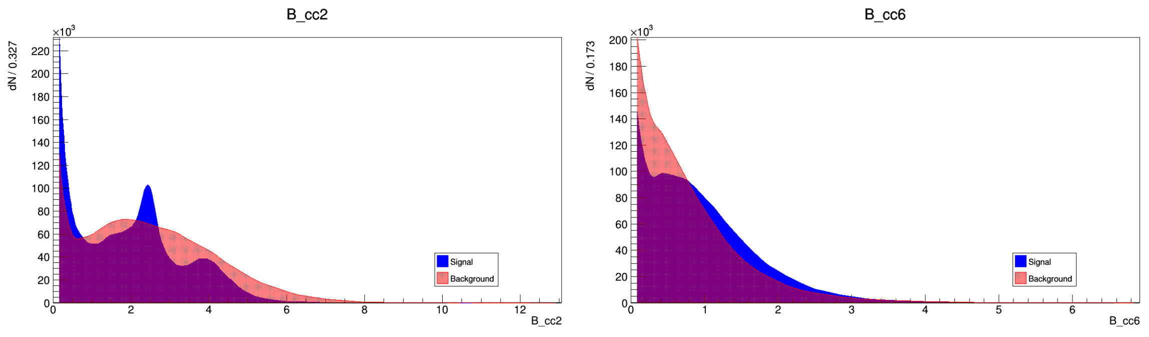


(a) $\cos(TBTO)$ (b) $\cos(TBz)$

Figure 3.4: Distributions of the angular features

### 3.2.3 Cleo Cones

Cleo clones are the cones separated by the angular intervals of 10° and placed concentrically around the thrust axis. Each cone measures the scalar momentum flow around the thrust axis. Total 9 Cleo cones features are defined, and two of the distributions are given in Figure 3.5.



(a) Cleo cone 2 (b) Cleo cone 6

Figure 3.5: Distributions of the Cleo cones with largest separation

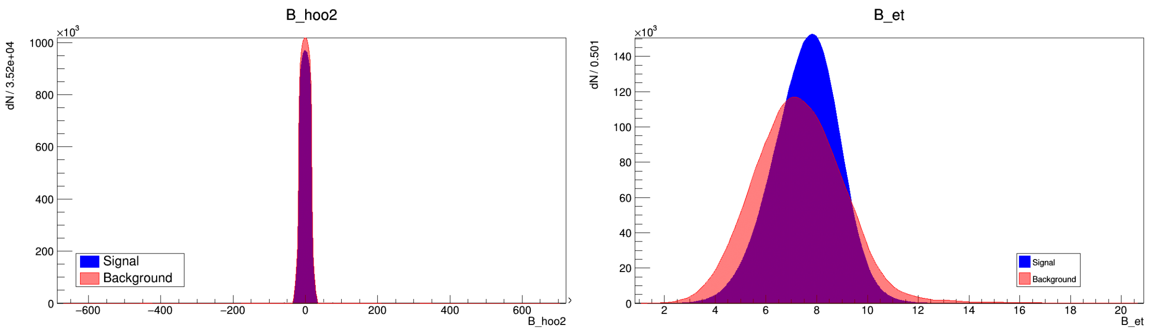
### 3.2.4 Fox-Wolfram Moments

The Fox-Wolfram moments are given as:

$$H_l = \sum_{i,j=1}^{N} |p_i||p_j|P_l(cos(\theta_{ij}))$$

where $i, j$ are the charged and $\gamma$ particles, $p_i, p_j$ are the momentum of the particles, $P_l$ is Legendre's Polynomial and $\theta_{i,j}$ is the angle between particles $i$ and $j$.

The Kakuno-Super-Fox-Wolfram (KSFW) moments are the refined version of the FW moments which are defined in the Ref[Aushev 10]. The missing mass square $M_m^2$ and the transverse energy $E_t$, which is defined as the sum of the transverse momenta of the particles are also considered in KSFW moments. Thus the complete features for the KSFW moments are 18 and are very good discriminant for the separation of signal and background. Some of the distributions for the KSFW moments are given in Figure 3.6.

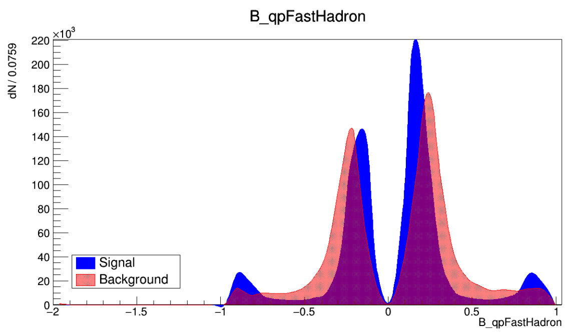


(a) KSFW hoo2 moment (b) Transverse energy

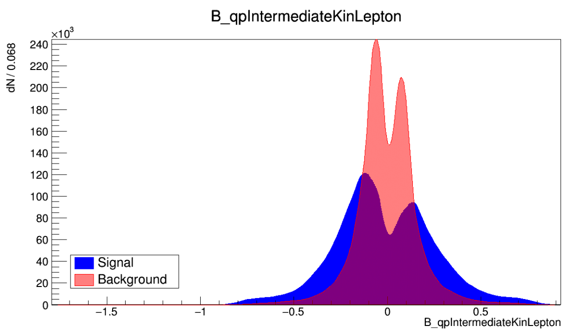
Figure 3.6: Some of the distributions of the KSFW moments

### 3.2.5 Flavor Tagging Features

The flavor tagging features are defined using a multivariate analysis method FastBDT. The principle is that if one of the entangled $B$-meson pair decays to CP eigenstate and other to the flavor specific state, the flavor of the latter can be determined at the time of its decay [Abudinn. 18]. Thirteen flavor tagging features have been described till now. The distributions for the two of the flavor tagging features are given in the Figure 3.7 and are different for signal and background events, thus can be used for the problem of continuum suppression.



(a) FastHadron ($\pi^+$,$K^-$ tagger)



(b) IntermediateKinLepton ($l^-$ tagger)

Figure 3.7: Some of the distributions of flavor tagging features

The features described above can be used all together to classify an event signal or background using the multivariate analysis methods (MVAs). Flavor tagging features are new that are being introduced, and high accuracy results have been obtained by using them with the previous ones as given in Chapter 6. The distributions of all the 44 variables are given in the Appendix A.

# Chapter 4

# Multivariate Analysis Methods

Multivariate Analysis (MVA) is used to analyze data sets having more than one variable [Dempster 71]. In the problem of continuum suppression, these methods can be used by loading the features and predicting the event as a signal or background. These are more useful when there is a correlation between different variables. Many different methods are being described for multivariate analysis like Fisher, Boosted Decision Trees, Neural Networks, etc. The method needs to be trained on the labeled training set so that it gets the best-fitted weights and then can be used on unknown data sets. Each model is described by a set of hyper-parameters which are required to be given by the user before training.

In this chapter, a brief introduction for Boosted Decision Trees and Deep Neural Networks is provided which are two most used multivariate analysis methods. These have been used for the problem of continuum suppression to separate signal and background events.

## 4.1 Boosted Decision Trees

Boosted Decision Tree (BDT) method is used for many problems in high energy physics. The problem of continuum suppression is a supervised learning problem because we have labeled data-set (events) which can be used for training. The schematic of supervised learning problem is given in Figure 4.1. The methods (BDT, Neural Network, etc.) are trained using the training data, and then the trained model can be used to classify unknown events.

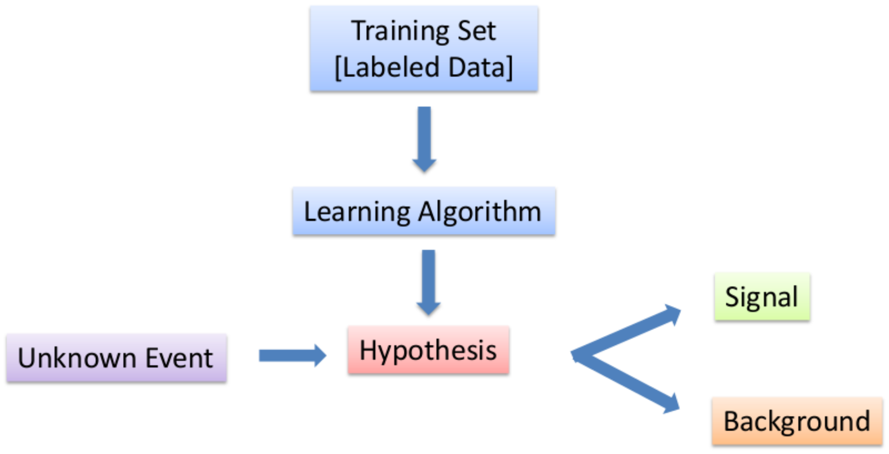


Figure 4.1: Schematic of supervised learning

Decision Trees consists of a consecutive set of nodes. These are used as a binary classifier [Drucker 95]. Each node has two outputs and a feature which is used for classification as given in Figure 4.2. The cut at each node and architecture of the tree is done by Classification And Regression Trees (CART) algorithm.

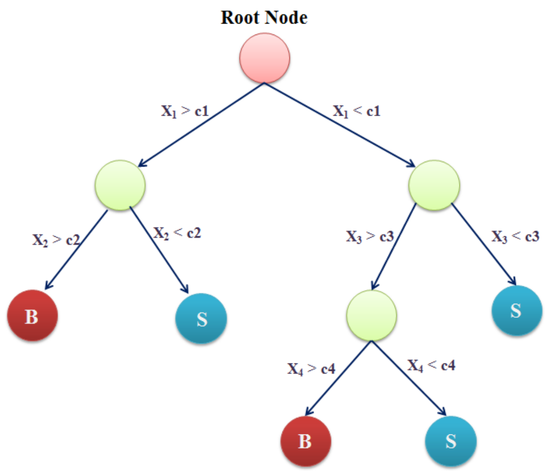


Figure 4.2: Schematic of a Decision Tree

Statistical fluctuations in the data can lead to an overtraining problem. Also, a single tree is not robust to problems with multiple features. To overcome this problem, a set of small decision trees are combined to form a forest known as random forest as given in Figure 4.3. The final decision is the majority of votes from all the trees. The small decision tree is called as a weak learner; thus a sum of weak learners results to a strong learner.

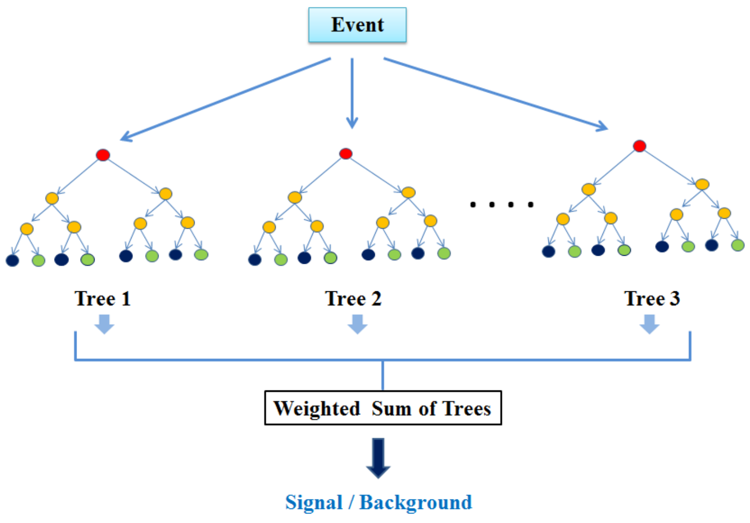


Figure 4.3: Schematic of a Random Forest

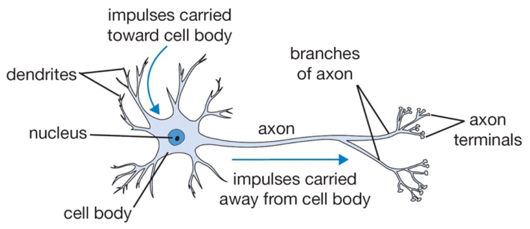
The most common method for training the random forest is Adaptive Boosting (AdaBoost) [Ke 17]. In this algorithm, higher weights are given to misclassified events and weights for correctly classified are reduced after each training. This is done to ensure that trees will do better on misclassified events in the following iterations.

The hyper-parameters that need to be defined by the user are the number of trees, the node size, the maximum depth of the decision tree and type of algorithm for boosting. By specifying the best parameters according to the problem, a good model can be constructed.

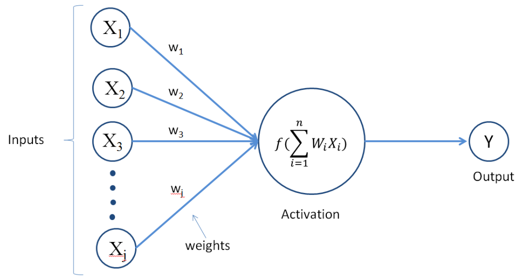
## 4.2 Artificial Neural Network

The concept of artificial neural network is inspired by the biological neural network. The mammal brain consists of about 100 billion neurons connected via dendrites. Neurons talk with each other via electrical signals that mediate through dendrites. There are thousands of connections between neurons which finally reach the cell body which consists of nucleus [Tang 07].

When the incoming signals from dendrites exceed some threshold value, the neuron will respond by generating a voltage impulse which is transmitted to other neurons by the axon. The generation of impulse depends upon the type of neurons that are being inputted. Some signals tend to prevent the impulse generation or firing while others encourage the firing. Thus the work of each neuron is to decide whether to generate impulse or not to depend on input signals and strength of its connections with other neurons.


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

(a) Biological neuron



(b) Artificial neuron (perceptron)

Figure 4.4: Comparison of a biological and artificial neuron working

In an artificial neural network, the nodes are equivalent to the neurons. The strength of a signal from input nodes is modeled by giving a weight to each input connection which gets multiplied by its value. The weighted signals are summed and then fed to the node which applies activation function. The output of the node after applying activation function is then transmitted to other neurons [LeCun 15]. The Figure 4.4 shows the equivalence between a biological and artificial neuron and the working of one artificial node/neuron. The 'network' term in neural network implies how the nodes are connected and its architecture. A different architecture is defined depending upon the problem and its complexity. Neural networks can be used to solve various problems like regression, classification, pattern recognition, object and anomaly detection. The problem can be categorized to supervised or unsupervised learning. Supervised learning problem includes the availability of labeled training data while in unsupervised learning, there is no labeled data but the algorithm looks for patterns in the data by its own to solve the problem.

### 4.2.1 Deep Neural Network

A deep neural network is defined as a set of layers which are connected to each other. Each layer has a defined number of nodes and connections runs between nodes of $i^{th}$ layer and $j^{th}$ layer. The network consists of an input layer, a set of hidden layers and an output layer as shown in Figure 4.5. A bias term (value) is given to each layer and is added to the weighted sum of nodes in the next layer. So each neuron works as ‘Input times weights, add bias and then activate it using the activation function’. The output of neural network with single hidden layer is given as:

$$y = F_2(\sum a_j \theta_j^{(2)} + b)$$

and

$$a_j = F_1(\sum x_i \theta_{i,j}^{(1)} + b_j)$$

where $i, j$ runs over number of features and number of nodes in the hidden layer respectively, $b$ is the biased term. $\theta_{i,j}$ are the weights defined from layer $i$ to layer $j$ and $F$ is the activation function [Schmidhuber 14].

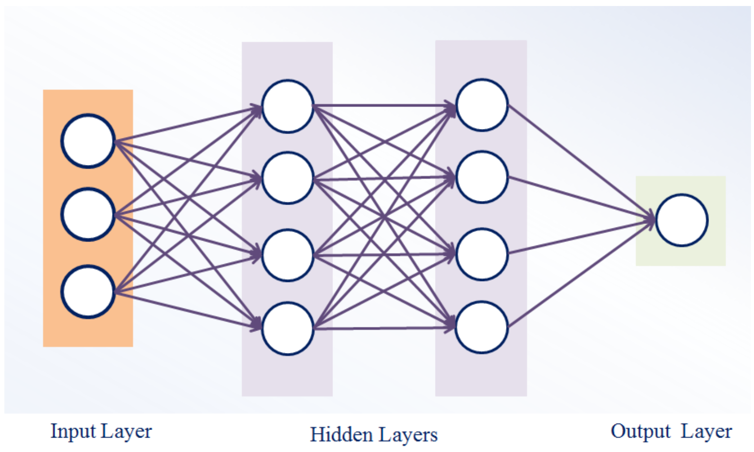


Figure 4.5: Schematic of the Neural Network

The input layer represents the features which are being used for training and testing. The nodes in the input layer equal the number of features. The nodes in the output layer depend upon the number of classes. For binary classification, the output layer can have 1 or 2 nodes depending on the activation of the output layer whereas for more than two classes the nodes are equal to the number of classes. The user defines the hidden layer and the number of nodes in each layer before training.

## 4.2.2 Back Propagation Algorithm

The algorithm used in multilayer neural network training is feed-forward backpropagation algorithm. It includes three stages:

1. Feed forward input features.

2. Calculating Error Term.

3. Re-weighting the weights depending on the error.

The data is first divided into training and testing set. The model (network) is then trained on the training data set using the backpropagation algorithm. In one cycle of algorithm one training example is used and then the next training example is used as an input feature vector, and the loop goes over all the training data. This makes the training time expensive as it includes much computation for each training example.

In the first stage, the input feature vector is fed into the network. All the weights are initialized randomly. Each hidden node gets the weighted sum over the input vector, and then it applies the activation function by adding a bias term. In this way, each hidden node in $i^{th}$ layer works, and it gives an output which is used for the $(i+1)^{th}$ layer as the input vector. Thus the feedforward propagates and gives output in the output layer.

After the feed forward, the output value of the training example is compared with the labeled or true value of that training example, and an error is calculated using a loss function. The user defines the loss function before training. The error is then used to re-weight the weights by the optimizer which is the application of gradient descent. The re-weighting is done by going backward in the network, and the weights for the last hidden layer are re-weighted first and finally for the input layer. That is why the algorithm is termed as Feed-forward Back-propagation algorithm. An epoch in a neural network is defined when all the training data is used for training once. Epoch number can be specified by the user to use the same training data to train the model again and again and thus getting better accuracy results [Bishop 06].

When dealing with a large amount of training data, it is better to divide the training data sets into batches. Then the algorithm works on each batch instead of each training example and makes the training faster and computationally efficient. The batch size should be based on the training data set size. By default, a batch size of 32 is considered in most of the deep learning frameworks.

### 4.2.3 Hyper-parameters for DNN

While defining the neural network, some parameters need to be specified by the user and are termed as hyper-parameters. These includes:

- Number of hidden layers
- Number of nodes in each hidden layer
- The activation function for each node
- Connection architecture in the network
- The loss function
- Optimizer
- Batch size
- Number of Epochs
- Regularization terms

It is important to choose these parameters sagaciously depending on the problem. The performance of the neural network relies on these. There is no hard rule defined to chose them, but it is possible to define a good network by looking at the best models already given for the different type of problems.

**Hidden Layers**: The parameters that need to be defined first are the number of hidden layers and nodes in each hidden layer. The hidden layers in the network range from one to five usually. The nodes in the hidden layer are equal or higher than the number of features in the input layer and can be different for each hidden layer. Higher the number of hidden layers or the nodes, more the network is prone to overtraining.

**Activation Function**: Many activation functions has been defined for the neural networks but the most commonly used are hyperbolic tangent ($Tanh$), rectified linear units ($ReLu$), sigmoid and softmax. $Tanh$ and $ReLu$ are preferably used for hidden layers while sigmoid and softmax are used for the outer layer. Sigmoid is used for problems with only one node in the output layer and softmax is used when there are more than one nodes (classes) for the output layer.

**Optimizer**: Optimizer plays an important role in the stability of the neural network. The optimizer used for most of the deep learning problems are stochastic gradient descent (SGD), Nesterov accelerated gradient (NAG), AdaGrad, AdaDelta and adaptive moment estimation (Adam). Each of them is used based on the structure of the data set available.

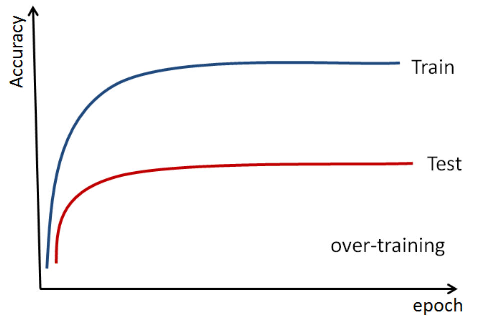
**Loss Function**: The loss (cost) function calculates the error term, and its value needs to decrease with each epoch for a good training model. Mean squared error, mean absolute error, categorical cross-entropy, binary cross-entropy, and squared hinge are some of the loss functions that are often used in deep learning problems.

**Epochs and Batch size**: The batch size depends on the training data. A fewer number of training data requires less batch size and vice-versa. The epoch number is chosen after checking the initial trend of accuracy and loss for a few numbers of epochs. If the accuracy for both training and testing data sets keeps increasing consistently, a higher value of epoch number can be set to get high accuracy results.

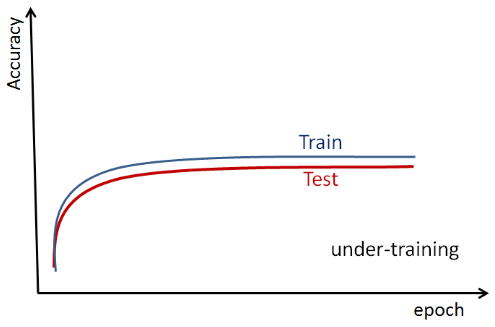
**Regularization terms**: Regularization terms are added to get rid of the overtraining problem. It includes adding a penalty to the cost function known as $L1$ and $L2$ regularization, adding dropouts, early stopping, maximum norm constraint, gradient clipping, and batch normalizing. Regularization is mostly done after checking the initial results from the model and to get an excellent model for a given problem.

### 4.2.4 Learning Curves

When building a neural network model, the main concern is to reduce the error (loss) and increase accuracy. The major problem in any neural network is over training or under training. Overtraining refers to a problem when the model performs better on the training examples, but the performance goes worst for the testing data set. It is also termed as high variance and low bias problem in data science. Under training is the case when the model performs badly on both the training and test data set. To check over training or under training, learning curves are plotted which tells about the performance of the model on training and test data set [Perlich 11]. The accuracy or loss is plotted against epoch number for both the train and test data set together. The Figure 4.6 shows the learning curve plot of the overtrained and undertrained model. The ideal model is the model in which the accuracy has a overall increase for both train and test data set consistently.



(a) Overtrained Model



(b) Undertrain Model

Figure 4.6: Accuracy plot for the over train and under train models

### 4.2.5 Regularization

When there is a large number of features, or the network architecture is complex, the neural network will probably prone to overtraining. Regularization involves methods to overcome the overtraining problem. The common methods to regularize the network are L1/L2 regularization, dropout regularization, and early stopping [Zhang 16]. The L1/L2 regularization works by penalizing the cost function. Extra terms are added to the cost function which prevents the model from overfitting. The cost function with regularization is defined as

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(y^i, \hat{y}^i) + L1/L2$$

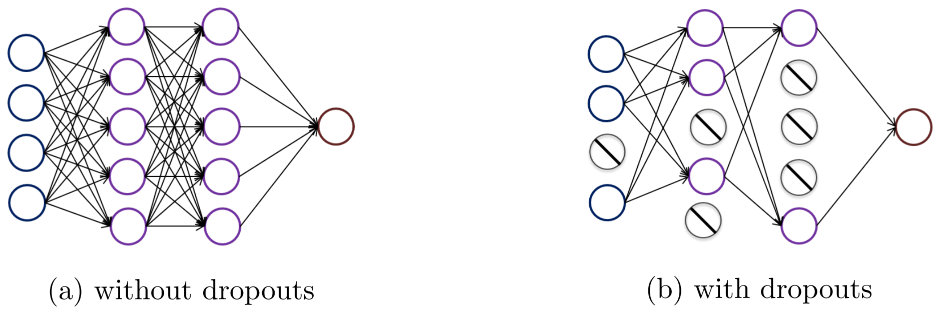
where m is the number of training examples, $w$ is weight, $b$ is bias term, $L$ is the loss function and $L1$, $L2$ regularization terms are defined as:

$$\text{L1 Regularization} \qquad \frac{\lambda}{m}\sum_{i=1}^{n} |w_i|$$

$$\text{L2 Regularization} \qquad \frac{\lambda}{m}\sum_{i=1}^{n} |w_i|^2$$

where $w_i$ are the weights for $i^{th}$ layer and $\lambda$ is a regularization parameter can be set based on the amount of regularization needed.

Dropouts are added to a layer, and it involves removing a fraction of nodes randomly while training the data. By removing nodes, the contribution of those nodes becomes zero for that batch size and thus helps in preventing the model from overtraining. The nodes removed are different and random when training is done over the next batch size. The dropout fraction is given as a parameter which tells about the percentage (probability) of nodes to be removed [Srivastava 14]. The Figure 4.7 shows the network before and after giving nodes.


(a) without dropouts

(b) with dropouts

Figure 4.7: Neural network without and with dropouts

Early stopping means to stop the training of data when the model starts behaving badly, or it becomes overtrained. This is done by checking the deviation of training and testing cost or accuracy plots and is shown in Figure 4.8. This way the weights

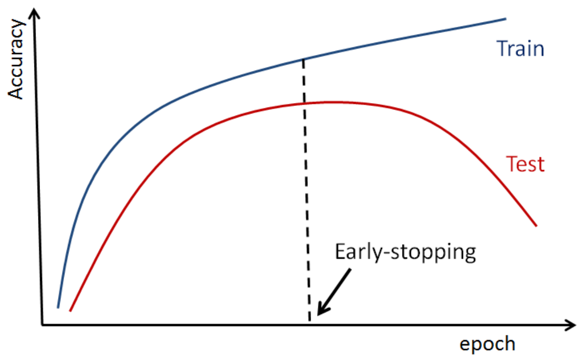


Figure 4.8: Example of early stopping

do not get overtrained values, but there are cases when the model behaves overtrained for only a few iterations and then restores, so early stopping needs to be implemented by knowing the behavior for a certain number of iterations only.

# Chapter 5

# Event Reconstruction

The data used for training and testing purposes for different multivariate methods is generated using Monte Carlo simulations using EvtGen package [Lange 01]. The signal events (collisions which results in $b\bar{b}$ production) are reconstructed using basf2 and Belle cluster whereas the background events (collisions which do not results in $b\bar{b}$ production) are reconstructed using data available at GRID servers.

## 5.1 Signal Events

The decay mode reconstructed for the signal events is given in Figure 5.1:

$$\begin{tikzcd}
B^+ \arrow[r] & D^0 \arrow[dr] & \pi^+ & & B^- \arrow[r] & D^0 \arrow[dr] & \pi^- \\
& & K^+ \pi^- \pi^0 & & & & K^- \pi^+ \pi^0
\end{tikzcd}$$

Figure 5.1: Decay mode analyzed

This mode is chosen for its high branching fraction. The decay mode is written in the decay.dec file and Monte Carlo events are generated using the EvtGen program. After the completion of the events generation, detector simulation is done using Geant4 (gsim) simulator detector [Agostinelli 03]. The output of the detector simulator is used for the reconstruction of the events which are specified in the python file for reconstruction. Events are generated parallelly using the Belle II cluster where many jobs are submitted simultaneously. Each job includes generation of 500 events,

| target | cut |
|---|---|
| $D^0$ | $1.835 \leq M \leq 1.885$ (GeV/$c^2$) |
| $B^+$ | $5.26 \leq M_{bc} \leq 5.29$ (GeV) |
| $B^+$ | $-0.15 \leq \Delta E \leq 0.15$ (GeV) |

Table 5.1: Selection cuts on the particles to build $B$ meson

and thus many jobs are submitted to obtain the required number of events. It ensures different seed for each job and thus making the signal data more robust. The final output root file contains the reconstructed features in the trees for training and testing purposes.

Number of signal events reconstructed : 12 Million

The $\Delta E$ and beam constrained mass ($M_{bc}$) are defined as

$$\Delta E = E_{beam} - E_b$$

$$M_{bc} = \sqrt{(E_{beam}^2 - p_b^2)}$$

where $E_{beam}$ is the beam energy and $p_b$, $E_b$ are the momentum and energy of decay particles respectively. The selection criteria for the event generation is given in Table 5.1

Figure 5.2 shows the invariant mass of the $D^0$ obtained from the reconstructed signal events.

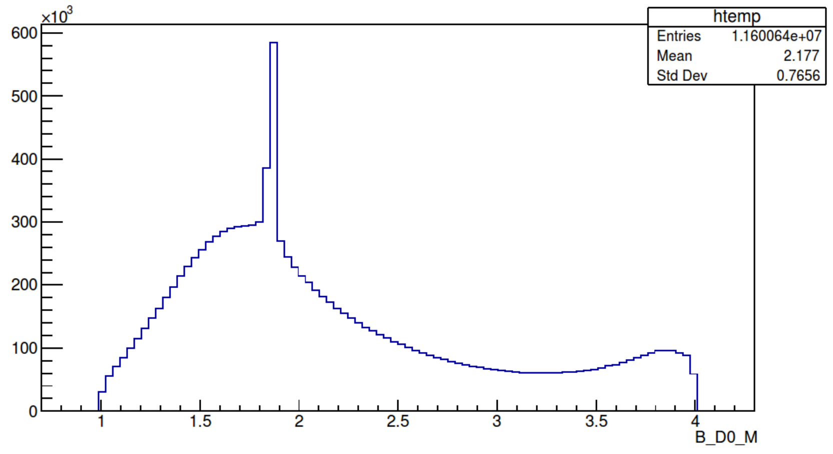


Figure 5.2: Invariant mass of the reconstructed $D^0$ from $B^+ \rightarrow D^0\pi^+$. The distribution is in GeV

The peak around 1.8 corresponds to the mass of $D^0$. Cutting down $D^0$ mass around this region and plotting $\Delta E$ and $M_{bc}$ gives the following plots in Figure 5.3 and Figure 5.4 respectively.

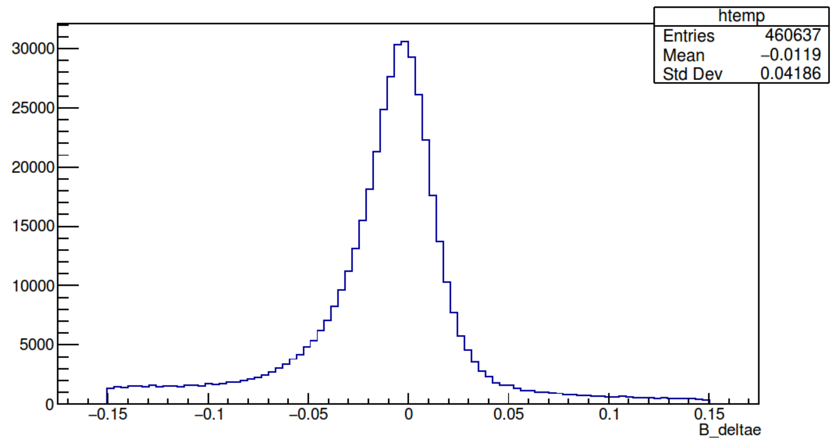


Figure 5.3: Reconstructed $\Delta E$ (in GeV)

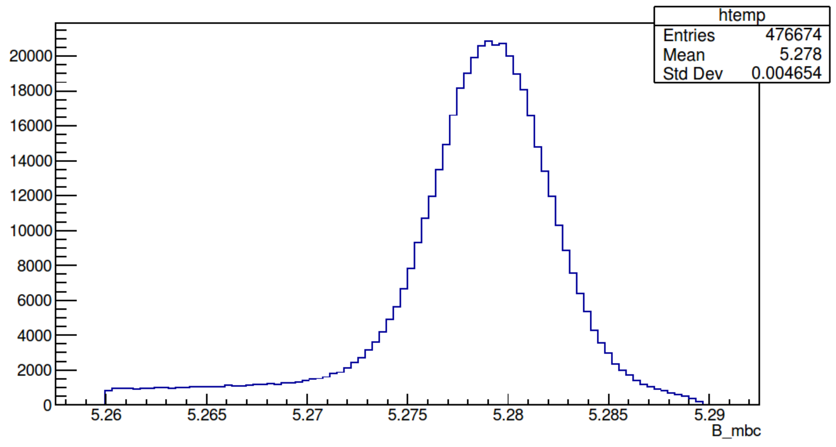


Figure 5.4: $M_{bc}$ distribution for $B^+ \to D^0\pi^+$ decay mode (in GeV/$c^2$)

## 5.2 Continuum Events

The simulated continuum (background) data is available at the GRID. The events are reconstructed using gbasf2 and the GRID server. Around 12 million events are reconstructed which includes almost equal number of the $u\bar{u}$, $d\bar{d}$, $s\bar{s}$ and $c\bar{c}$ events as given in Table 5.2. All are concatenated to form a single file which contains all the background/continuum data [MC9 ]. The $\Delta E$ and $M_{bc}$ plots from the continuum events obtained are given in Figure 5.5 and 5.6 respectively.

| Event Type | Number of Events |
|---|---|
| $u\overline{u}$ | 3 Million |
| $s\overline{s}$ | 3 Million |
| $c\overline{c}$ | 3 Million |
| $d\overline{d}$ | 3 Million |
| Total | 12 Million |

Table 5.2: Reconstructed background data

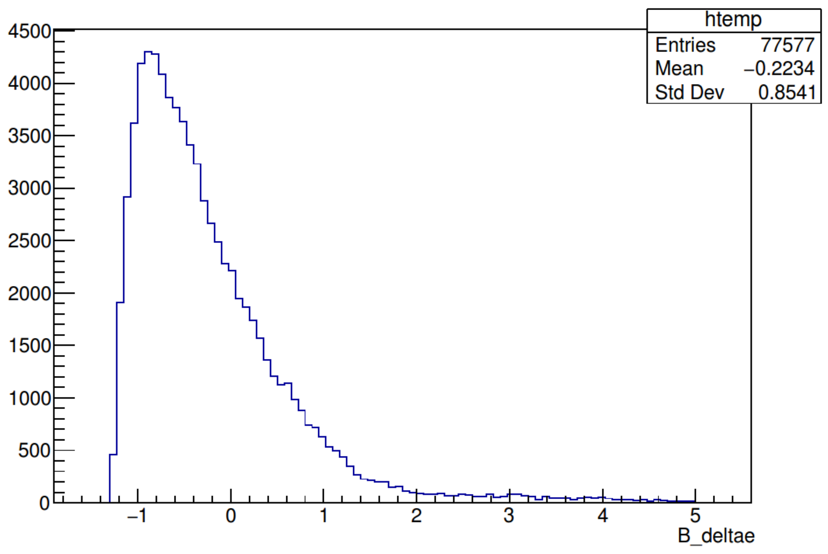


Figure 5.5: Reconstructed $\Delta E$ (in GeV) (continuum events)

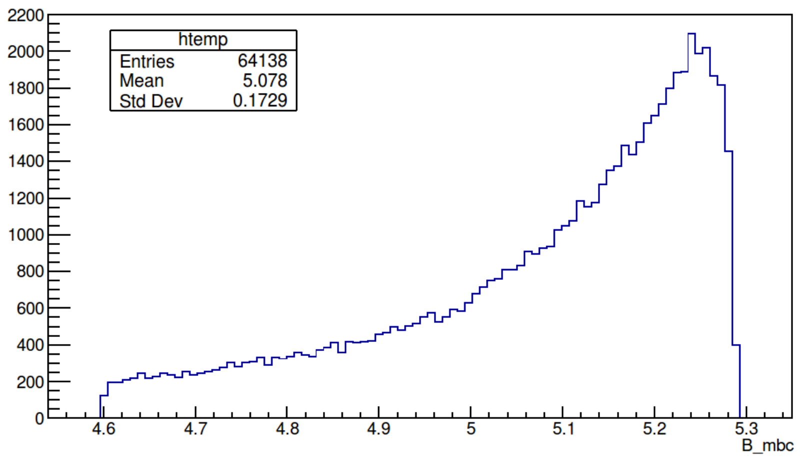


Figure 5.6: $M_{bc}$ distribution for continuum events(in GeV$/c^2$)

# Chapter 6

# DNN for Continuum Suppression

The continuum suppression is a binary classification with two outputs and a set of input features. Various machine learning algorithms and methods are being used for the problem. Many new features have been introduced to get better accuracy results. The open source framework like TensorFlow, Keras has been constructive in applying different deep learning models more efficiently and quickly. Deep learning refers to using the neural networks to get the optimal model for the classification with good accuracy and thus, can be used to classify the unknown events.

This chapter explains about different methods that are being used for continuum suppression and the results obtained from them. Section 6.1 provides information about the data set and the features that has been used in the problem. Section 6.2 provides information on TMVA package which is used for deep neural network and boosted decision trees. Section 6.3 focuses on deep neural network framework Keras and Tensorflow which are used for training and testing the different models and the results obtained. In the last section, some methods to extract the feature importance is discussed.

## 6.1 Data Features

The features that are described in Chapter 3 are used for the problem of continuum suppression and it consists of total 44 features as given in the Table 6.1.

The data obtained by Monte Carlo simulation as described in Chapter 5 is used as the benchmark dataset. It contains around 12 million signal events and an almost

| **Feature Type** | **Number of features** |
|---|---|
| Thrust | 2 |
| Angular | 2 |
| Cleo-Cones | 9 |
| KSFW Moments | 18 |
| Flavor tagging | 13 |
| Total | 44 |

Table 6.1: Feature Count

equal amount of background events which has all the four major background events. A total of around 24 million labeled data set is used for training and testing purpose. The data is shuffled before training to ensure a better result. The shuffled data is divided into 70% training set and 30% as the testing set as given in Figure 6.1. While

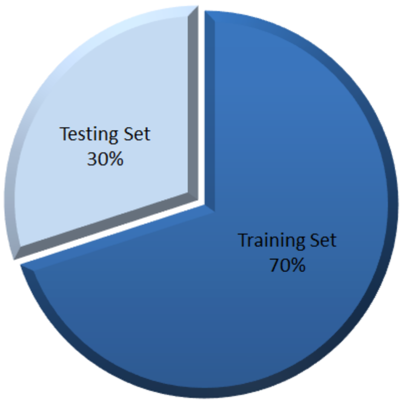


Figure 6.1: Train and Test data split

training only the training data set is used, and the independent testing set is used for testing and finding the accuracy and other metrices.

## 6.2 TMVA

The toolkit for multivariate data analysis (TMVA) is provided with ROOT to do multiple variable analysis in high energy physics problems. The package includes different methods and algorithms for example likelihood estimation, artificial neural network, support vector machine, boosted decision tree, etc. to do the data analysis. The code implementation is object-oriented in C++/ROOT for different multivariate analysis (MVA). It comes with different scripts for training, testing, and visualization of data including different scripts for regression and classification problems. This package is useful to compare different models and algorithms by using the same training and test data. The package also includes a graphical user interface (GUI) to analyze the output of the model [Hocker 07].

Pre-processing in TMVA includes applying different transformations to the data set before doing training and testing. The pre-defined transformations available in TMVA are uniform, identity, Gaussian, decorrelation, principal component analysis (PCA) and these transformations can be applied one after another in a sequence.

The method to check the efficiency of any model in TMVA is given by Receiver Operating Characteristic (ROC) curves which tell about the true positive rate versus the false positive rate. ROC curve is a plot between signal efficiency and background rejection and the area under the curve (AUC) tells about the performance of given model (more the area under the curve better the performance) [Fawcett 06].

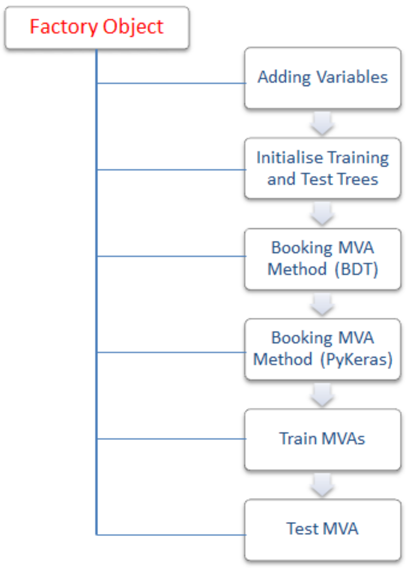


Figure 6.2: Sequence of commands in TMVA

The multivariate analysis methods in TMVA are implemented by first instantiating a *Factory* object. The data set is then given to the *Factory* object by initializing the training and testing Trees. The MVA methods are implemented by using BookMethod of *Factory* object. Different transformations can be done by giving the options for transformations as arguments while booking the MVA methods. The training and testing are done by the train and test instances of factory object. The sequence of commands to train methods in TMVA is given in Figure 6.2.

### 6.2.1 TMVA - Results

A small set of data is used for the initial analysis in TMVA. The data set includes equal amount of signal and background events as given in Table 6.2.

| **Type** | **Number of events** |
|---|---|
| Signal Training Events | 52000 |
| Background Training Events | 52000 |
| Signal Testing Events | 22000 |
| Background Testing Events | 22000 |

Table 6.2: Train Test Split while using TMVA

Boosted Decision tree (BDT) method using Adaptive boosting is used to compare the performance of the model by using a different set of features. The parameters of the BDT method is given in Table 6.3.

| **BDT method parameters** | |
|---|---|
| Decision Trees Trained | **850** |
| Boosting Method | **Adaptive Boosting** |
| Separation Index | **Gini** |

Table 6.3: Parameters used for Boosted Decision Tree in TMVA

The Figure 6.3 shows the ROC curve for the BDT-Adaptive Boosting method and compares different features. The blue curve is for the case when only thrust and angular features (total 4) are used for the training and testing. The green curve shows the performance when Cleo-cones are used along with thrust and angular features. The purple curve uses thrust, angular, Cleo cones and KSFW moments as features for the method. The last curve, the red one is for the case when all the 44 features which include thrust, angular, Cleo-cones, KSFW moments and flavor tagging variables are being used for the BDT method.

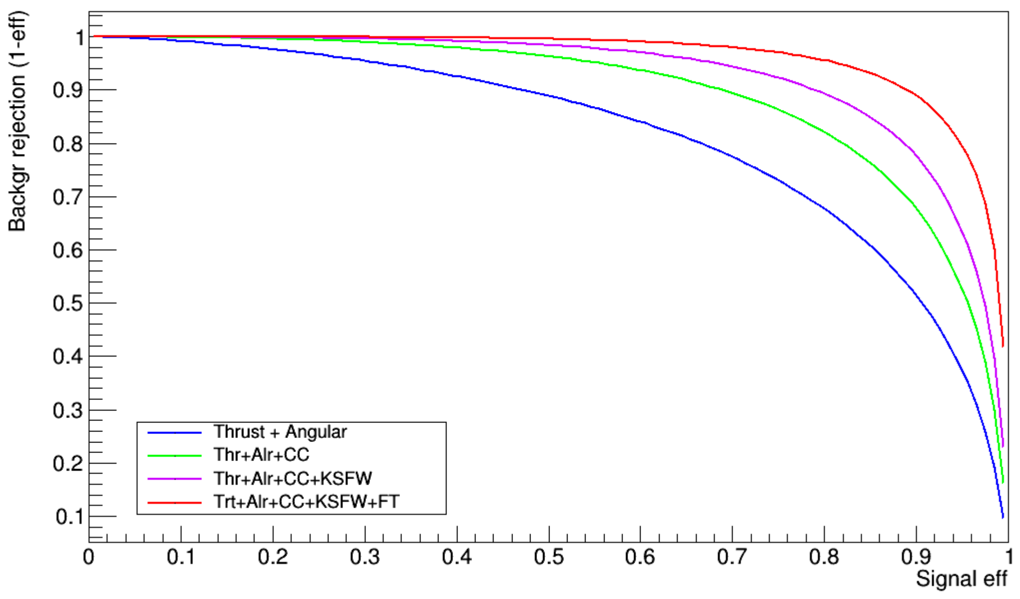


Figure 6.3: ROC curve by using different feature set

The area under the curve (AUC) get larger when more features are added and thus supporting the contribution of each feature in the classification. The area is maximum when all the 44 features are used together thus confirming the addition of flavor tagging as one of the critical features in the continuum suppression.

By using all the features, different methods can be compared to check the best method to classify the events. Figure 6.4 shows the ROC curve in which the performance of different methods is given. The methods compared are Fisher, BDT-Gradient boosting, BDT-Adaptive boosting and PyKeras. PyKeras refers to the artificial neural network with parameters given in Table 6.4.

The area under the curve for PyKeras method is maximum among all the methods. Thus, artificial neural networks perform best for continuum suppression, and this method can be optimized more by setting the best parameters and analyzing the outputs. Neural networks implementation is given in more detail in Section 6.3 where open source library Keras is used to train and test the model.

| PyKeras parameters | | |
|---|---|---|
| Input layer | **44 nodes** | |
| Hidden Layer 1 | **64 nodes** | **Activation - ReLu** |
| Hidden Layer 2 | **64 nodes** | **Activation - ReLu** |
| Hidden Layer 3 | **64 nodes** | **Activation - ReLu** |
| Output Layer | **2 nodes** | **Activation - Softmax** |
| Loss Function | **Categorial Crossentropy** | |
| Optimizer | **SGD (Stochastic Gradient Descnet)** | |

Table 6.4: Hyper-parameters used for neural network in TMVA

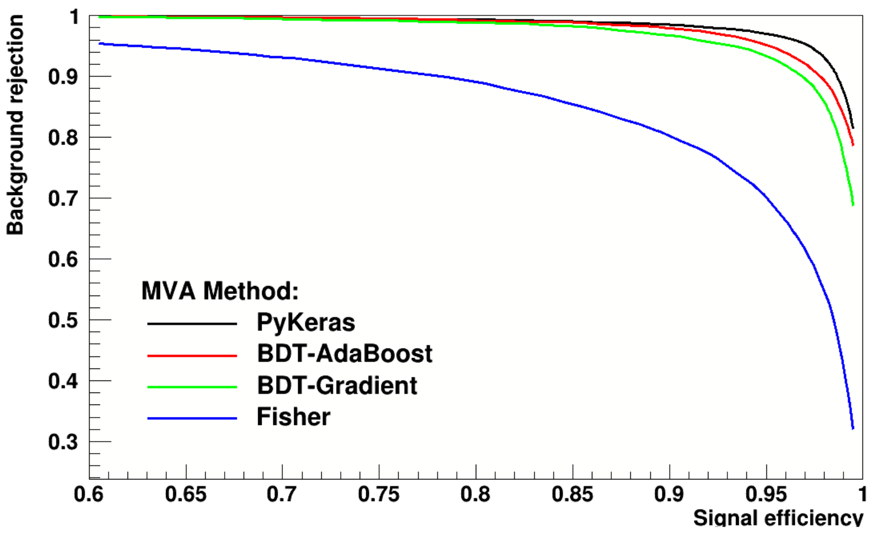


Figure 6.4: ROC curve by using different methods

## 6.3 Keras and TensorFlow

Keras and TensorFlow are the open source frameworks designed specifically to solve deep learning problems using neural networks. Keras uses TensorFlow at the backend, and it provides an easy implementation for the neural networks. In this thesis work, Keras is used to implement all the neural network models. The network architecture, optimizer, loss function, regularization terms, and epochs are defined in the model class of the Keras library. Keras and TensorFlow libraries are designed to use all the computer resources including GPUs efficiently and thus minimize the time taken by the neural network to train [Abadi 16]. The whole data set as given in Section 6.3 is used to train and test models in Keras. The data has around 24 million signal and background events and 70 : 30 split is done for training and testing respectively.

The Figure 6.5 shows the accuracy plot for the model that has four hidden layers with no dropouts. The nodes in the hidden layers are 64, 32, 32 and 64 respectively and activation function for each hidden layer is rectified linear unit (ReLu). The output layer has two nodes with softmax as the activation function. The loss function used for the model is cross-entropy, and the optimizer is Stochastic gradient descent (SGD). The blue line corresponds to the training data accuracy, and the red line is for the testing data accuracy. Both the train and test curves are consistent which

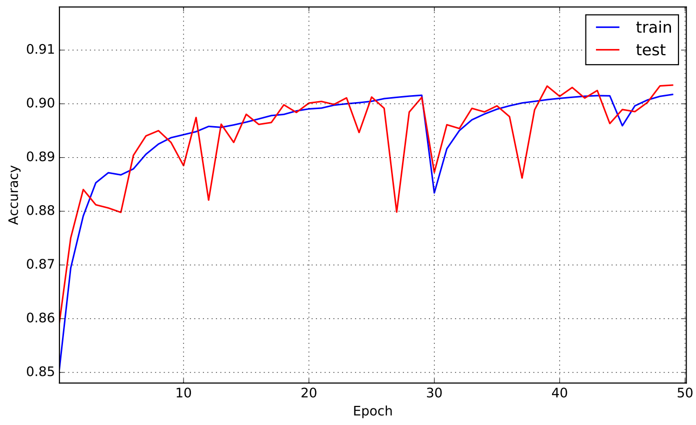


Figure 6.5: Accuracy plot for train and test data sets

ensures that the model is trained properly. The model around epoch 12 can be said to be overtrained because the training accuracy is high while the testing accuracy drops, but the model regains the accuracy again and becomes consistent. The decline at around epoch number 30 can be seen as undertrained as both the test and train accuracy drops. The maximum accuracy obtained is about 90% for this model.

The loss plot for the same model as described above is shown in Figure 6.6 . The accuracy and loss plots are opposite and consistent for both train and test data. The minimum loss obtained from the model is about 0.25.

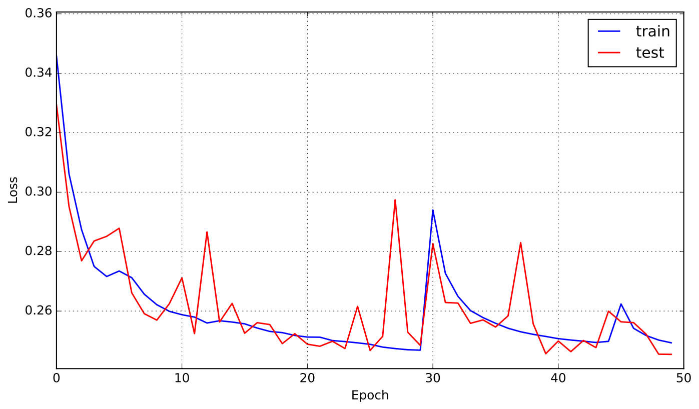


Figure 6.6: Loss plot for train and test data sets

Different models are compared by training on the same data set. Figure 6.7 shows the accuracy plot for the training data set for different models. The labels tells about

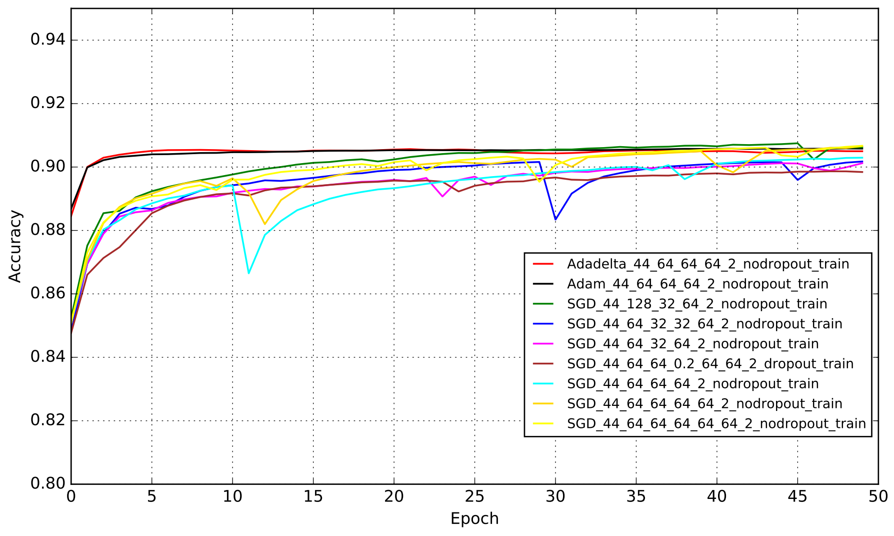


Figure 6.7: Accuracy plots for different models

the network architecture and fraction term specify the dropout added to that layer.

For instance, the plot labelled as 'SGD_44_64_64_0.2_64_64_2_dropout' refers to the model which uses SGD as optimizer and has 4 hidden layers with 64 nodes for each hidden layer and 20% dropout for the second hidden layer. The loss function used for all the models is cross-entropy with ReLu as the activation layer for each hidden layer. The model with Adam as an optimizer is more stable than other. The loss for the model is given in Figure 6.8

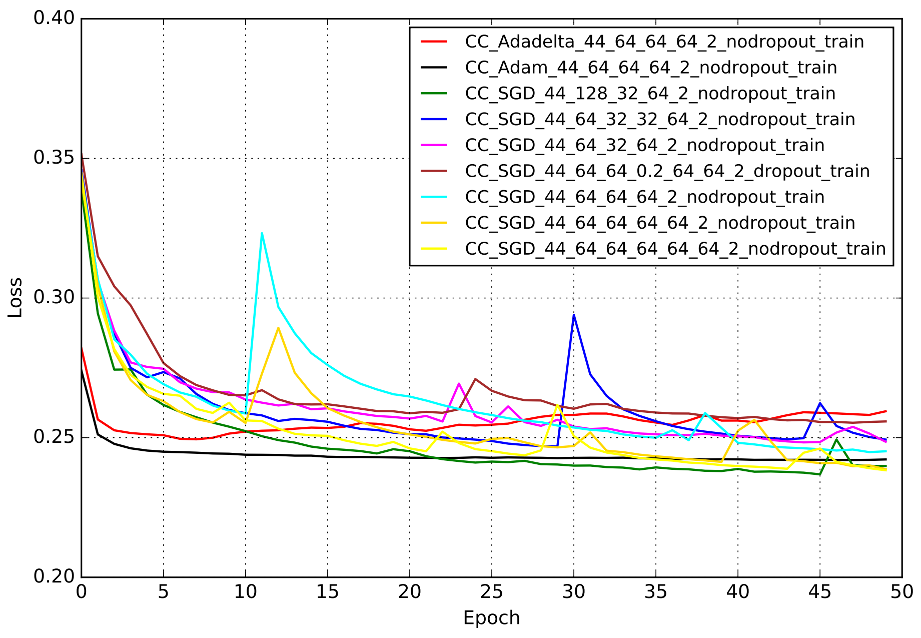


Figure 6.8: Loss function plots for different models

### 6.3.1 Optimal Network

The different network models described in the previous section are unstable to some extent. More models with different parameters are compared to check for the possible best combination of hyper-parameters for continuum suppression. The optimal network architecture which has been obtained in this thesis work is given in the Figure 6.9. It includes three hidden layers with 256, 128 and 64 nodes for the hidden layer 1, 2 and 3 respectively and rectified linear unit (ReLu) as the activation function for each layer. The output layer has one node with sigmoid as the activation function. The loss function used for this model is the root mean square error and optimizer is Adam. Dropouts are added to each layer for regularization.

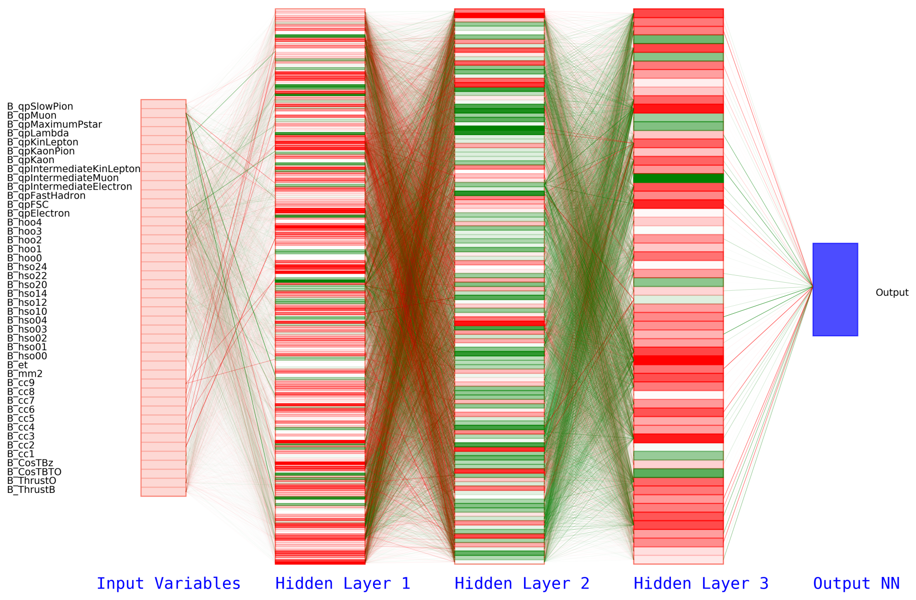
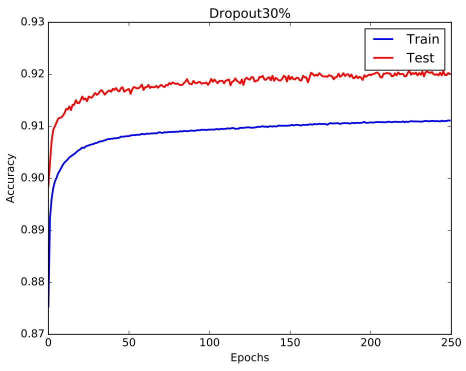


Figure 6.9: Optimal network architecture. The red colour corresponds to positive weight values while green is for negative weight values and intensity tells about the weight strength.

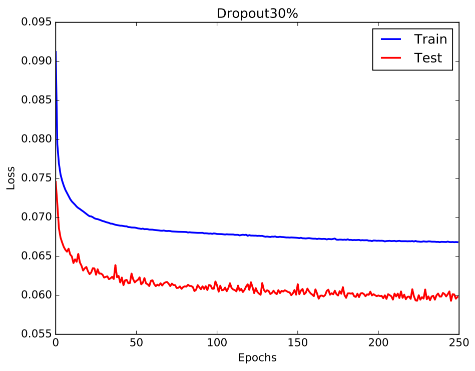
The accuracy and loss plots for the optimal model with different dropouts are shown in Figure 6.10, Figure 6.11 and Figure 6.12. The high separation between the train and test plot for the models with 30% and 20% dropouts is due to the under training. Thus, by reducing dropouts to 10%, gives a better model with higher accuracy results of about 93%. The test accuracy is higher than the training accuracy due to the addition of dropouts. The reason for this is that while training the fraction of nodes are completely removed randomly as given by the dropout term and their contribution becomes zero, but when testing, all the nodes are used which gives higher accuracy for the test data set.

The separation plot for the signal and background events is given in Figure 6.13. The signal purity is defined as

$$\text{Signal Purity} = \frac{\text{No. of true positive(signal) above cut}}{\text{Total no. of events}}$$

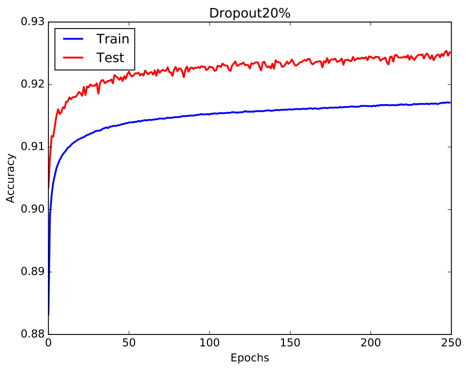


(a) Accuracy plot

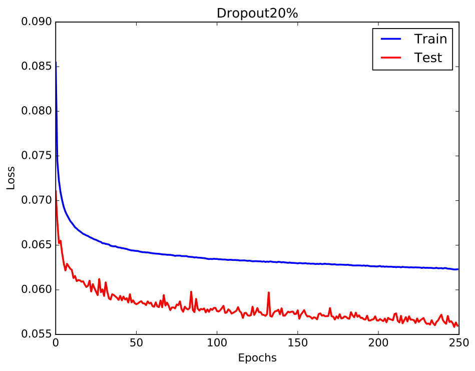


(b) Loss plot

Figure 6.10: Accuracy and loss plots for the optimal network with dropout of 30%

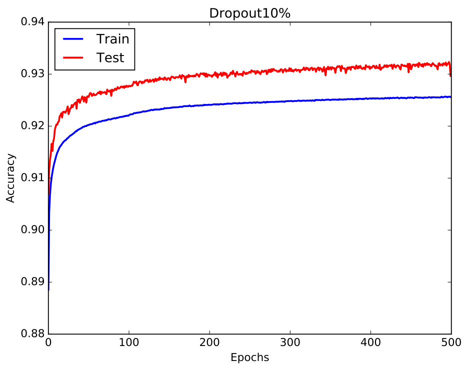


(a) Accuracy plot

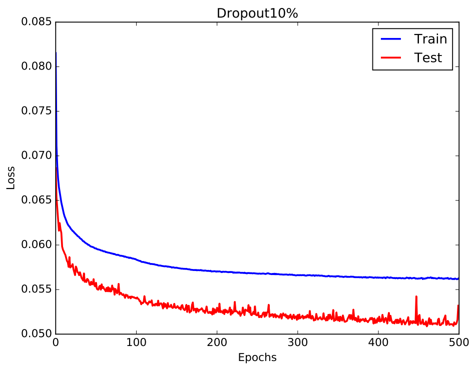


(b) Loss plot

Figure 6.11: Accuracy and loss plots for the optimal network with dropout in 20%



(a) Accuracy plot



(b) Loss plot

Figure 6.12: Accuracy and loss plots for the optimal network with dropout of 10%

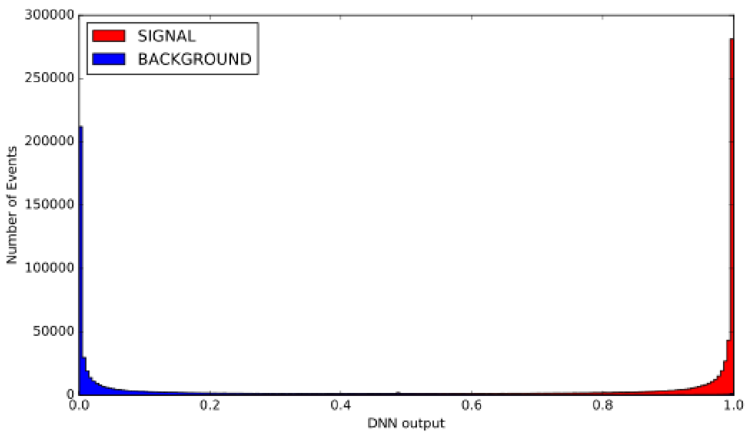


Figure 6.13: Separation plot for the optimal network

| DNN Cut | Signal Purity |
|---|---|
| 0.5 | 94.53% |
| 0.6 | 96.06% |
| 0.7 | 97.12% |
| 0.8 | 97.97% |
| 0.9 | 98.71% |

Table 6.5: DNN cut vs Signal Purity

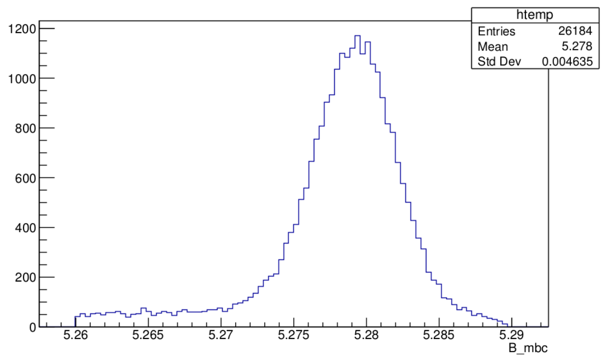


Figure 6.14: $M_{bc}$ distribution after 0.6 DNN cut (in GeV/$c^2$).

The Table 6.5 shows the Signal purity values after applying different cuts. The $M_{bc}$ distribution for the sample test data is given in Figure 6.14 after applying neural network cut at 0.6. The background rejection in this case is found to be around 95%.

## 6.4 Feature Importance

There has not been a proper package or method available yet to check the importance of each variable in the classification problem. The most preferred method is to remove each feature in the neural network and then check the accuracy of the model. The decrement in the accuracy by removing a feature tells about the importance of that feature. The only problem in this approach is the time taken for each model to train and test particularly in the continuum suppression problem when there are 44 input features and big data set. It is useful if a good amount of computation resources are available so that all the models with different features can be tested simultaneously. In this thesis work, two different approaches have been used to check the importance of each feature.

In the first approach, the trained model is used to check feature importance. The trained model uses all the 44 features while training. The model is used to test the accuracy on a test data set by shuffling the data of each feature one by one. Shuffling the data means to shuffle the data column of one feature at a time and to keep all other features data same; thus it gives an error to that feature for each training event. In this way, each feature data is shuffled one at a time and accuracy is plotted. The feature for which the accuracy decreases most by shuffling can be seen as an important feature. This may not be a proper method, but at least it tells that if there is some error in some particular feature while measurement or in the calculation, it can lead to wrongly classifying the events. The Figure 6.15 gives the plot of accuracy and the feature that is shuffled.

The second approach is based on the Shapley variables as defined in [Lundberg 17]. It uses the concept of game theory to check for feature importance. A small number of data set is used as a background, and the events are tested using that background data set by the DeepExplainer method of shap. The Figure 6.16 shows the shap values of each data feature. The features with a lower value can be seen as important features.

A similar trend can be seen in the two plots obtained from different approaches to get the feature importance. The Pearson correlation is 0.89 for the output values from the two approaches. Thus, Cleo cones 1,2,3 along with missing mass square features can be considered important for the problem of continuum suppression.

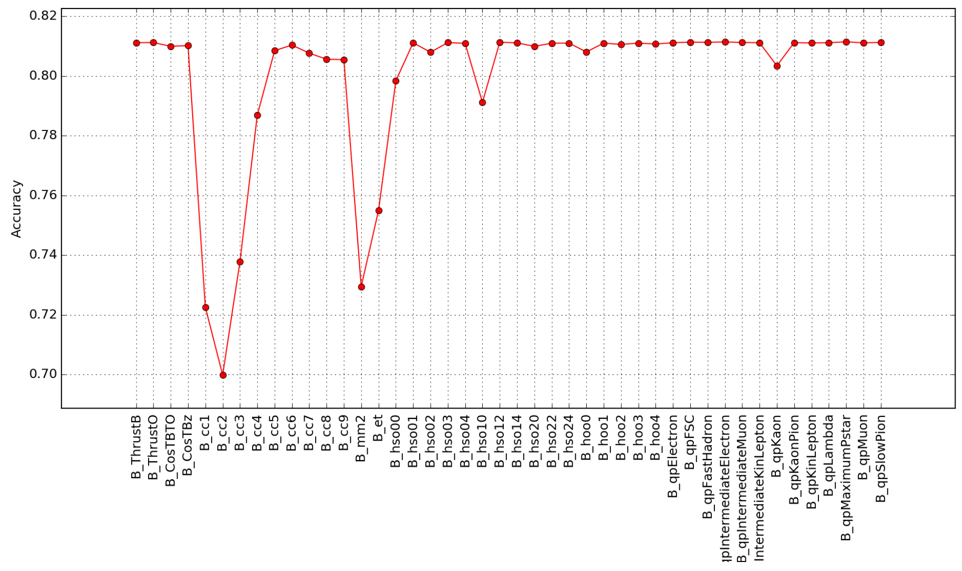


Figure 6.15: Accuracy plot after shuffling each data feature

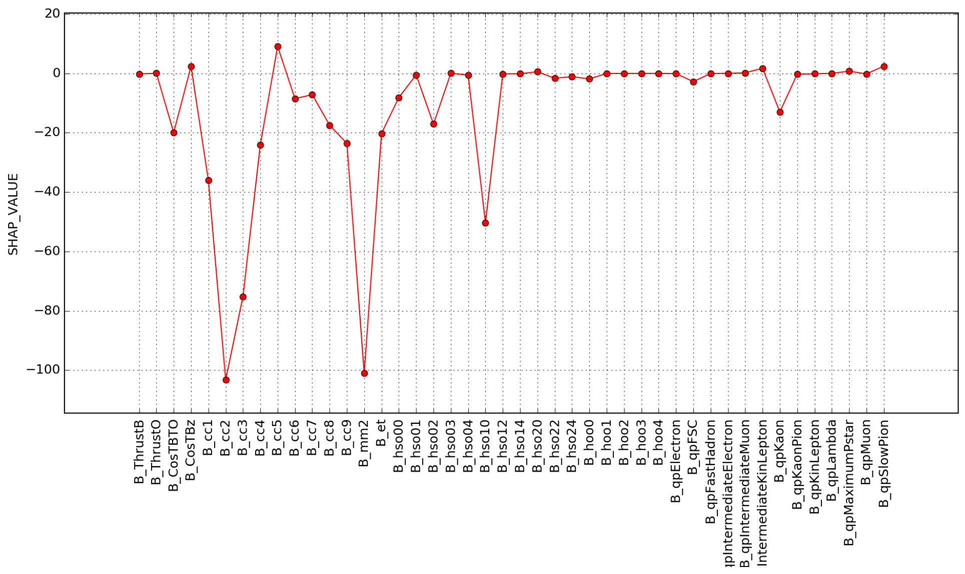


Figure 6.16: Shap values for each data feature

The top 8 ranks of the features are given in Table 6.6. All other features are also important but the major contribution in continuum suppression is coming from these features.

| Rank | From Shap Value | Shuffling Method |
|---|---|---|
| 1 | B_cc2 | B_cc2 |
| 2 | B_mm2 | B_cc1 |
| 3 | B_cc3 | B_mm2 |
| 4 | B_hso10 | B_cc3 |
| 5 | B_cc1 | B_et |
| 6 | B_cc4 | B_cc4 |
| 7 | B_cc9 | B_hso10 |
| 8 | B_et | B_hso00 |

Table 6.6: Top eight ranks

# Chapter 7

# Summary & Conclusions

In this thesis work, the problem of continuum suppression has been analyzed using various algorithms and frameworks. The flavor tagging features are newly added to the analysis along with thrust, Cleo cones and KSFW moments and thus comprising of total 44 features. The $B^+ \to D^0(K^+\pi^-\pi^0)\pi^+$ decay mode is used as a signal event. The data is simulated using EvtGen package provided in basf2. An equal amount of continuum (background) data is reconstructed from the GRID server. The complete data set is simulated, and no real data is being used to test for the performance.

The initial analysis is done using the TMVA (Toolkit for multivariate analysis) package provided by root. The flavor tagging features contribute significantly to continuum suppression. Various methods are compared, and the performance of the deep neural network is found to be best among all. Extensive work has been done on the implementation of neural network using open source libraries specifically Keras with Tensorflow at backend. Many network architecture is trained and tested to get optimal hyper-parameters. PyTorch (an open source deep learning library) is also used to analyze the data, but it was not optimal to use all the computer resources efficiently, and the performance was almost the same as that of Keras. The importance of features is discussed using two independent methods, and a significant correlation is found among the two methods.

The performance has increased by around 4% which is equivalent to 2 months of data taking. This helps in improving the sensitivity of the experiment. It is importance for this experiment which is going to run for next ten years.

The performance of neural networks can be improved by implementing different network architectures and training over more number of events. This will require a high-performance computer resource which has sufficiently high RAM and GPU's memory. New features can be extracted that can be added to the analysis part, and thus a more robust network can be attained. The model implementation on real data will provide new insights, and more accurate results can be obtained. Further one can test this method in data for the control sample.

## Future work

To implement the general framework for training and testing of neural network in the basf2 and testing the optimal model on real data set.

# Bibliography

[Abadi 16] Martín Abadi*et al.* *TensorFlow: A System for Large-Scale Machine Learning.* In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 265–283, Savannah, GA, 2016. USENIX Association.

[Abudinn. 18] Fernando Abudinn. *The Belle II flavor tagger.* Jul 2018. 12+3 min.

[Adachi 18] I. Adachi, T.E. Browder, P. Krian, S. Tanaka & Y. Ushiroda. *Detectors for extreme luminosity: Belle II.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 907, pages 46 – 59, 2018. Advances in Instrumentation and Experimental Methods (Special Issue in Honour of Kai Siegbahn).

[Agostinelli 03] S. Agostinelli*et al.* *GEANT4: A Simulation toolkit.* Nucl. Instrum. Meth., vol. A506, pages 250–303, 2003.

[Aushev 10] T. Aushev*et al.* *Physics at Super B Factory.* 2010.

[Bishop 06] Christopher M. Bishop. Pattern recognition and machine learning (information science and statistics). Springer-Verlag, Berlin, Heidelberg, 2006.

[Dempster 71] A.P. Dempster. *An overview of multivariate data analysis.* Journal of Multivariate Analysis, vol. 1, no. 3, pages 316 – 346, 1971.

[Drucker 95] Harris Drucker & Corinna Cortes. *Boosting Decision Trees.* volume 8, pages 479–485, 01 1995.

[Fawcett 06] Tom Fawcett. *An Introduction to ROC Analysis.* Pattern Recogn. Lett., vol. 27, no. 8, pages 861–874, June 2006.

[Fox 78] Geoffrey C. Fox & Stephen Wolfram. *Observables for the Analysis of Event Shapes in e+ e- Annihilation and Other Processes.* Phys. Rev. Lett., vol. 41, page 1581, 1978.

[Hocker 07] Andreas Hocker*et al.* *TMVA - Toolkit for Multivariate Data Analysis.* 2007.

[Ke 17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye & Tie-Yan Liu. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, editeurs, Advances in Neural Information Processing Systems 30, pages 3146–3154. Curran Associates, Inc., 2017.

[Kobayashi 73] Makoto Kobayashi & Toshihide Maskawa. *CP-Violation in the Renormalizable Theory of Weak Interaction.* Progress of Theoretical Physics, vol. 49, no. 2, pages 652–657, 02 1973.

[Kuhr 18] Thomas Kuhr, Christian Pulvermacher, Martin Ritter, Thomas Hauth & Nils Braun. *The Belle II Core Software.* Jul 2018.

[Kurokawa 03] S. Kurokawa & Eiji Kikutani. *Overview of the KEKB accelerators.* Nucl. Instrum. Meth., vol. A499, pages 1–7, 2003.

[Lange 01] D. J. Lange. *The EvtGen particle decay simulation package.* Nucl. Instrum. Meth., vol. A462, pages 152–155, 2001.

[LeCun 15] Yann LeCun, Y Bengio & Geoffrey Hinton. *Deep Learning.* Nature, vol. 521, pages 436–44, 05 2015.

[Li 15] Yao Li. *Analysis on $B^{\pm} \to \pi^{\pm}\pi^{+}\pi^{-}$.* PhD thesis, Virginia Tech., 2015.

[Lundberg 17] Scott M Lundberg & Su-In Lee. *A Unified Approach to Interpreting Model Predictions.* In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, editeurs, Advances in Neural Information Processing Systems 30, pages 4765–4774. Curran Associates, Inc., 2017.

[MC9 ] *DESY Confluence.* `https://confluence.desy.de/display/BI/DataProductionMC9`.

[Miyake 15] Hideki Miyake, Rafal Grzymkowski, Radek Ludacka & Malachi Schram. *Belle II production system.* Journal of Physics: Conference Series, vol. 664, no. 5, page 052028, dec 2015.

[Perlich 11] Claudia Perlich. *Learning Curves in Machine Learning.* 01 2011.

[Schmidhuber 14] Jürgen Schmidhuber. *Deep Learning in Neural Networks: An Overview.* CoRR, vol. abs/1404.7828, 2014.

[Srivastava 14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever & Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research, vol. 15, pages 1929–1958, 2014.

[Tang 07] W. K. S. Tang, M. Yu & L. Kocarev. *Identification and monitoring of biological neural network.* In 2007 IEEE International Symposium on Circuits and Systems, pages 2646–2649, May 2007.

[Weyland 02] Dennis Weyland. Continuum Suppression with Deep Learning techniques for the Belle II Experiment. Master's thesis, KIT, Karlsruhe, ETP, 2017-11-02.

[Zhang 16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht & Oriol Vinyals. *Understanding deep learning requires rethinking generalization.* CoRR, vol. abs/1611.03530, 2016.

# Appendix A

# Event Topology Features

## A.1 Thrust

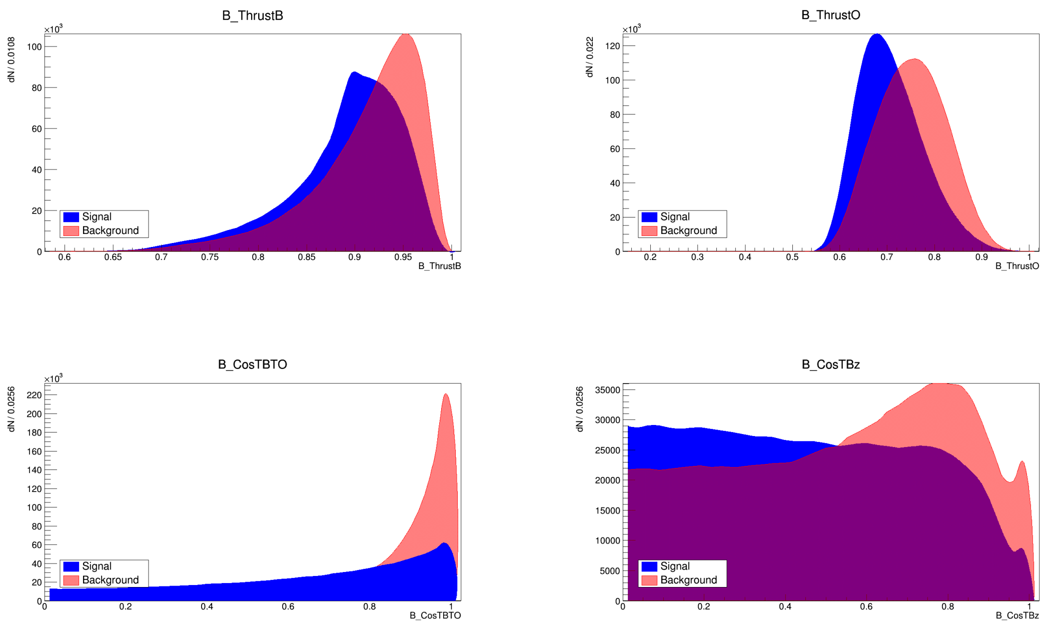


Figure A.1: Distribution of the Thrust Features

## A.2 Cleo Cones

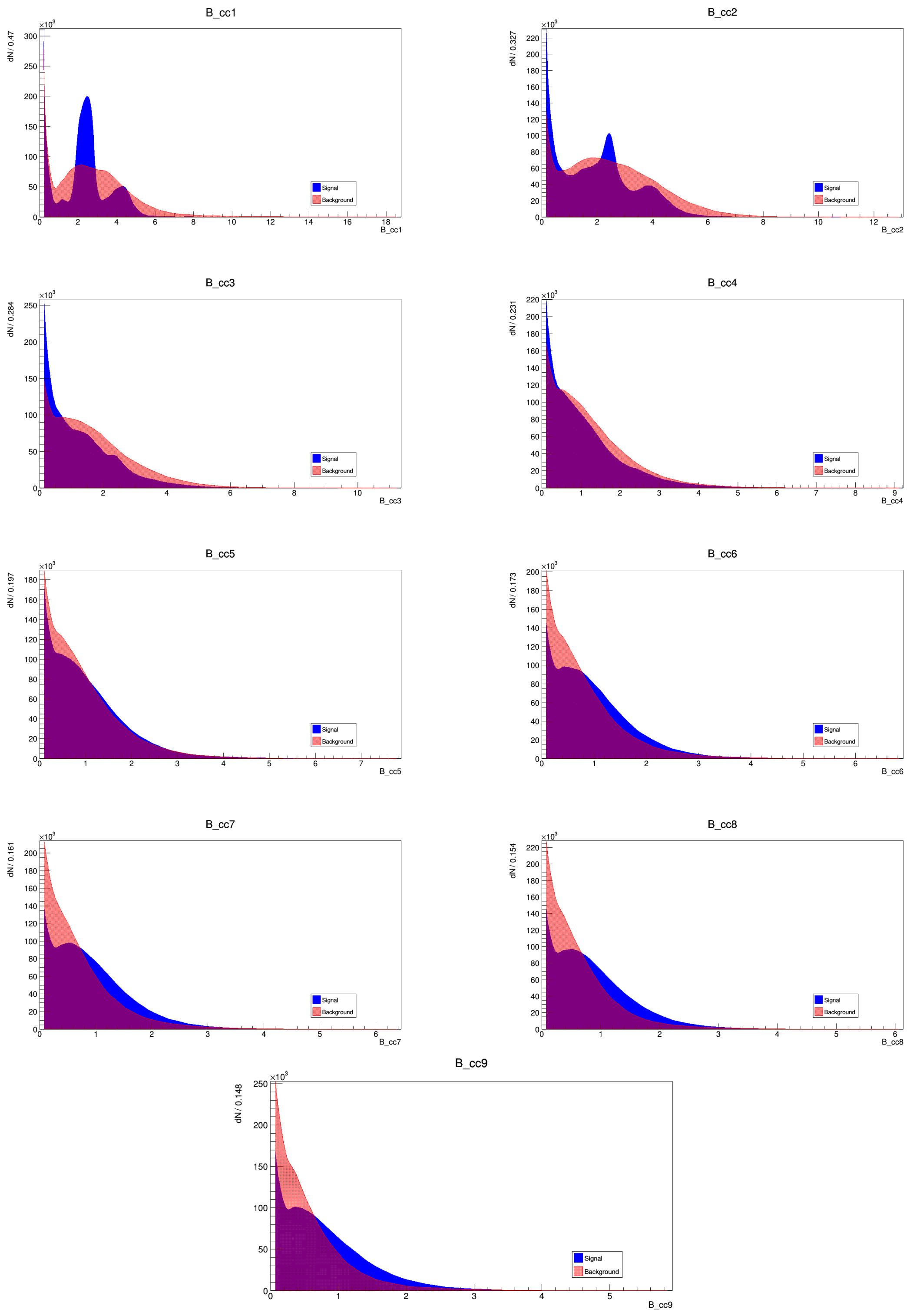
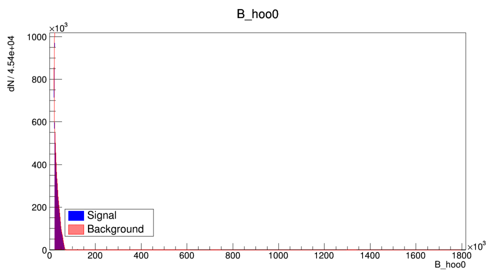
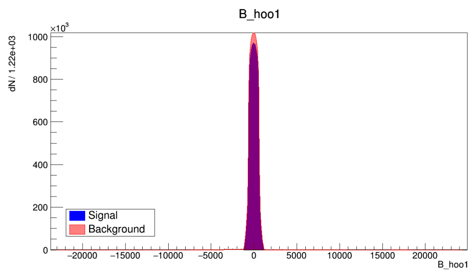
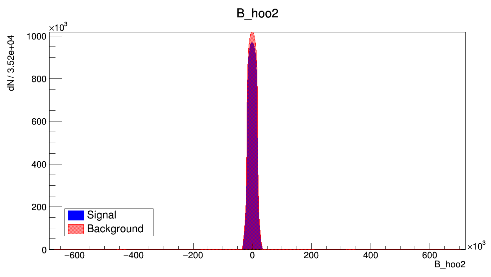


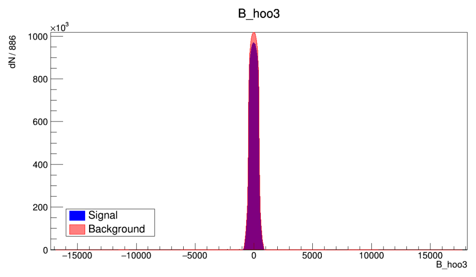
Figure A.2: Distributions of the Cleo cones for signal and background events

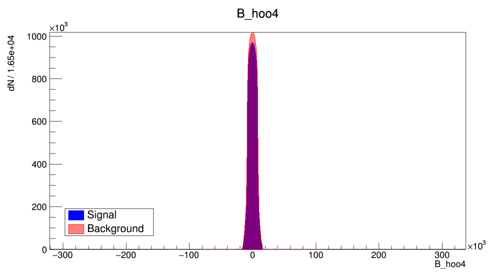
## A.3 Fox-Wolfram Moments

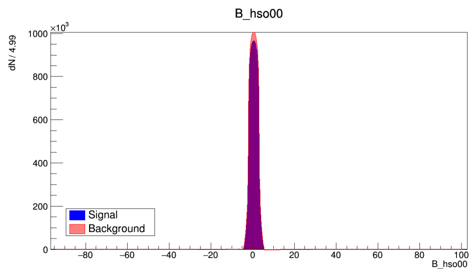


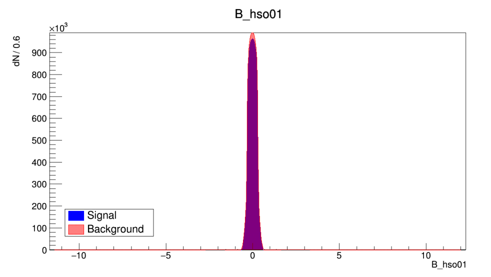


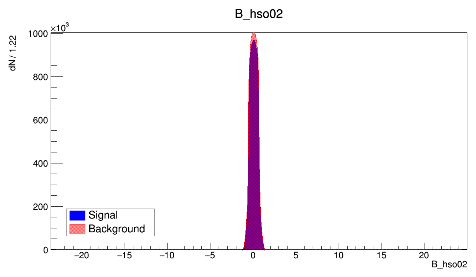


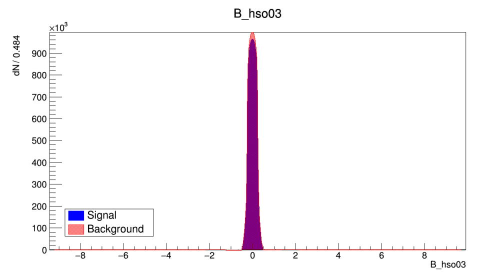


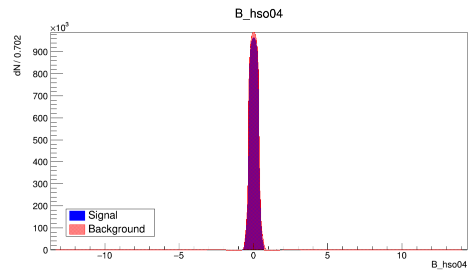












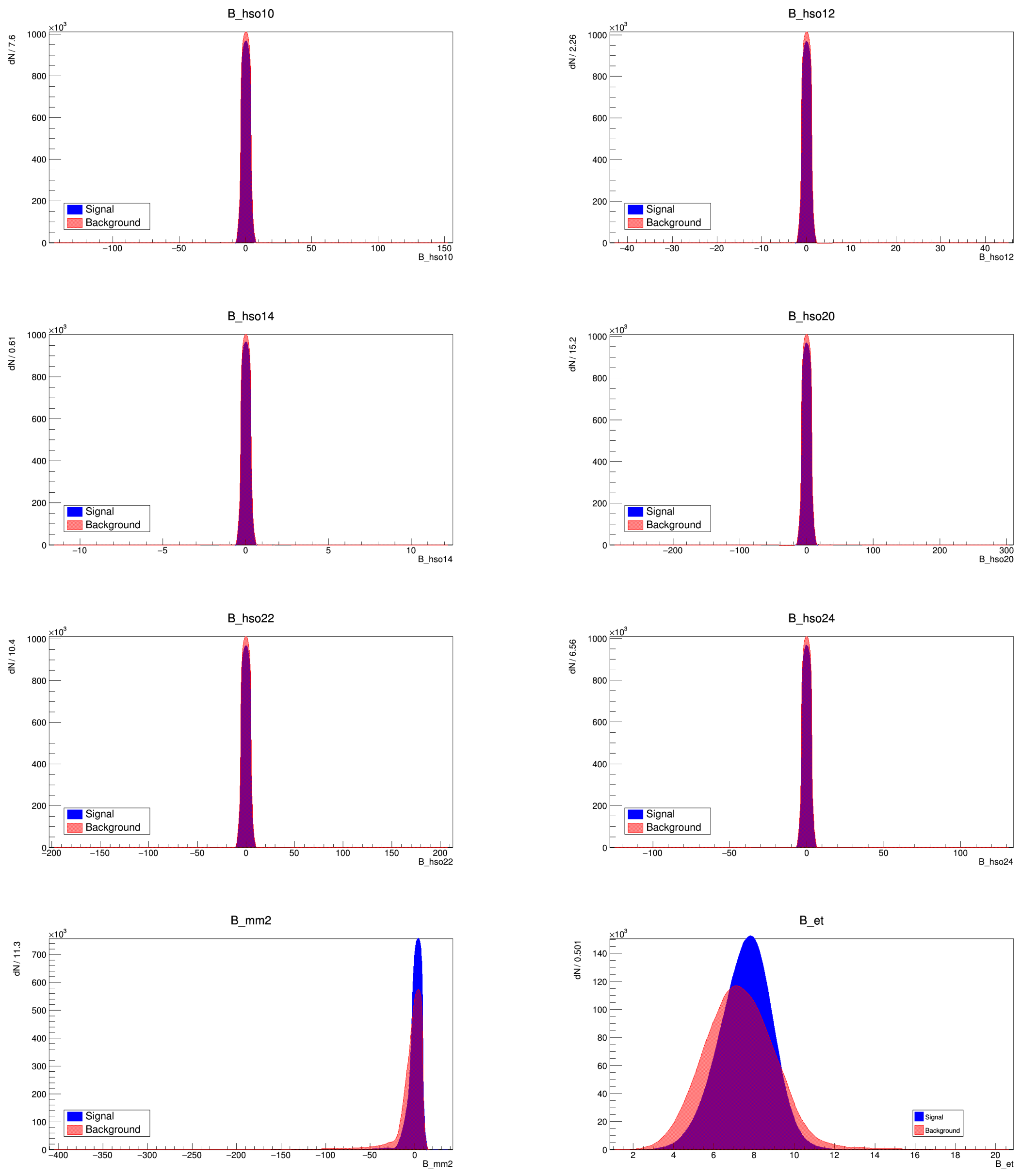
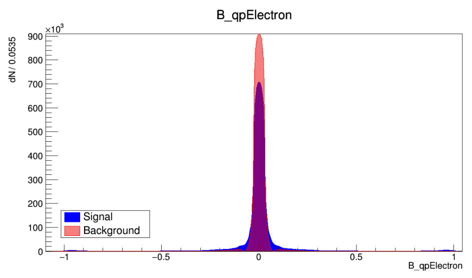
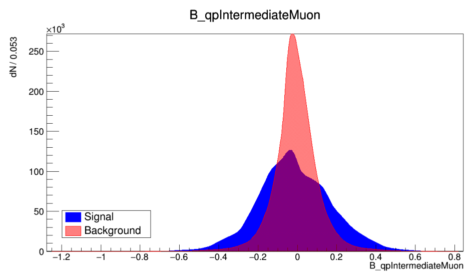
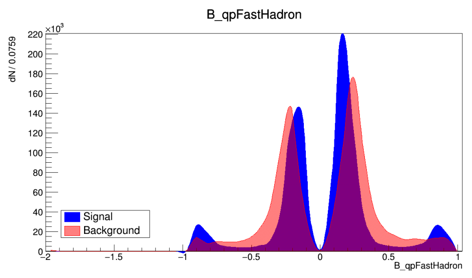


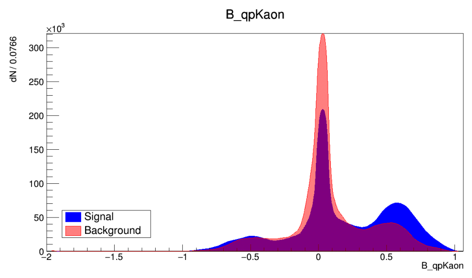
Figure A.3: Distributions of the FW and KSFW moments for signal and background events

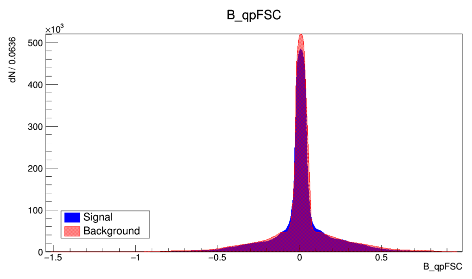
## A.4 Flavor Tagging Variables

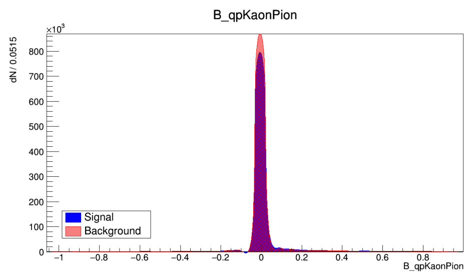


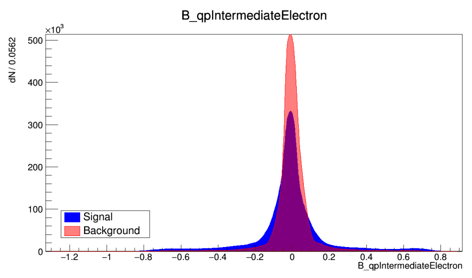


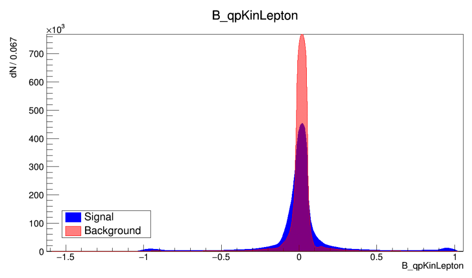


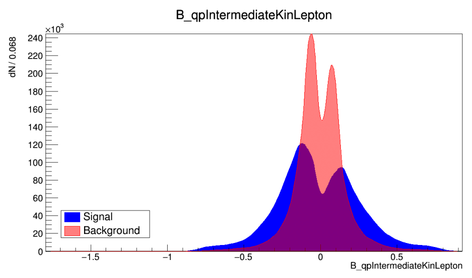


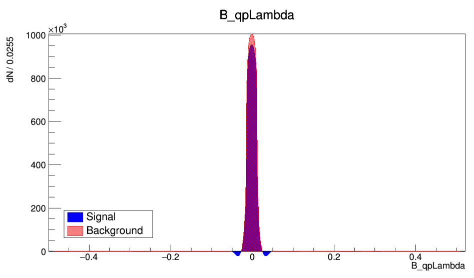












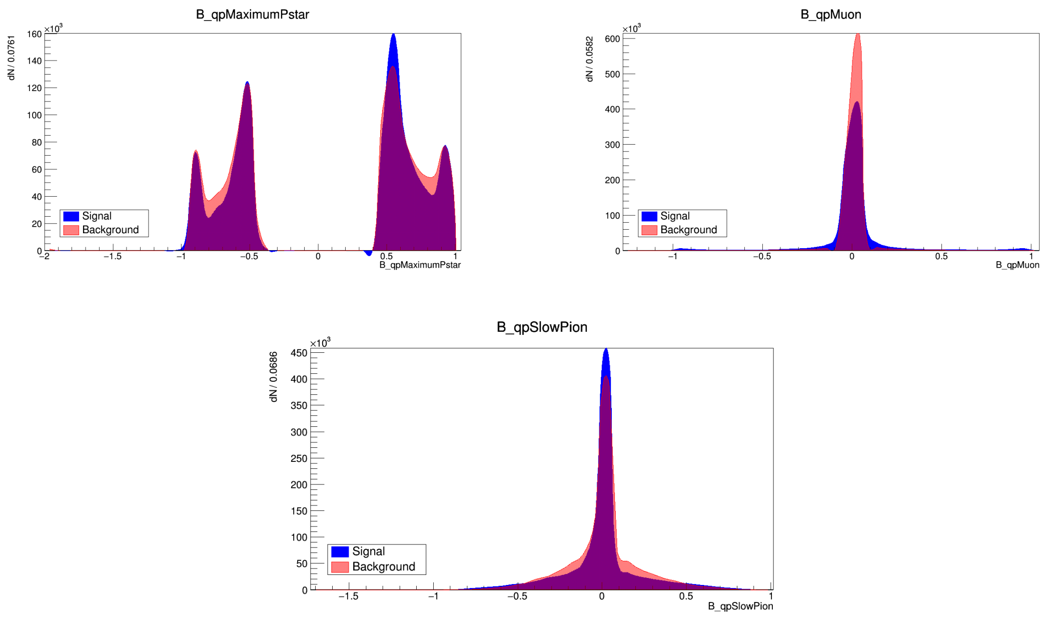


Figure A.4: Distributions of flavor tagging variables for signal and background events

# Index