Classification of B Meson Daughter Particles Using Machine Learning Methods at Belle II

Niklas Heckmann

Master's Thesis
at the Faculty of Physics
Ludwig–Maximilians–Universität München
Chair of Elementary Particle Physics

Supervisors:
Prof. Dr. Thomas Kuhr
Dr. Nikolai Hartmann

Munich, 13th September, 2024

Klassification von B Meson Tochtereilchen mit ML Methoden am Belle II Experiment

Niklas Heckmann

Masterarbeit
an der Fakulität für Physik
Ludwig–Maximilians–Universität München
Lehrstuhl für Experimentelle Flavorphysik

Betreuer:
Prof. Dr. Thomas Kuhr
Dr. Nikolai Hartmann

München, den 13. September 2024

Abstract

With new advances in machine learning B-tagging at Belle II could be improved, making data usage more efficient. This thesis presents a new way of B-tagging, where a ML classification algorithm does not follow the classical way of fully reconstructing events, but to brute force cluster all final state particles into tag and signal side. The network achieved great results on MC truth data level clustering $\sim 20\%$ perfectly. The reconstruction of final state particles from detector simulation caused some issues and resulted in the final B-tagging accuracy significantly loosing performance. This proof of concept however shows that with some better final state particle reconstruction, it has a lot of potential in achieving great B-tagging results.

Contents

Chapter 1

Introduction

For years, particle physicists have been trying to answer a few questions about our universe at the elemental level. What are we made of? How do the fundamental forces work? What happened during the first seconds after the Big Bang? The best and closest answer to these questions is the Standard Model of Particle Physics (SM). The SM provides excellent results when testing it on the interactions in the universe; however, it lacks in explaining some phenomena we observe. It can't explain gravity, the matter-antimatter asymmetry, dark matter, dark energy, neutrino oscillations, just to name a few.

To understand these phenomena, particle physicists use particle accelerators to test various theories that could explain the gaps left by the Standard Model. These accelerators collide particles at speeds very close to the speed of light and then use detectors around the collision vertex to examine what kind of interactions occurred after the collision. There are two main kinds of modern particle accelerators. First, the high-energy accelerators accelerate heavy baryons like protons. These collisions have very high energy, enabling the possibility of directly creating heavy and exotic particles, but they also come with a lot of background events that create a lot of noise in the detector data. One of these is the LHC at CERN in Geneva, Switzerland. On the other hand, some accelerators use less energy and lighter collision particles like electrons. With their ultra-sensitive detectors and comparatively low background, one can search for rare decays to test the Standard Model for deviations[26].

One of the leading particle accelerators belonging to the latter group is the SuperKEKB, located in Tsukuba, Japan. Here, e^+ and e^- collide at the center of mass energy of around 10.5 GeV. This is the resonance energy of the $\Upsilon(4S)$ state, a bound state between a b and a $\bar{b}$ quark. This then almost certainly decays further into two B mesons, which then further decay into many different intermediary particles until reaching the final state particles that can be detected using the detectors. The main detector at the SuperKEKB accelerator is called the Belle II detector.

Rare decays that contain neutrinos are especially hard to find, as they can not be detected

by Belle II. For that reason physicists try to fully reconstruct one of the B-mesons (tag), to constrain the other B-meson (signal) that contains these rare decays. As the sheer combinatorial possibilities of such a decay shower are too large to solve analytically, machine learning algorithms are used. The reconstruction algorithm used by Belle II is called the Full Event Interpretation (FEI) [15]. It uses multivariate boosted decision trees (BDT).

FEI takes the measurements of the detector and predicts the type, charge, momentum, etc., of the final state particles (FSP) of a given event. Then it hierarchically builds up the whole decay tree by predicting certain FSPs decaying out of an intermediary particle. In the process, many events must be cut out of the analysis as the algorithm often does not have a high certainty of these connections and ultimately the decay trees. The hierarchical approach of building up the decay tree can lead to easy errors, e.g. if just one stage of the reconstruction has a wrong prediction, the errors in the step-wise buildup can accumulate. Also, only easily identifiable decay chains can be considered when trying to reconstruct the tag B meson. FEI will be covered in more detail in chapter 2.5.

In the rising age of machine learning (ML), the idea of an end-to-end alternative for FEI seems realistic. One of the largest strides in this area was made by an ML algorithm called GraFEI [24]. GraFEI takes in the information of the final state particles and trains a so-called graph neural network (GNN) on it. Rather than defining possible decay chains to an algorithm like FEI and building the decay chain from the bottom up, the GraFEI model aims to learn how to reconstruct an event from scratch with no prior information.

Similar to GraFEI, this thesis explores the possibility of an end-to-end solution using a machine learning algorithm, with a slightly different approach. Rather than reconstructing entire decay trees, the plan is to only reconstruct the FSPs and then cluster them into two groups, corresponding to one or the other B meson.

Chapter 2

Background

This chapter will explain all the relevant background information necessary regarding the Belle II experiment. Chapter 2.1 will start by providing more detail about the Standard Model and all its components. In chapters 2.2 and 2.3 the Belle II hardware and software components will be described. Chapter 2.4 will explain how the tagging side method functions and what it is used for. Finally, in chapter 2.5, the FEI will be explained in more detail followed by a short introduction to GraFEI.
.

2.1 Standard Model

The Standard Model of particle physics was first developed in the 1970s and arose from the mathematical framework of Quantum Field Theory (QFT) [22]. It describes all known elementary particles and their interactions, except gravity. It is one of the most successful theories in physics, as it can explain almost every experimental result with high accuracy. Since the discovery of the Higgs Boson in 2012 all predicted particles of the SM have been experimentally verified[2].

There are two main categories of particles in the SM: Fermions and Bosons. Fermions are spin 1/2 particles and are considered the building blocks of matter, whereas Bosons have integer spins and are the interaction particles of the SM.

Fermions underlie the Fermi-Dirac statistics, meaning they have to obey the Pauli exclusion principle. This states that no two Fermions, of the same type in the same quantum state can be within the same system. Each Fermion has a corresponding antiparticle. The antiparticle has the same properties as its regular particle counterpart with the exception that its charge, baryon- and flavor quantum numbers are flipped. All Fermions are split into two groups, called quarks and leptons, which are further divided into three generations. Each of the leptons also has a sister particle in the SM called the neutrino. Neutrinos are neutral particles with almost no mass and are therefore very difficult to detect.

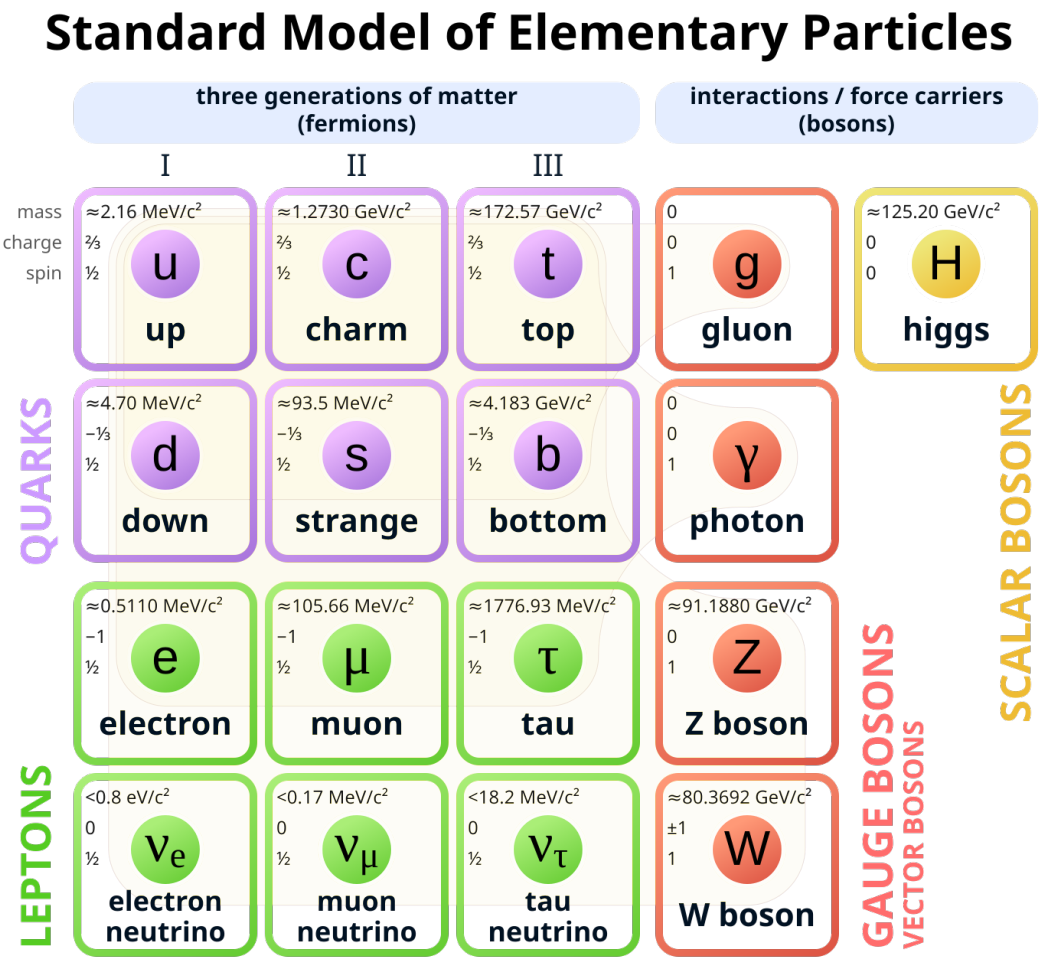

Figure 2.1: Overview of standard model [30]

Quarks can never appear alone in our universe. They either appear in a group of three (Baryons), or a quark-antiquark pair (Mesons). An example for a Baryon would be the proton (uud) and an example for a meson would be the π^+ meson ($u\bar{d}$). In each generation, there is a quark with a +2/3 and -1/3 charge. The main difference between the generations is their mass. The higher the generation, the greater the mass of all its particles.

Bosons are interaction particles that underlie the Bose-Einstein distribution, allowing identical bosons to be in the same quantum state in the same system. The bosons are split up into the gauge and scalar bosons. Each of the gauge bosons is responsible for an interaction of one of the fundamental forces. The photon (γ) is the carrier particle for the electromagnetic force, which couples to the charge of particles. The gluon (g) is the carrier particle for the strong force that couples to color charge, which is present in all quarks. The

Z and W Bosons (Z^0, W^+, W^-) are the carrier particles for the weak force and couple to weak isospin, which is present in all left-handed fermions and right-handed anti-fermions. The handedness of a fermion is related to the spin direction relative to the motion of the particles [29]).

The only scalar Boson is called the Higgs boson. The Higgs boson is the quanta of the Higgs field, which explains the intrinsic mass of all particles. When massive particles move through the Higgs field, they gain mass and thus exhibit inertia. Since the Higgs boson is a concrete prediction of the Higgs mechanism, its discovery was substantial.

2.2 The Belle II detector at SuperKEKB

The SuperKEKB is currently the only B-factory in the world. A B-factory is a type of particle accelerator that produces a large amount of B mesons and analyses its properties. At SuperKEKB electrons collide with positrons at an energy of 7 GeV and 4 GeV respectively. This results in a center of mass energy of roughly 10.58 GeV which is the resonance energy of the $\Upsilon(4S)$ bound state. With a probability of $\approx 96\%$ this decays into a $B\bar{B}$ meson pair and rapidly decays further into its decay products, the final state particles (FSP). These particles are then detected by the various sub-detectors of Belle II.

Originally B-factories were used to test and calculate CP-Violation, which is predicted by the CKM matrix [13]. The predecessor of Belle II, the Belle experiment, was one of the leading experiments that measured time dependant CP-Violation in B meson systems (calculated the ϕ_1 angle [10]) and resulted in a Nobel prize for the theorists that developed the CKM matrix. Due to the success of Belle, the upgrade of the accelerator and detector was granted. The new upgrade results in collision rates that are 30 times higher and will reach an integrated luminosity of 50 ab^{-1}, which is more than 50 times higher than its predecessor.

The Belle II detector is made from 5 main components, where the inner components are enclosed in a 1.5T magnetic field [6].

Vertex Detector (VXD)
The Vertex Detector is composed of two parts, the pixel detector and the silicon strip detector. These are used to reconstruct the tracks of charged particles emerging from the decayed B-mesons and sit very close to the beam pipe.

Central Drift Chamber (CDC)
The Central Drift Chamber is made from wire chambers filled with a Helium-Methane gas. Here the charged particles' charge and momentum can be determined, as the particle ionizes the gas and the resulting electrons give hits on the wire mesh. It also plays a crucial role in reconstructing the tracks for charged particles. Through the energy loss of the particles, it is also possible to use the CDC as a particle identification tool, using $\frac{dE}{dx}$.

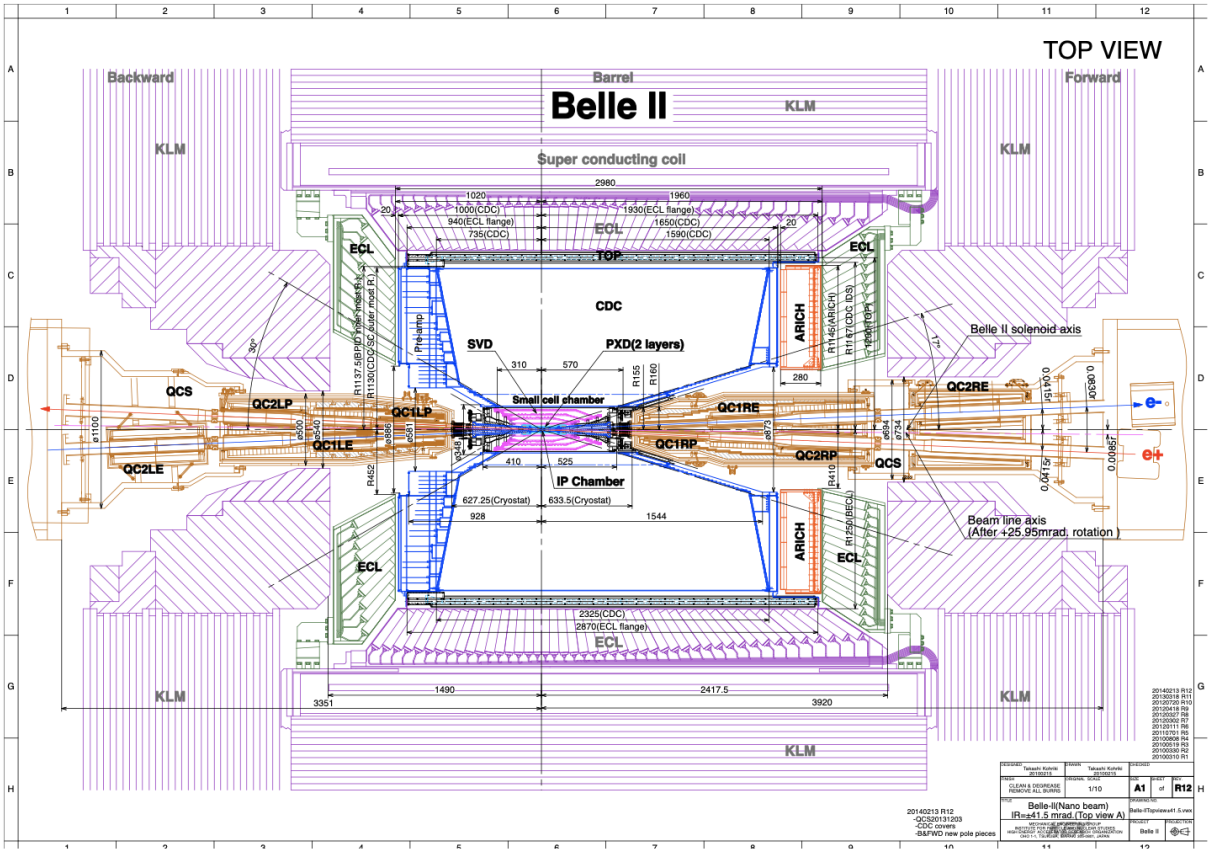

Figure 2.2: Belle II Detector [6]

Particle Identification (PID) with Arich and TOP

The Aerogel ring-imaging Cherenkov detector (Arich) is positioned in the forward direction, whereas the time of propagation counter (TOP) is barreled around the primary vertex and is made from quartz crystals. Both detectors are used to determine the velocity of the incoming particle by calculating the Cerenkov angle of the radiated Cherenkov photons. This information is then used for particle identification.

Electromagnetic Calorimeter (ECL)

The ECL is made from CsI(Ti)-crystals that are used for PID and to measure the energy from photons and neutral particles. When the high-energy particles hit the crystals, they produce particle showers which can be used to determine the particle's energy.

K_L^0-muon Detectors (KLM)

The KLM consists of alternating iron plates and detectors and is used for K_L^0 energy measurements and muon PID. Only K_L^0s and muons can traverse the detector for so long that they reach the KLM. Here K_L^0s can produce hadronic showers and muons can produce

tracks that can be measured.

The information provided by all these sub-detectors is then used to reconstruct all final state particles and their features (PID, charge, energy, etc.). However, there are some limitations in the precision of the measurements. Here are the main issues:

- Background particles can hit the detectors, which would then falsely be identified as particles originating from the primary collision.
- Neutrinos can not be detected at Belle II making it more difficult to fully reconstruct semi-leptonic decays.
- Kinematic features and PID can not always be accurately determined.

2.3 Software at Belle II

2.3.1 Monte Carlo Methods

In addition to the Belle II hardware, the software plays a crucial role in guaranteeing a successful analysis. Due to the messy and sometimes incomplete nature of the data provided, Monte Carlo (MC) methods are used to first develop and tune any analysis. Since the MC data is simulated, this allows the analyst to do his analysis on clean and complete data and learn a lot about the particular analysis. It also provides a way to compare the results from the detector to the theoretical expectations. These MC events are generated in 2 steps:

Event Generation
Given the initial features of the electron and positron in the beam pipe, collisions are generated based on various physics models. This could be basic SM physics or advanced theories like Supersymmetry (SUSY) or Dark Matter (DM). There are also different types of simulations used for analysis: "Signal MC" is used for specific decay channels, "Generic MC" for comparisons to the SM, and "Background MC" for excluding unwanted processes. There are various generators like EvtGen, KKMC, or Madgraph for different analyses and performance studies.

Detector Simulation
Once the features of all the particles in an event are generated, they are simulated so that they appear as if they were hits in the various Belle II detectors. Software like Geant4 simulates interactions with a virtual detector, producing energy deposits and particles in each sub-detector. Then custom software uses the output from Geant4 and turns it into signals that mimic actual detector outputs. Simulating the full detector is computationally intensive and takes about a second per event at Belle II and up to minutes for higher energy experiments like LHC [7].

After the detector simulation, the data is similar to real detector outputs and allows for comparisons with actual measurements. Geant4 uses an idealized model, as the real detector, with its complex hardware, can't be perfectly replicated in the simulations and small deviations to real output are to be expected.

2.3.2 Reconstruction

After acquiring data or running simulations, the raw detector responses are processed to reconstruct the original FSP and their four vectors as accurately as possible. Identifying all particles is difficult, due to short-lived particles decaying before detection, intrinsic noise, and background signals. The detector responses are analyzed to determine the most likely particles and then perform a statistical analysis. The same reconstruction algorithms are used for both, real and simulated data. However, the advantage of MC data is the certainty about the true type and features of the simulated particles. This process is known as "MC truth matching" [7].

The reconstruction of the FSPs is done in the following steps [4]:

Clustering
Clustering combines responses from several sub-detectors and groups them if they are related. For example, several neighboring pixels in the PXD can be hit by one particle. Then by taking the average, weighted by the deposited charge, the center of the cluster can be determined. Noise and other intrinsic properties of the PXD are also taken into account when clustering. The same is true for clusters in the ECL, where especially photons can be reconstructed.

Tracking
Tracking involves finding fitting patterns of hits in the tracking detectors to determine the trajectories of the particles, and estimating their positions and momenta near the interaction region.

Particle Identification
Using various sub-detectors, energy loss and Cherenkov rings are determined for each particle. These are then compared with the expected values for different particle types and calculate a likelihood prediction for the type of particle for the given track [1].

This process reconstructs the final state particles (FSPs) and their four-momenta. To reconstruct intermediary particles, the four momenta of the FSPs are summed and checked for resonances in the invariant mass. If the combination matches a known intermediary particle, that particle can be reconstructed. There is also another class of particles that have to be combined using a different method called V0. These are neutral particles that decay into two charged particles far away from the primary interaction vertex. This decay leads to the typical V-shaped signature and therefore gives a clear indication of which two

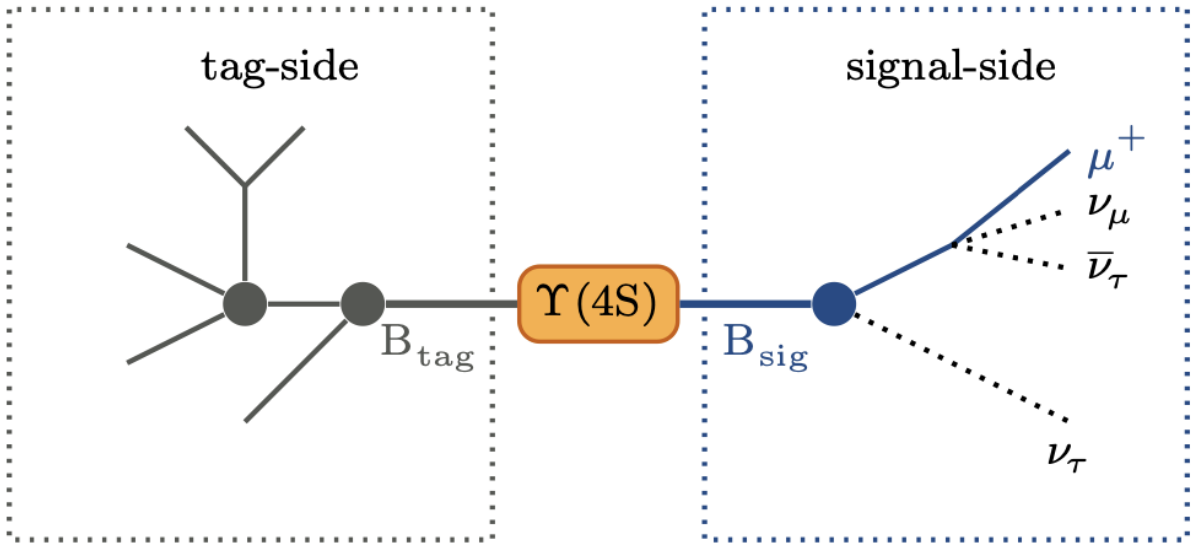

Figure 2.3: Schematic overview of $\Upsilon(4S) \rightarrow B\bar{B}$ decay chains split into tag- and signal-side [15]

particles are its daughters. The combination of FSPs and intermediary particles can then be continued up the decay tree. The software framework for the reconstruction of FSPs and the combination of them is called Basf2 and was developed exclusively for Belle II [3].

2.4 Tagging B Mesons

One of the main missions of the Belle II experiment is to find rare decays and measure their frequency to probe the SM. Especially decays like $B \rightarrow l\nu\gamma$ that include neutrinos can't be directly detected at Belle II. To address this challenge the concept of tagging is used. [15]

As explained in earlier chapters, the primary collisions produce two B mesons. In events where the signal is present, the mesons can be split into two sides, the tag (B_{tag}) and signal (B_{sig}) side. The four momenta of the $\Upsilon(4S)$ resonance is precisely known, and no additional particles are produced in the primary collision. Therefore if the properties of the B_{tag} are calculated, all the information about the B_{sig} can be determined. This is because if the B_{tag} would be completely reconstructed all the remaining FSPs can be assigned to the signal side.

Figure 2.3 shows a schematic overview of an example decay tree, split into the signal- and tag-sides. In reality, the FSPs of each of the B mesons are not spatially separated on two sides, and more complex algorithms have to be used. This thesis describes an approach to

how this could be done using ML techniques.

2.5 The Full Event Interpretation (FEI)

A complete reconstruction of an event requires accounting for all tracks and clusters for accurate data interpretation. The Full Event Interpretation (FEI) is an exclusive tagging algorithm that was developed for Belle II and is integrated into the Belle II analysis software [15].

This means that the algorithm constructs possible decay chains for the tag side that are compatible with the observed tracks and clusters. "Exclusive" means that only specific decay channels can be taken into consideration when reconstructing the B_{tag}. These are the hadronic and the $B_{tag} \to D^{(*)}l\nu$ decay channels. Both of these channels can be easily identified leading to a high purity in the tagging, but also to a low efficiency as all the other tag side decay channels have to be discarded.

The FEI follows a six-step hierarchical approach in reconstructing the event, as shown in Figure 2.4. As input, it takes the detector information in the form of tracks, clusters, and displaced vertices as explained in Chapter 2.3.2. It then reconstructs final state particle candidates and combines them to form intermediary particles up until the B meson. For each FSP and each decay channel of the intermediary particles, a separate multivariate classifier is used that takes the particles' input features and provides a probability estimate of it being correctly classified. If the user needs very pure events, he can set a threshold value for this probability score to be very high, at the cost of losing more efficiency. The multivariate classifiers are in the form of boosted decision trees and are trained on MC data.

When the tag side is then correctly reconstructed to high accuracy, the remaining clusters combined with the knowledge of the $\Upsilon(4S)$ resonance can then be used to fully reconstruct the event. This makes it possible to calculate the branching fraction of exclusive decays like $B_{sig} \to \tau\nu$ and inclusive decays like $B_{sig} \to Xl\nu$

The main issues of FEI are however the large amount of data that has to be cut out, specifically only $\mathcal{O}(10000)$ decays can be considered resulting in 85% of all events having to be discarded. Also, the stage-wise approach to full reconstruction, which relies on domain-specific optimizations at each step, is prone to error accumulation [28].

To optimize FEI, Lea Reuther et al. used a machine learning architecture called a Graph Neural Network (GNN) (Explained in Chapter 3) to completely reconstruct the tag side from scratch. This algorithm is called GraFEI. Since this thesis uses similar techniques to GraFEI, here is a short explanation of its functionality and the idea [24].

GraFEI requires the reconstructed FSPs and their features as input for its network. These

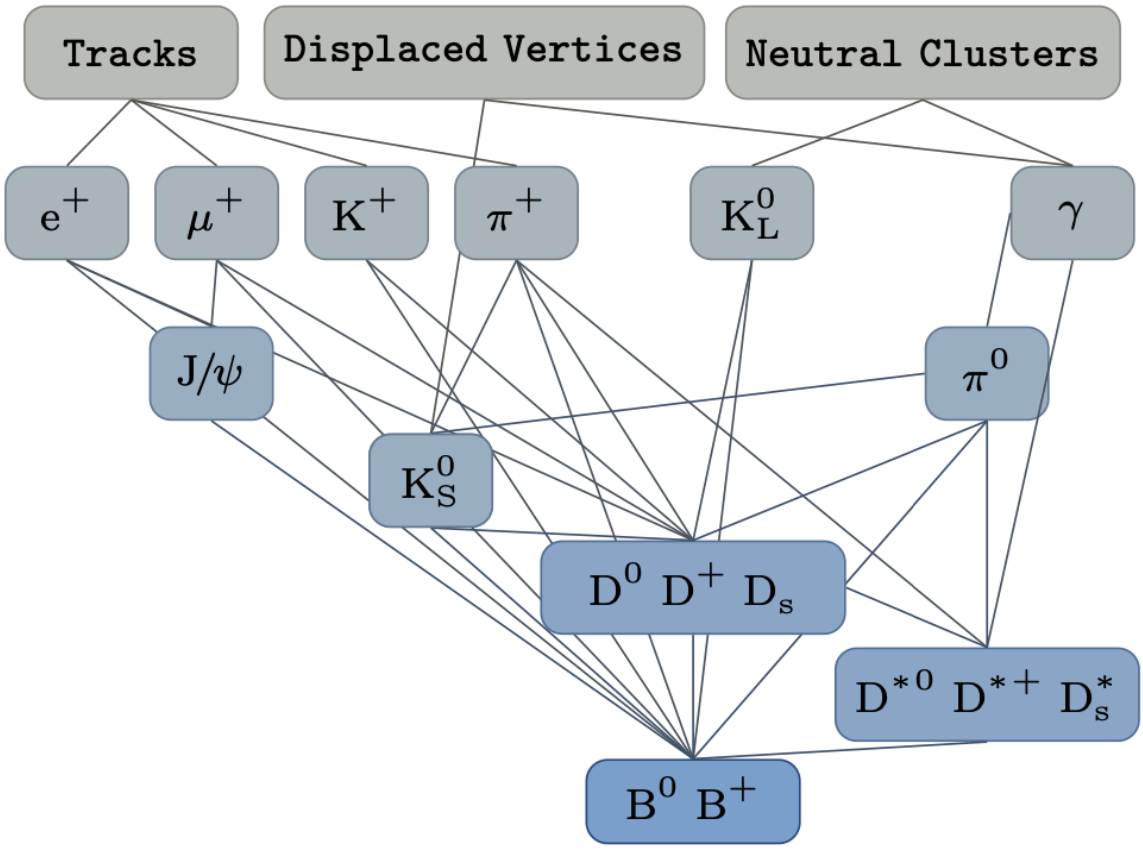

Figure 2.4: Schematic overview of multi-step reconstruction approach of FEI for an example event [15]

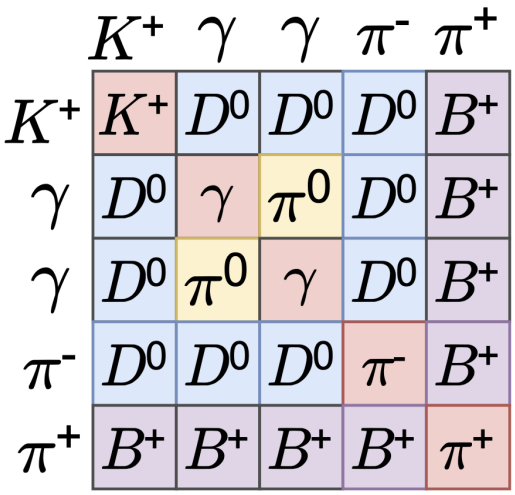

Figure 2.5: Sample LCA matrix for $B^+ \to \bar{D}^0(\to K^+\pi^-\pi^0(\to \gamma\gamma))\pi^+$ [24]

are then placed in the GNN and a lowest common ancestor (LCA) matrix is given as an output. An LCA matrix displays the closest mother particles for each of the FSPs, an example of this is shown in Figure 2.5 for the $B^+ \to \bar{D}^0(\to K^+\pi^-\pi^0(\to \gamma\gamma))\pi^+$ decay chain. From this output, the complete decay channel can be reconstructed. Here all of the possible decay channels could technically be taken into consideration and it is also an end-to-end approach. This network was however only trained on a toy dataset of only hadronic decay channels.

Chapter 3

Machine Learning

In recent years, the field of Machine Learning (ML) has revolutionized nearly all global industries and research areas, including High Energy Physics (HEP). Although ML had already been utilized in HEP, the recent surge in interest and advancements in the field have significantly impacted HEP as well.

As neural networks will be used in this thesis, chapter 3.1 will explain the basics of neural networks. To better visualize the structure, an example of a simple feed-forward neural network performing a classification task is described. Chapters 3.2 and 3.3 will cover graph neural networks and transformers, respectively, as these are the architectures used in this thesis. These are both more advanced forms of deep learning but are based on the idea of the simple feed-forward model.

3.1 Neural Networks and Deep Learning

Neural networks (NN) are a type of ML program designed to teach a computer to process data and make decisions in ways that mimic human decisions [18].

Every NN is made from connected nodes, that loosely model the neurons in a brain. Each node receives input from connected nodes, processes the information, and sends it out to the next nodes. The information that is calculated in these nodes is just numbers, one number per node.

As shown in Figure 3.1, a NN is made from multiple nodes, each aligned in layers. The first is the input layer the last is the output layer and the middle ones are the hidden layers. The number of hidden layers and the number of neurons per layer can be adjusted to best suit the task of the network. All tunable parameters of a network are called hyperparameters and play a fundamental role when optimizing these networks. In this simple, so-called feed-forward NN, each neuron from the previous layer is connected with all neurons from the next layer, passing the information through the network. But how does the passing of

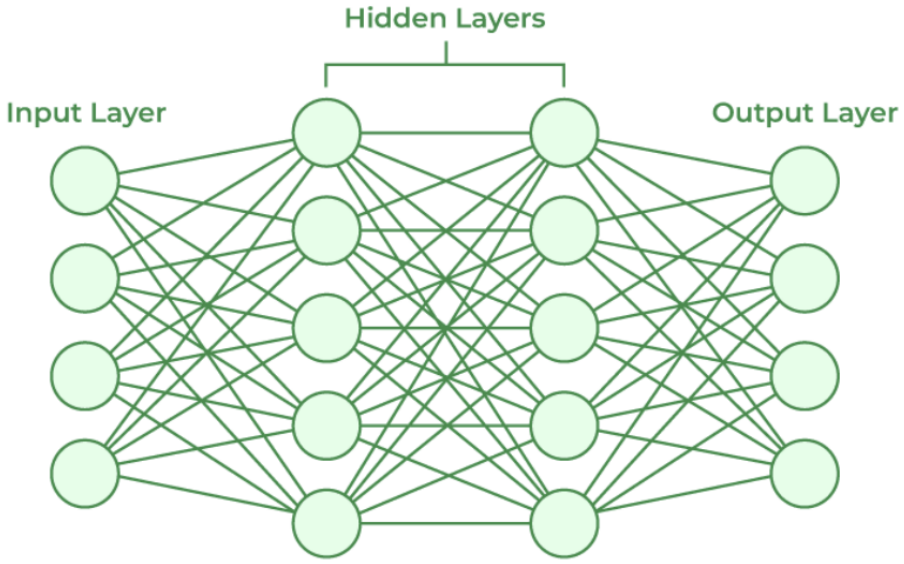

Figure 3.1: Labeled model of a simple feed-forward neural network [17]

information work explicitly?

To better understand the learning and information-passing process, let's look at an example that is often used when first explaining NN. In this example, a user has a set of 10,000 grey-scale pictures that contain 28x28 pixels. In each of the pictures, a number between 0-9 is drawn. Each pixel of the image will have values between 1 (black) and 0 (white). In Figure 3.2 examples of these pictures are displayed. For humans it is easy to identify each of the numbers displayed in each of the pictures, but not for an untrained NN.

First, the data will be split into 2 categories, training and validation data. Training data

Figure 3.2: Sample dataset used for classification of numbers [27]

will be used to train the model and validation data to see how well the network performs on previously unseen data. The training data will then be batched in multiple sets of data, in this case, multiple pictures, and the run through the network. Every time a whole batch has run through, the network parameters update and it learns.

When initiating a new, untrained NN each of the connections between the nodes is randomly given a number called a weight (ω_{ij}), and each neuron (except input neurons) is also given a random value called a bias (b_i). These are the training parameters of the model. The neural network will need 784 (28 x 28) nodes in the input layer, as each node can only hold one value. The information from one layer to the next is passed according to

$$x_j^{(2)} = f\left(\sum_i^n (x_i^{(1)}\omega_{ij} + b_j^{(2)})\right) \tag{3.1}$$

Where $x_i^{(1)}$ is the ith node in the first layer, $b_j^{(2)}$ is the bias of the jth node in the second layer and ω_{ij} is the connection weight between the ith node in the first layer and the jth node in the second layer. The function $f(x)$ is called the activation function and traditionally is used to turn neurons in the NN on and off. There are many different activation functions and some work better than others for certain tasks. The ones used in this paper are the sigmoid and the LeakyRelu functions that are defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

$$\text{LeakyRelu}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.1x & \text{otherwise} \end{cases} \tag{3.3}$$

This process is repeated for all nodes in all the layers following the input layer. The output layer consists of 10 neurons, each representing one of the 0-9 possible outputs. After the information of one picture is run through the network, each of the output nodes will have a value. The node with the highest value will be the prediction of what number it saw in the picture.

Of course, when first initializing a network with random weights and biases, the network will probably not give the correct prediction at first. The model will have to be trained. This is done by a process called back-propagation. First, we have to define a loss function L, which can be chosen depending on the type of model. This loss function gives the network a score on how well it predicted the input, the lower the score, the better the network's performance. There are many different types of loss functions and they have to be chosen depending on the task of the network. Essentially what is done to "train" a

model, is to tune all the training parameters. To train these parameters, the derivative of the loss function w.r.t. the desired parameter is calculated according to

$$\delta_{ij}^{(n)} = \frac{\partial L}{\omega_{ij}^{(n)}} \qquad \omega_{ij,new}^{(n)} = \omega_{ij,old}^{(n)} + \eta \delta_{ij}^{(n)} \tag{3.4}$$

Generally, it is not that easy to calculate the derivative, so we calculate the derivative for each step made through the information passing (e.g. activation output of layer l and weight of layer $l-1$). This is done using the multivariate chain rule.

$$\frac{dL(f(x))}{dx_i} = \sum_j \left(\frac{dL}{df_j} \frac{df_j}{dx_i} \right) \tag{3.5}$$

Where $f(x) = f(x_1, x_2, ...)$ is a vector-valued function taking input vector x. The η is called the learning rate parameter and this can be tuned. These updates are made every time a batch of data is passed through the network.

Then when all batches of training data run through the network, the validation data is passed through the network to check how well it performs on unseen data. Then the training data files are scrambled and run through the network again. The amount of times the same data is run through the network is called an epoch. Generally, if the network does not experience any large improvements anymore the training will be stopped, therefore each network type has a different amount of epochs required for training.

Another important aspect of machine learning is the concepts of overtraining and vanishing gradients [14]:

Overtraining

Overtraining in ML is the process of the network learning details and noise of the training data that negatively impacts the performance of the network. This often occurs when the network is too complex for the task at hand. To counteract this either the hyperparameters have to be tuned, or one could introduce a dropout. A dropout means that for each batch during training a random amount of nodes are just left out when doing a pass through the network. This helps the model understand the general trends of the data and will be forced to ignore specific small variations in each of the data inputs. Then during evaluation, all the neurons are used again but their outputs are multiplied by (1 - dropout) to effectively have some kind of averaging effect.

Vanishing gradients

Vanishing gradients occur when the network has too many layers so that when performing backpropagation the updating of the trainable parameters near the input layer, updates very slowly. A simple example of this would be, if the network has the task of finding the trend line in a scatter plot, and the trend is linear. Then if you give the network more than two degrees of freedom to fit the trend curve, it will not find the simple trend, but overcomplicate it so that the averaged trend line is heavily influenced by statistical fluctuations in the data. There are however methods to mitigate this, explained in later parts.

3.2 Graph Neural Networks (GNNs)

One of the network architectures that are used in this thesis is Graph Neural Networks. The advantage of such a network type is that the input size can vary and the output is invariant under permutation of the input nodes. These are all features that are also true for a particle detector event.

A GNN consists of nodes and edges. Unlike to the feed-forward network, the nodes and edges can hold a vector of features instead of just single numbers.

Figure 3.3 shows an example of a GNN. Here the nodes represent particles of an event and the edges are the connection of two particles. The nodes hold lists of kinematic features of the various particles and the edges contain the invariant mass and the cosine angle between two connected ones. The user of the network can freely choose what nodes are connected and in which direction they act when training. In the case of particle physics, this is mostly chosen depending on the particle type or features, e.g. two positively charged particles will most likely not come from the same decay branch and therefore do not necessarily have to be connected.

GNNs train very differently to feed-forward networks. Instead of passing information from one input layer to the output layer, each node gets updated by receiving information about its connected nodes and edges. Figure 3.4 shows the training process of a GNN. Here the ReLu function is the activation function of the network.

As the size of the feature vector for the nodes and edges is generally very small in the input layer, they often get increased to a larger size for the hidden layers. There are several different ways to perform the information passing in GNNs, depending on the type of task the GNN should perform. The nodes and edges in the output layer can also have different forms. One possibility would be to classify the nodes into two or more classes. This could be done by applying a final weight matrix on each of the node vectors, that would compress each node value into a single number. Depending on the number of all the

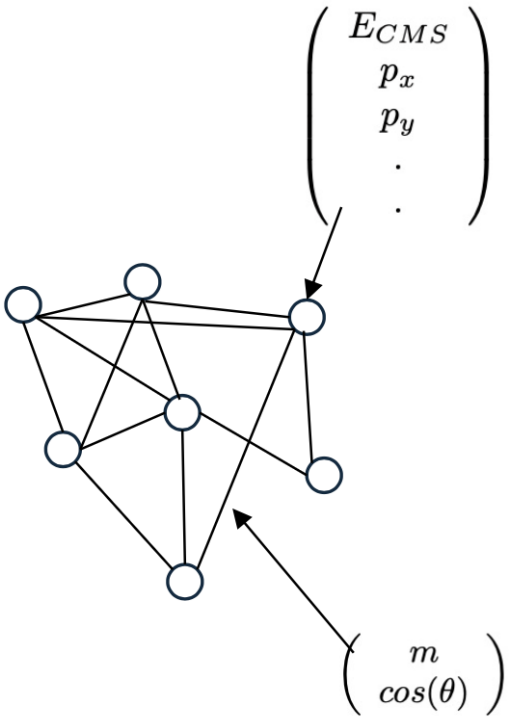

Figure 3.3: Schematic view of sample GNN with edge and node features

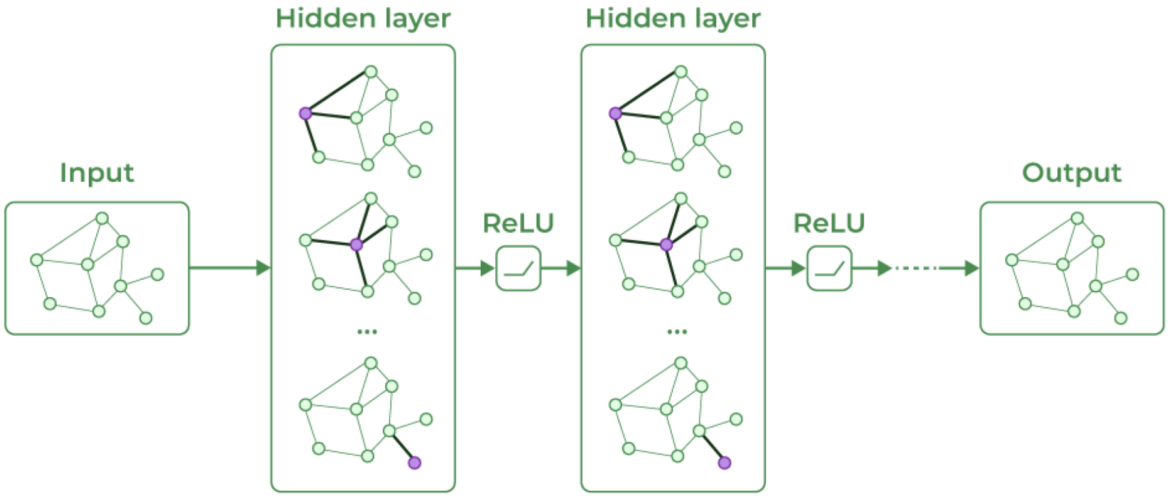

Figure 3.4: Schematic view of information passing in GNNs

particles, these could be clustered into separate groups, which is done in the first part of this thesis in Chapter 4. There the network used for this thesis is also explained in more detail and its message passing between layers is explained.

The training process is then similar to the one in the feed-forward network. A loss function will be defined that provides a performance score of the network for each batch, and then through back-propagation, each of the trainable parameters can be updated.

3.3 Transformers

The transformer model was introduced in the famous paper "Attention is all you need" by Vaswani et al. in 2017 [16]. It revolutionized machine learning especially when it comes to natural language processing. The most famous network using a transformer architecture is GPT-4. As the name of the paper would suggest the key ingredient of a transformer is the attention mechanism. On a high level, the attention mechanism can differentiate how important various parts of the input data are to each other. This gives it the ability to isolate important parts and assign more weight to them in the process of determining the output.

A general transformer is comprised of multiple layers of encoders and decoders that are stacked on top of each other. Figure 3.5 shows an example of a transformer architecture. Here are explanations for all of the sub-components:

Embedding - The input data first gets embedded into a vector of fixed size.

Positional encoding (if required) - This ensures that the embedding vector also reflects the position of each data part in the input sequence. This part is only relevant if the order of the input sequence is relevant, which is not the case in particle physics.

Multi-head self-attention mechanism - Multiple attention mechanisms are performed on the input data. Each of these "heads" tries to identify a different aspect of how the input data is connected.

Skip connections - In this part, the output from the previous component is concatenated and normalized with its input, but can be placed in different parts of the network, depending on the implementation. This tries to mitigate the vanishing gradient problem.

Feed forward NN - After this, the output is passed through a feed-forward network.

Important to note that Figure 3.5 shows a transformer model that was used for natural language processing, and therefore feeds its output back into the decoder block to get the next output.

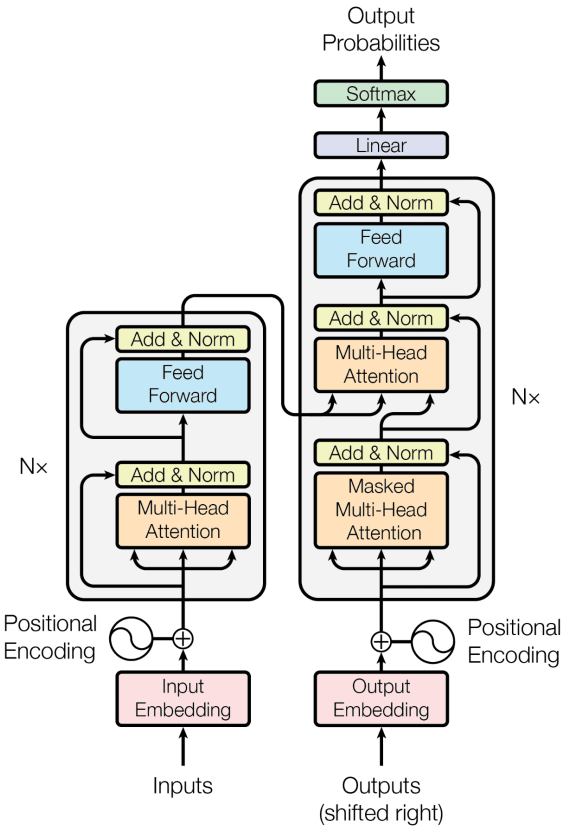

Figure 3.5: Basic Transformer Model (Encoder - left grey box, Decoder - right grey box [8]

The attention mechanism works by first transforming the embedded input vectors into three different matrices called Query(Q), Key(K), and Value(V), using trainable weight matrices [16]:

$$Q = XW^Q \qquad K = XW^K \qquad V = XW^V \tag{3.6}$$

Where X represents the embedded input vectors of one batch. W^Q, W^K & W^V are the trainable weight matrices.

To calculate the Attention matrix, the following formula is used:

$$\text{Attention(Q, K, V)} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.7}$$

The Softmax function is defined by:

$$\text{Softmax}(z)_i = \frac{z_i}{\sum_j^k (z_j)} \tag{3.8}$$

where z is a k-dimensional vector.

To then get a multi-headed attention matrix, the simple attention matrix is calculated multiple times but with different W^Q, W^K & W^V matrices. Each head gets a new set of these trainable Query, Key, and Value matrices. Then the multiple attention matrices are concatenated and multiplied by yet another weight matrix W^O:

$$\text{MultiHead(Q,K,V)} = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{3.9}$$

All of these trainable matrices are trained using backpropagation.

Chapter 4

Clustering particles using GNNs on MC truth data

Although the objective and motivation of this thesis were mentioned a couple of times in previous chapters, here is a quick summary.

Due to the difficulty of detecting rare decays that include neutrinos, the tagging method was developed, which tries to constrain the possible B_{sig} decay products by finding all tracks and clusters of the B_{tag} meson. In FEI this is done by choosing easily identifiable decay chains for the tag-side and reconstructing them fully in a hierarchical manner, where each decay has its own boosted decision tree. GraFEI tries to reconstruct the tag side fully with an end-to-end approach. The network also tries to build up the decay chain from the FSPs, however, this is done within one network, thereby limiting the issues caused by build-up errors in FEI.

This thesis will try to classify the particles directly without the need to reconstruct full events. The network will try to classify each of the particles of an event into two groups, each belonging to one of the B mesons. We will call these groups clusters in later parts of the thesis. In GraFEI there is an inductive bias since we tell the network that it's supposed to solve the problem by explicitly reconstructing the decay tree. Here with this model, we try to remove this bias and let the network figure out how to cluster the particle groups on its own, without any predetermined approach. This could make it more difficult for the network at first as it needs to find a way to cluster the particles, but could be a more powerful approach when given enough data. As the network does not have to reconstruct the full decay tree, we can include all decay channels assuming a successful $e^-e^+ \to \Upsilon(4s) \to B\bar{B}$ primary collision.

As the structure of a particle physics event matches that of a GNN network, it was logical to choose this ML architecture for the classification task. This chapter will explain the specific GNN architecture utilized, the chosen input parameters, and the type of MC data employed. Subsequently, the performance of the network will be discussed in detail.

4.1 MC Training Data

The raw data in this thesis are samples produced by the Belle II collaboration in the MC15 campaign[7]. Charged meaning that all events had a primary $\Upsilon(4s) \to B^+B^-$ decay. For this part of the thesis a total of 6.48 million events were used for training, split up into 36 files with 180k events each. For each of these events, the FSPs are extracted with all of their features on the MC truth level. As explained in Chapter 2.3, because the particles are simulated, each of the particles and their features are completely known. This form of data was used as a proof of concept, to see how good such a network might perform without any noise and with the idea to add background, a detector simulation, and reconstruction in later stages if the network proved to be successful.

To keep the number of input particles from becoming too big, not all FSPs were used in the training data but rather a combination of intermediary and final state particles were used. The particle types that where used, are: $\pi^0, \pi^+, p^+, K^+, e^-, \mu^-, K^0_S, \Lambda^0$ and their corresponding antiparticles. The most notable final state particle missing from this list is the γ. As the initial network setups showed bad performance when using photons directly, π^0s were used instead. This could be done, because nearly all gammas come from decayed π^0s. When constructing all of these particles, it was ensured that none of the energy was counted multiple times. For example, when a $\Lambda^0 \to p^+e^-$ decay occurs in an event, only the Λ^0 particle and its features were added to the training sample, and the p^+ and e^- were removed. With these types of particles selected and no background present, the average event has ~ 16 particles.

Each of the particles also has a list of features. The features that are used for the network are the particle energy in the center of mass (CMS) frame, momentum in x, y & z direction in the CMS frame, the particle charge, the PID of the particle, and a label that shows if the particle derives from the one or the other cluster. The label used as a training target provides each particle with the value zero or one depending on which cluster they belong. The charge tag gives +1, 0, or -1 values for the particles and the momenta and energy are calculated in units of GeV/c and GeV respectively. Finally, each particle carries its PID information in the form of an 8-dimensional vector, where each entry represents one particle type.

The only edge feature in the GNN is the invariant mass of the connected particles. This is calculated for each particle pair in the prepossessing of the data:

$$m_{inv} = \sqrt{E^2_{total} - (p^2_{x,total} + p^2_{y,total} + p^2_{z,total})} \tag{4.1}$$

Where total refers to the sum of momenta and energy of both connected particles.

4.2 GNN Architecture

This network will work as a node classifier, as each node will have a single value in its output, zero or one. All the nodes with zeros and all the nodes with ones will be classified as originating from one of the two B-mesons.

The GNN architecture of the model used in the thesis evolved over many trial-and-error steps, from tuning hyperparameters to changing input dimensions. The following is the network structure of the best-performing model.

As mentioned in chapter 3.2, a GNN consists of nodes that are connected by edges, where each of the nodes contains a list of feature vectors for each particle and the edges contain features that come from both of the connected particles. As explained in Chapter 4.1, the node features are the energy, momentum (in CMS), the charge of the particles, and their particle type. The edge feature used is the invariant mass of the connecting particles. The idea behind the invariant mass is that the network can find resonance energies of the two connecting particles, making it easier to cluster some of the particles together in advance.

The message-passing mechanism in this network follows a graph attention approach, which aims to determine the importance, or attention score, of neighboring nodes. This score helps weigh the contribution of each neighbor during the node update process [9].

$$f'_{ij} = \text{LeakyReLU}(A[h_i \parallel f_{ij} \parallel h_j]) \tag{4.2}$$

$$e_{ij} = \vec{F}(f'_{ij}) \qquad h'_i = \sum_j^N (\text{softmax}(e_{ij}) W h_i) \tag{4.3}$$

Here, f_{ij} represents the edge features between nodes i and j, A is a trainable weight matrix, h_i denotes the features of node i, $\vec{F}$ is a trainable weight vector, W is another trainable weight matrix, and the prime ($'$) indicates values for the next layer.

Node-to-Edge Update
Each edge is updated by utilizing the edge features from the previous layer along with the features of the connected nodes. These features are concatenated and multiplied by a trainable weight matrix A, and the result is passed through the LeakyReLU activation function (equation 4.1).

Edge-to-Node Update
In this step, the updated edges are multiplied by a trainable weight vector $\vec{F}$, resulting in a one-dimensional attention score. This attention score is then passed through a softmax function (equation 3.8) over all edges connected to the node being updated. We then do

a weighted sum, provided by the attention scores, of the trainable weight matrix W multiplied by the node features of the previous layers (equation 4.2).

This GNN network also uses heads in the same way transformers use them. The message passing happens simultaneously for each head, where each head has different weight matrices. Then the node and edge feature vectors for each of the heads are concatenated and reduced using another trainable weight matrix.

Every layer has its own set of parameters for the weight matrices and vectors.

After message passing has been completed for each hidden layer in the network, the final node representations are passed through a linear layer to produce a one-dimensional output. This output is a value between 0 and 1, where a score close to 0 indicates high confidence that the particle belongs to cluster 0, and a score near 1 suggests strong confidence that it belongs to cluster 1. Now the label that is saved in the MC truth data will be used to calculate a loss.

The loss function used in this network is called cross-entropy loss, and is very common when having some form of classification task [21]

$$L(x, y) = \frac{1}{N_p} \sum_{i=1}^{N_p} (y_i \log(x_i) + (1 - y_i) \log(1 - x_i)) \tag{4.4}$$

Where N_p is the number of particles in a batch, x is the output score (a number between 0 and 1) and y is the label of each particle (either 0 or 1).

As it is ambiguous to label the particles as one or zero, and the only thing that matters is the clustering of the particle groups, the best assignment loss is used to correctly calculate the loss function. To do this, the loss is calculated twice, one time shown in equation 4.4, and once by changing the labels for each particle to its opposite ($0 \to 1$ & $1 \to 0$). We can define these labels $y^{(1)}$ for the regular labels and $y^{(2)}$ for the flipped ones. Then the loss with the lower value (better score) is chosen for every event, and the network trains on the averaged loss for an entire batch.

$$L_{k,j} = \sum_{i=1}^{N_j} (y_{i,j}^{(k)} \log(x_{i,j}) + (1 - y_{i,j}^{(k)}) \log(1 - x_{i,j})) \tag{4.5}$$

$$L_{final} = \frac{1}{N_p} \sum_{j=1}^{N_e} \min(L_{1,j}, L_{2,j}) \tag{4.6}$$

Where N_j is the number of particles in the event and N_e is the number of events in a batch. For these formulas, i is the index for the particle in an event, and j is the index for the event in the batch.

Then backpropagation is used to update the weight matrix elements as explained in Chapter 3.

The specs and hyperparameters used for this network are summarized here:

- Node input features (13 total): E_{CMS}, $p_{x,cms}$, $p_{y,cms}$, $p_{z,cms}$, $charge$, π^0_{PID}, π^+_{PID}, p^+_{PID}, K^+_{PID}, e^-_{PID}, μ^0_{PID}, $K^0_{S,PID}$, Λ^0_{PID}
- Edge input feature (1 total): m_{inv}
- 6 hidden layers
- 2 heads
- 512 node features in hidden layers
- 128 edge features in hidden layers
- Batch size: 256
- 6.48M events
- Dropout in final layer: 0.1
- Optimizer: Adam [20]
- Learning rate (η) = 1×10^{-3}
- Edges between all pairs of particles except for pairs of K^+

4.3 Results of the GNN

To quantify the performance of the network, the loss score, particle accuracy, and event accuracy are calculated.

Loss Score - The loss score is calculated for each batch and then averaged over all batches in a file

Particle Accuracy Score – The network output for each particle is rounded to 0 or 1, based on whether the node output x is less than 0.5 or greater than or equal to 0.5, respectively. For each event, the rounded network output is compared to the true label,

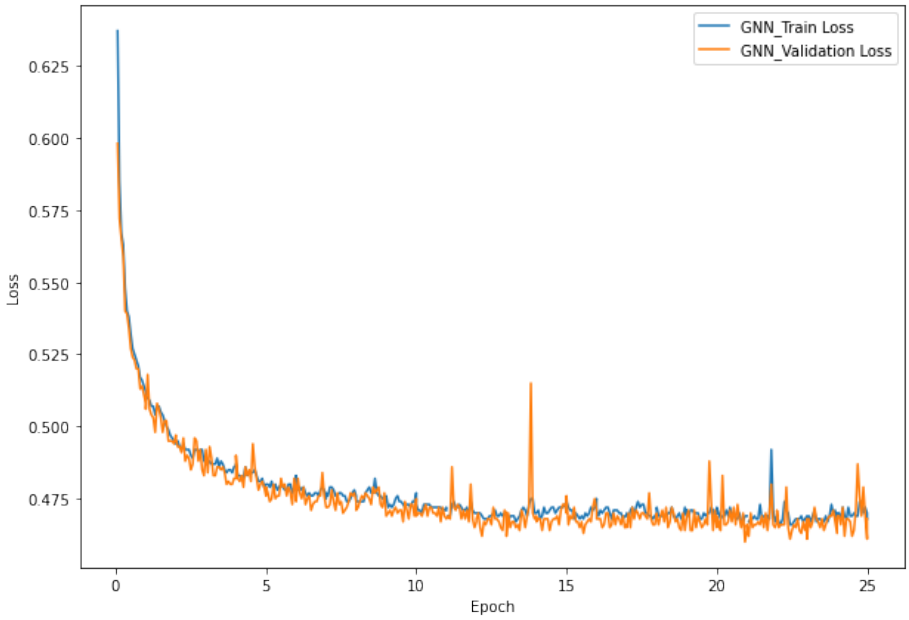

Figure 4.1: Training of GNN - Average loss reducing after training over some epochs

and the proportion of correctly matched particles is averaged across the event. Since the labels 0 and 1 are interchangeable and only the clustering of particles matters, the best assignment accuracy is computed, following the same logic as the best assignment loss. While one might expect the accuracy of an untrained network to be around 50%, the best assignment accuracy yields a default accuracy of approximately 59% for an untrained network.

Event Accuracy - This parameter calculates the amount of perfectly classified events, so if the rounded classification output for all particles matches their OldestAncestorId tag. Here the best assignment accuracy method is employed again, for the same reasons as before.

Figures 4.1, 4.2, and 4.3 show the training results for this GNN. Each of the 36 files per epoch creates a data point in each of the figures. Here the top particle accuracy achieved for one event file was 74.1%. Two curves are sketched in Figures 4.1 and 4.2, one for the validation data, and one for the training data. Here it was structured so that 10% of each data file was the dedicated validation data and the rest was training data.

Figure 4.3 shows how the perfect event accuracy evolves through training, which peaked at 9.8%. This seemed odd at first, as the naive approach to estimate the perfect event accuracy would be $(76\%)^{16} = 1.2\%$, where 16 is roughly the average amount of particles in an event. This must mean that the predictions are correlated, and some decay trees are easier fully identified than others.

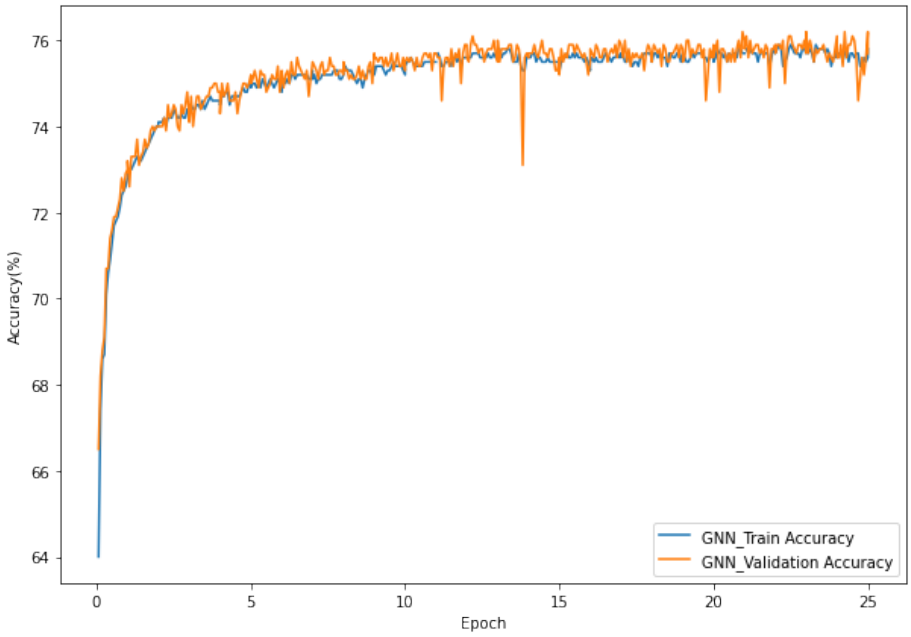

Figure 4.2: Training of GNN - Average accuracy increasing after training over some epochs

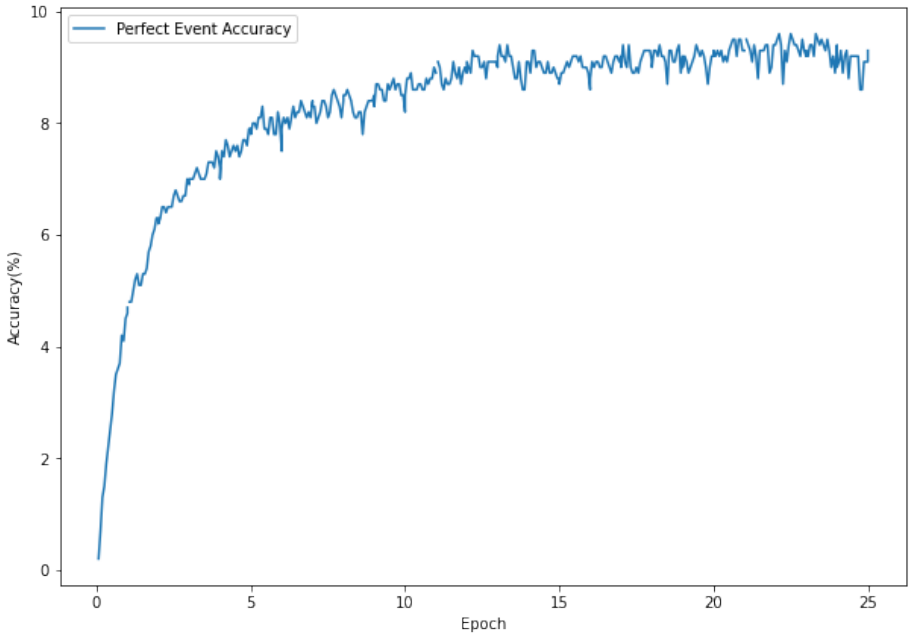

Figure 4.3: Training of GNN - Average perfect event accuracy increasing after training over some epochs

Particle type	Prediction accuracy
π^+	71.3%
p^+	94.6%
K^+	78.6%
e^-	82.7%
μ^-	86.7%
π^0	69.3%
K_S^0	72.8%
Λ^0	96.3%

Table 4.1: Predicted classification accuracy for GNN model (Total accuracy of 74%)

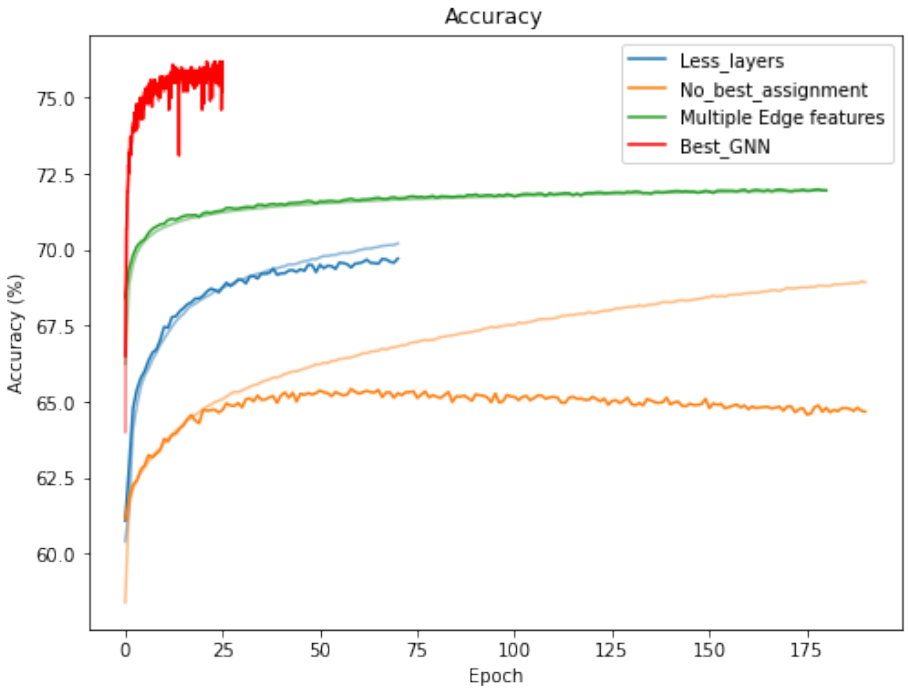

Figure 4.4: Training of the network using the particle accuracy metric for different GNN model types. Transparent lines are training data, solid lines are validation data

To further investigate this, the predicted particle accuracy for each particle type is listed in Table 4.1. The total average prediction accuracy of this data is 76%. It is important to note that the average of all the individual prediction accuracy does not average up to 76%, as some underperforming particles like π^0 or π^+ are much more common in the events. This list might suggest that some specific events, e.g. if there is a Λ^0 present, are easier identifiable by the network.

A big part of finding the best working hyperparameters and input features was done through trial and error. Many different ideas were tested and put to work. The changes that resulted in the largest impact were the use of best assignment loss, connecting all nodes with edges, and increasing the complexity of the network. Figure 4.4 shows how the network operated before these features were implemented. In the beginning, the network could only reach an accuracy score of $\sim 69\%$ and that was with massive overtraining (orange line). Then more data was used the best assignment loss was introduced, and an accuracy of $\sim 70\%$ was reached (blue line). Then by making the network more complicated, by adding more layers and heads we reached $\sim 72\%$ (green line). Another big jump was reached by changing the input features for the edges and switching to an attention-based message-passing algorithm (red line), which is described above in Chapter 4.2.

4.4 Alternative GNN Architectures

The network presented in this chapter was the best working out of all GNN architectures investigated. Alternative model types were also tested for the same classification task. Here are two main alternative GNN model architectures that were tested, but ultimately performed worse.

Edge Classification

As not only the nodes are updated in a GNN, but also the edges, one idea was to cluster the particles together by edge classification. This means that not all the final node values are the outputs, but each of the edge connections outputs an integer value between 0 and 1. The idea was that all the edge outputs between particles that belong to the same cluster would be high, and the edge outputs would be low when the two particles are in different clusters. Not all the details of the network are listed here, the structure of the network was however very similar to node classification.

After training 4 million events, of the same data type as explained in chapter 4.1, over 50 epochs the trained network got a loss score of ~ 0.573, and an accuracy of $\sim 63.7\%$. In this network, the best assignment loss could not be used, therefore the loss and accuracy scores of this network can't be directly compared with the prior one. Nevertheless, both the accuracy and the loss scores are significantly worse than those of the node classification model.

Contrastive loss

Rather than classifying the nodes by giving them integer values, a common practice is the use of contrastive loss [19]. For this, each node is given a vector output rather than an integer value and plotted in a multidimensional embedding space. Then in this space the particles that are from the same cluster, get pulled together and particles that are from different clusters are pushed apart, using the following loss function:

$$L = \sum_{i \in I} -\log \left(\frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)} \right) \tag{4.7}$$

Where I denotes all the particles in the event, $|P(i)|$ is the number of particles that are from the same cluster as i and $A(i)$ are all the particles in the event except I itself. z_i, z_p and z_a are the output embedding vectors of the corresponding particles and τ is a changeable hyperparameter.

From the resulting embedding vectors, the clustering algorithm AgglomerativeClustering was used to force a classification of the particles into two categories. Explaining this algorithm would go beyond the scope of this thesis, but here is its documentation [25].

Using this contrastive loss function, the network achieved an output accuracy of 71.4%, with the best assignment accuracy employed. Compared to the node classification model this is a significantly lower score, which is why the idea was ultimately dropped.

Chapter 5

Clustering particles using transformers on MC truth data

One of the advantages of a GNN is that the nodes that are connected via edges can be pre-determined, so not every node has to be connected to every other node. However after the network was optimized several times by changing hyperparameters and message-passing algorithms, the best working architecture seemed to be one where almost all nodes in an event are connected. Furthermore, the best message-passing algorithm seemed to be an attention-based one. This network structure strongly resembles that of a transformer model. As the transformer models were optimized and greatly improved in the past years, employing a cutting-edge transformer model for this classification task made sense.

The form of the input data used for the transformer model was for the most part the same as the one used for the GNN. The only major difference is that instead of only calculating the invariant mass as an edge feature, the logarithm of the Minkowski inner product for the particles was also calculated, resulting in a two-dimensional edge feature vector. The inclusion of this was mainly due to trial and error, as several different edge features were tested. The logarithm of the Minkowski inner product is calculated in the following way:

$$\log(v_i \cdot v_j) = \log(E_i E_j - (p_{x,i}p_{x,j} + p_{y,i}p_{y,j} + p_{z,i}p_{z,j})) \tag{5.1}$$

In this chapter, the transformer architecture used will be explained in more detail followed by the results achieved by the model and compared to the GNN results from the previous chapter.

5.1 Transformer Architecture

Although transformers are mainly used in natural language processing, there are some cases where they also seem to work great in particle physics. One such case is for the classification of jets at LHC by H. Qu et al. in 2024 [23]. Here a transformer model was used to classify jets into different categories of origins (e.g. from heavy particles like H,

W^+, Z^0) and tag them to signal. This network achieved great results and outperformed the current state-of-the-art jet tagging algorithm (ParticleNet [11]) at LHC. This transformer is called the Particle Transformer (Par-T). As Part-T's classification task is similar to the classification task used in this thesis and its architecture can be easily adjusted, it seemed like a good idea to implement Part-T.
.
Some adjustments to the transformer architecture needed to be made, so the modified Part-T model that was used for this thesis will be explained in more detail. The in-depth explanation of the original Part-T model can be found in [23].

Figure 5.1 shows a complete overview of the transformer model. Here is a more detailed explanation:

Each of the particle features is embedded into a 128-dimensional vector and the edge features into a 64-dimensional vector. The features are then run through 8 particle attention blocks (PAB) each time updating their embedding. The edge features are always added to each of the PABs without being updated. There is no positional encoding required, as the position of particles in the input list does not matter.

Inside the PAB, the particle embeddings are run through a particle multi-head attention block (P-MHA) with 8 heads. This works very similar to how it was explained in chapter 3.3. The main difference is that the edge embeddings are added to the $QK^T/\sqrt{d_k}$ part in equation 3.7. After the P-MHA block several linear layers, LayerNorm (LN) layers, skip connections (denoted as +), and a GELU non-linearity[12], which is very similar to a Relu activation function, are added.

After the final PAB, the particle embeddings get sized down to a single score using a trainable weight matrix, resulting in the same style of output as for the GNNs.

One important difference between the GNN and transformer architecture is the input sizing. With the type of implementation that was used for transformers, the dimension of each input matrix has to be the same, while this could vary for GNNs. As in particle physics, the amount of particles per event varies from event to event, this can cause problem. To address this, the input matrix for each event in a batch is revised to match the size of the event with the most particles per batch. Events with fewer particles are padded with zeros to reach the size, and a masking technique is applied to ensure the padded values do not affect the model's computations.

Another difference in training the transformer compared to the GNN was that 17.64M events (98 files) instead of only 6.48 M events (38 files) were used. This was caused due to some overtraining in the initial trials of the transformer model. Also, the batch size was increased from 256 to 512 events per batch. This was done for better computation.

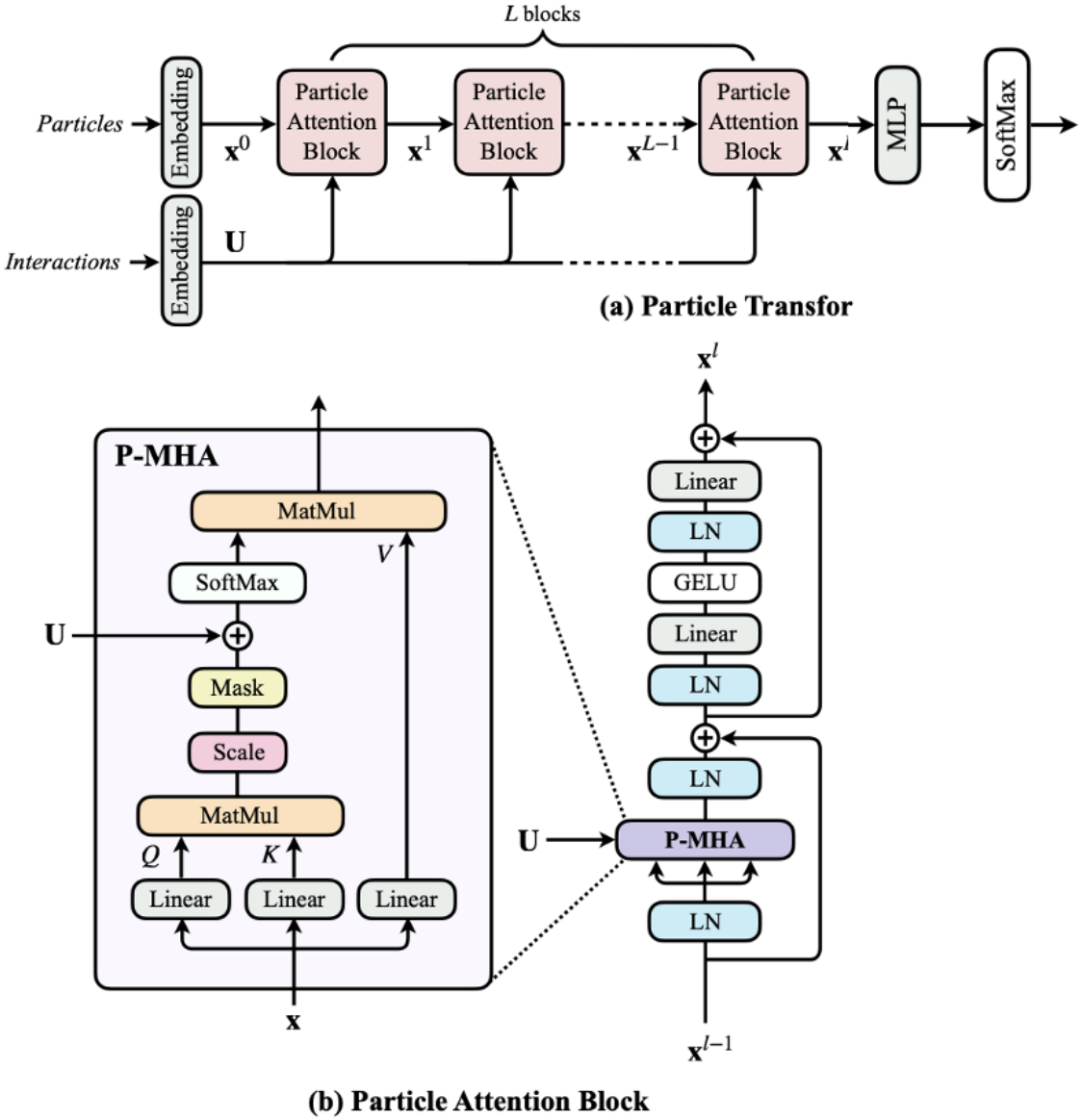

Figure 5.1: Schematic overview of the transformer model [23]

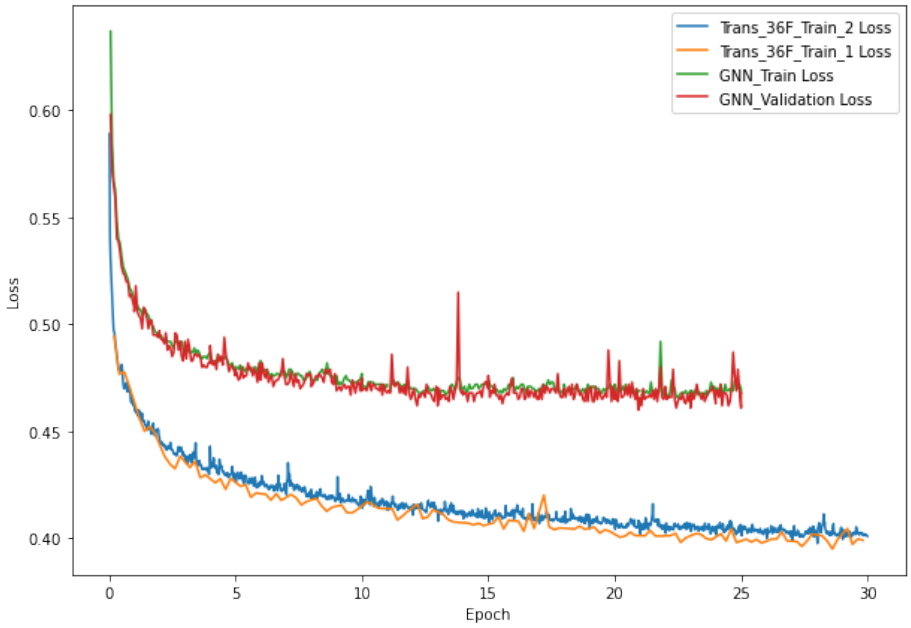

Figure 5.2: Training of GNN and Transformer - Average loss reducing after training over epochs

5.2 Results of the Transformer

Figures 5.2, 5.3, and 5.4 show the training results of the transformer model. As we can see, the transformer outperforms the GNN in every metric.

Remember that these accuracies were calculated using best assignment, where the accuracies were calculated for the original training label and for when these labels are flipped. Then the best accuracy is taken for each event.

The highest particle accuracy measured for one file was 79.9% and the highest perfect event accuracy for one file was 21.5%. To compare these results with FEI, the network has to be trained on reconstruction-level data, which will be discussed in more detail in the next chapter.

Figure 5.5 shows the calculated invariant mass for all particles in one class. For a perfect classification, the assumption would be that all the particles of one predicted class would create an invariant mass of a B meson at $5.28 GeV/c^2$. Due to the missing neutrino energy, this is not the case. This is displayed as the blue histogram in Figure 5.5. This shows the invariant mass of all the MC truth labeled classes. The red histogram however shows the invariant mass for all clusters predicted by the transformer model.

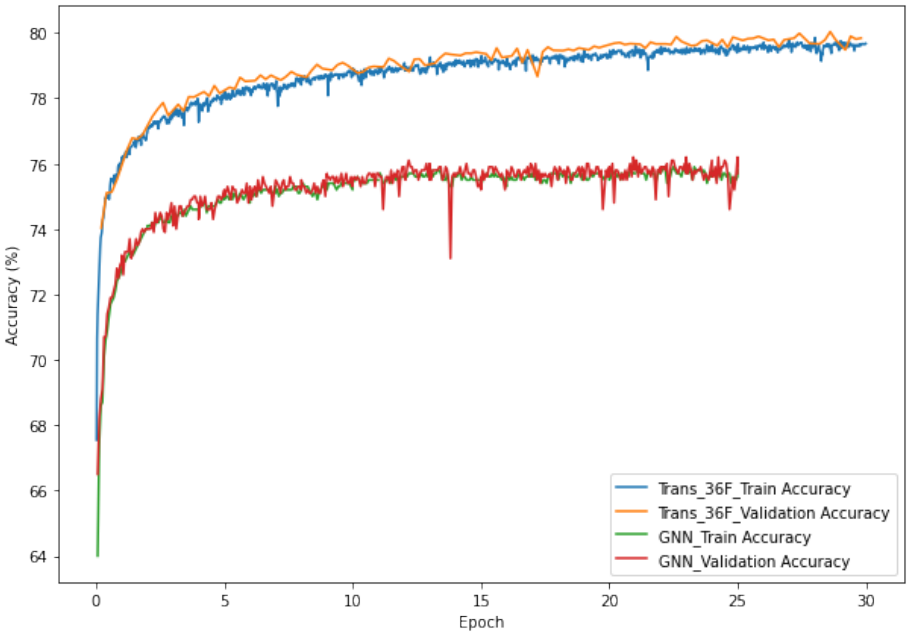

Figure 5.3: Training of GNN and Transformer - Average accuracy increasing after training over epochs

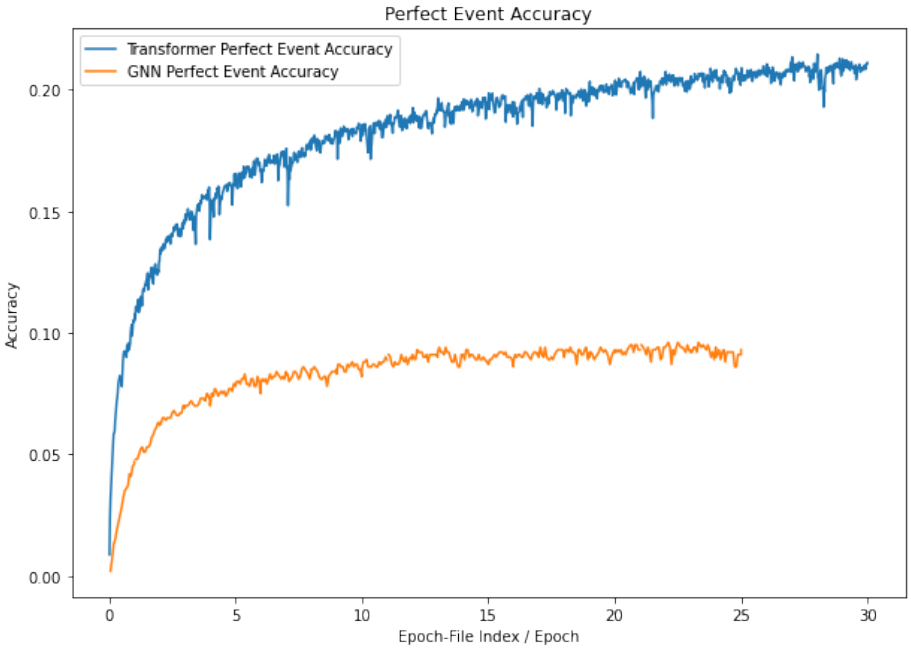

Figure 5.4: Training of GNN and Transformer - Average perfect event accuracy increasing after training over epochs

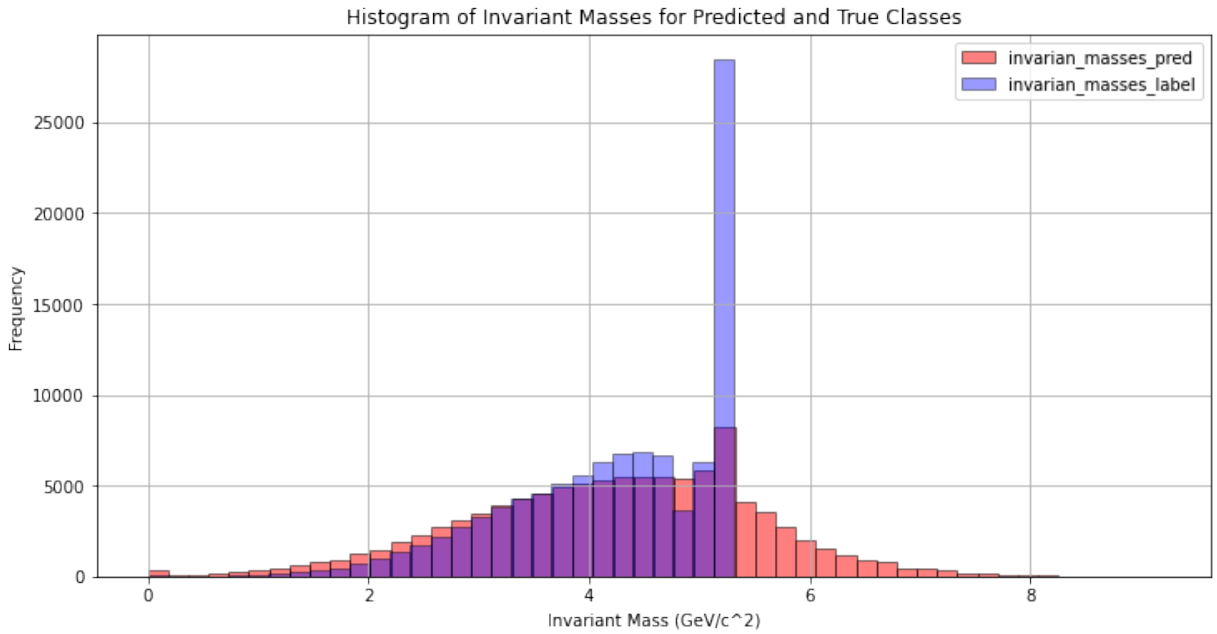

Figure 5.5: Invariant mass for predicted and labeled data for one file (180K events)

Chapter 6

Clustering particles using transformers on reconstruction level data and the comparison to FEI

To use this thesis' classification network on real data and compare it to the FEI benchmarks, testing the model on reconstruction-level data needs to be done. As explained in Chapter 2.3, reconstruction-level data is derived from simulated events that pass through a detector simulation. A detector simulation is performed by simulating all the various Belle II detectors and providing an output to each event that would be similar to the data received from real collisions.

Chapter 6.1 will explain how the various features of each of the particles are determined and the common problems that arise when reconstructing events from the data of a detector simulation. Then chapter 6.2 explains why not all the input particle types that were used in MC truth training can be applied to reconstruction-level data. The benchmarks of FEI will be covered in chapter 6.3, and how they can be compared. Finally, the results of the transformer for reconstruction level data will be shown and a comparison to FEI will be made in chapter 6.4.

6.1 Belle II reconstruction level data

6.1.1 Issues with detector level data

When reconstructing events from a detector simulation, all the various sub-detectors are simulated and with it all their statistical fluctuations and errors. This has the advantage that it simulates how the detectors would handle an event for real particle data in the detector, but with the added benefit that we still have the option to check on the actual particle features using the integrated MC Truth features. The main issues that arise when

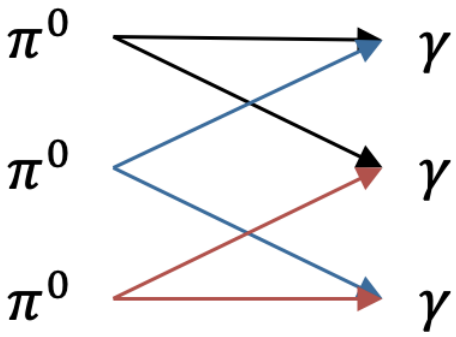

Figure 6.1: Basf2 trying to reconstruct π^0 from two gammas ($\pi^0 \to \gamma\gamma$), but faces double counting as "real" reconstruction cannot be resolved. (FSPs on the right and reconstructed pions on the left)

using the data from a detector simulation are listed below:

- **Background particles:** One of the main issues that arise when taking real data or using a virtual detector is that background FSPs are added to the event. These background particles come from several sources, but are not a product of the primary collision.
- **Missing particles:** When reconstructing the FSPs, the software tries to reduce the noise created by background particles by applying selection criteria on some features of the detected particles. Sometimes this leads to signal particles (from primary collision) being removed from the event.
- **Miss-identification:** As these detectors are not perfect in their resolution and have some statistical fluctuations, the identification of each FSP type is not always correct. Especially if two different particles cause similar responses in the detectors, the classification of these particles might be difficult. For the same reason, the particle features, like momentum and energy are not perfectly accurate.
- **Issues when reconstructing intermediary particles:** As the resolution and identity of each of the FSPs can not be perfectly determined, the reconstruction of intermediary particles from these FSPs often causes issues. As the input data for the model does not exclusively use FSPs, and these intermediary particles must first be reconstructed, this can result in issues. To reconstruct intermediary particles, all possible combinations that are allowed in the SM are considered for reconstruction, and specific selection criteria are applied to all potential reconstruction candidates. These selection criteria try to determine which reconstructions seem improbable and

Particle Type	Selection Criteria
Charged particles (π^+, e^-, μ^-, p^+, K^+)	$17° < \theta < 150°$ nCDChits > 20
γ	$E > 0.0225$ GeV for cluster in forward direction $E > 0.020$ GeV for cluster in barrel and backward direction

Table 6.1: Selection criteria for reconstruction of FSPs

then remove them. For example, if the two FSPs invariant mass is far away from the resonance energy of their intermediary particles, this reconstruction would be removed.

- **Double counting:** As the reconstruction of intermediary particles is not perfect, and as all possible reconstructions are considered, sometimes multiple intermediary particle candidates are constructed from one FSP. For example, if there are three photons, and we try to reconstruct a π^0 from two of them, the software considers all three combinations of reconstructing the pion and creates all three pion candidates. Then these candidates go through selection criteria, that filter out the least likely candidates. However, if the selection criteria are weak, or the intermediary particle is difficult to reconstruct, multiple candidates pass the criteria and result in double counting. This example is displayed in Figure 6.1.

6.1.2 Reconstructing FSPs

Since the input data for the network includes not only FSPs but also intermediary particles like π^0 and Λ^0, which typically need to be reconstructed, it became necessary to address the reconstruction of these particles. During the reconstruction process, the π^0 was particularly challenging to reconstruct accurately due to issues such as double counting. To resolve this, the γ particle was used directly instead of the π^0, thereby eliminating the errors associated with its reconstruction. The Λ^0 particles were also removed from the possible input particles, as they are rare in general and should not affect the model significantly

Therefore the resulting input particles for the model are π^+, e^-, μ^-, γ, p^+, and K^+. After including photons as possible input particle types, the average event size increases from ~ 16 to ~ 35 particles per event, making it more difficult to achieve a high perfect event score. Charged particles are derived from reconstructed tracks, whilst photons from the ECL clusters.

To find the particle identity of each of the charged input particles, a PID likelihood is calculated for each particle. This involves applying a mass hypothesis and determining the likelihood of a given particle having a specific PID, using data from the detectors. The function used for this process, stdCharged.stdMostLikely, is part of the Basf2 software framework [5].

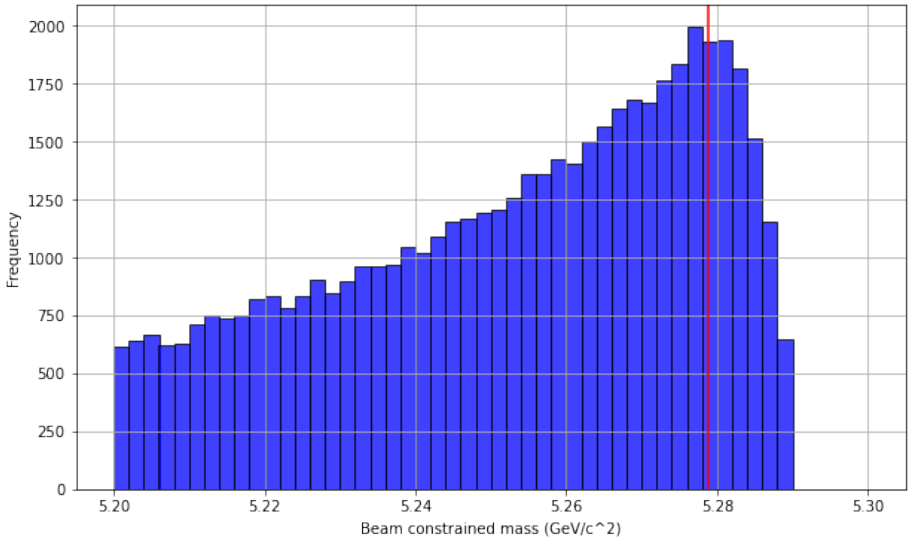

Figure 6.2: Beam constrained the mass distribution of candidates reconstructed from FSPs with the same true B-label without any further selection. The red line indicates the B-meson mass at 5.279 GeV

To then ensure a good reconstruction, some selection criteria are imposed on the particle candidates. These are given in Table 6.1. For the photon, these selection criteria cut out 40% of all photons detected.

The particle features differ slightly from those used in the MC truth section of the thesis. Since the true particle IDs via MC truth are no longer accessible, it is necessary to provide a probability score for each particle being a particular type. Even though each particle is reconstructed as a specific type, it is beneficial to include a probability score indicating the likelihood of the particle being classified into that type. This is accomplished using the PID score, which is calculated in the following way:

$$\mathrm{PID}_x = \frac{\mathcal{L}_x}{\mathcal{L}_\mu + \mathcal{L}_e + \mathcal{L}_\pi + \mathcal{L}_K + \mathcal{L}_p + \mathcal{L}_d} \qquad x \in \{\mu, e, \pi, K, p\} \tag{6.1}$$

where the $\mathcal{L}_x$ is the likelihood of the particle being of type x.

The following input particles are used: E_{CMS}, $p_{x,CMS}$, $p_{y,CMS}$, $p_{z,CMS}$, charge, PID_μ, PID_e, PID_π, PID_K, PID_p. Since kinematic features are not determined using MC truth, they are limited by the resolution of the various detectors. The edge features remain the same as in Chapter 5. Training the network on this data requires the use of MC truth to give each of the particles a classification label. This label however is not included in the input of the

network and is exclusively used to determine the loss score to train the network.

To evaluate how well the events have been reconstructed, the beam-constrained mass (M_{bc}) is calculated and plotted in Figure 6.2 for particle candidates that share the same label within each event. For each event, candidates labeled as originating from the same B meson are grouped, and their combined momenta are used to compute the beam-constrained mass.

The beam-constrained mass is given by:

$$M_{\text{bc}} = \sqrt{E_{\text{beam}}^2 - \vec{p}_B^2} \tag{6.2}$$

Where E_{beam} is half the energy of the beam in the center-of-mass frame (5.29 GeV), and $\vec{p}_B$ is the total momentum of the candidate particles assigned to a B meson in the CMS frame. By plotting the beam-constrained mass for these candidates, one can assess the quality of the reconstruction process. A successful reconstruction will result in a peak of the M_{bc} distribution at the known B meson mass, which is approximately 5.28 GeV.

This method serves as an indicator that the reconstruction has been correctly performed. If the M_{bc} distribution peaks at the expected B meson mass, it suggests that the candidates have been accurately reconstructed, confirming the reliability of the analysis. This approach is widely used in the Belle II experiment to validate the reconstruction process and ensure the accuracy of particle identification.

6.2 Transformer with reconstruction level data

6.2.1 Changes to the model due to new data type

As mentioned in the previous chapter, the introduction of the new data type presents additional challenges, requiring modifications to the network. The most significant change is the introduction of background particles, which requires incorporating a new label into the analysis. Previously, each particle was assigned one of two labels, representing its origin from one of the two B mesons. Now, a third label for particles that do not originate from any of the two B mesons and are thus classified as background is introduced.

To accommodate this new label, the loss function defined in equation 4.4 must be expanded to consider all possible particle labels. The updated loss function, still based on cross-entropy, now includes three possible classes (labels 0, 1, and 2):

$$L(x, y) = -\sum_{i=1}^{N_p} \sum_{j=0}^{2} y_{i,j} \log(x_{i,j}) \tag{6.3}$$

In this expanded loss function:

- N_p is the number of particles in a batch.
- $x_{i,j}$ is the predicted probability that particle i belongs to class j (where $j = 0, 1, 2$).
- $y_{i,j}$ is the true label with a value of 1 if particle i belongs to class j and 0 otherwise.

This formulation ensures that the network can differentiate between particles originating from either B meson or those classified as background. The network is trained to minimize this expanded loss function, which considers all three classes simultaneously. As before, backpropagation is used to update the weight matrix elements, as explained in Chapter 3.

The best assignment loss operates similarly to the previous method (equation 4.6), where the loss function is calculated both normally and with the 0 and 1 labels swapped. Since the background label is distinct and unambiguous compared to the other labels, it does not need to be swapped.

6.2.2 Transformer results for reconstruction level data

A total of 16.9 million reconstruction-level data events (spread across 94 files) were used to train the model over 20 epochs. These events exclusively consisted of $\Upsilon(4S) \to B^+B^-$ decays from the MC15 dataset which were briefly explained in chapter 4.1.

Figures 6.3 and 6.4 show the evolution of the accuracy during the training process on reconstruction-level data. With the inclusion of background particles, it is more appropriate to shift from the "perfect event" metric to a "perfect B meson accuracy" metric. This new metric evaluates how many events have at least one perfectly reconstructed B meson, which is the critical requirement when tagging a B meson. For MC truth-level data, this metric would be equivalent to perfect event accuracy.

The peak particle accuracy for the classification task reached 75.5%, while the highest perfect B meson accuracy measured 0.67%. As anticipated, the performance metrics for this network were lower compared to those achieved using MC truth particles. The perfect B meson accuracy, in particular, saw a significant decrease, largely due to the increased event size caused by the inclusion of photons.

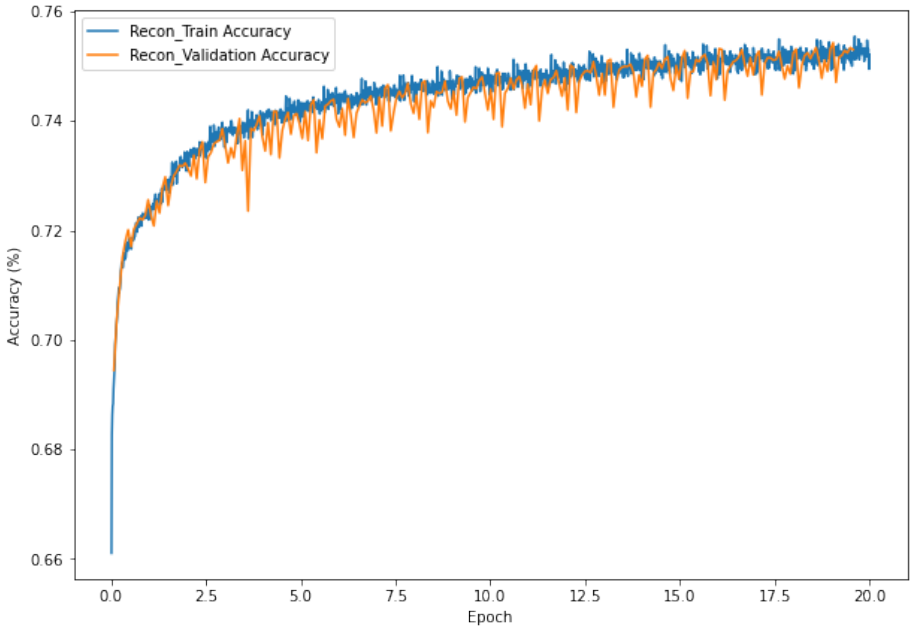

Figure 6.3: Training of transformer model on reconstruction level data - Average particle classification accuracy

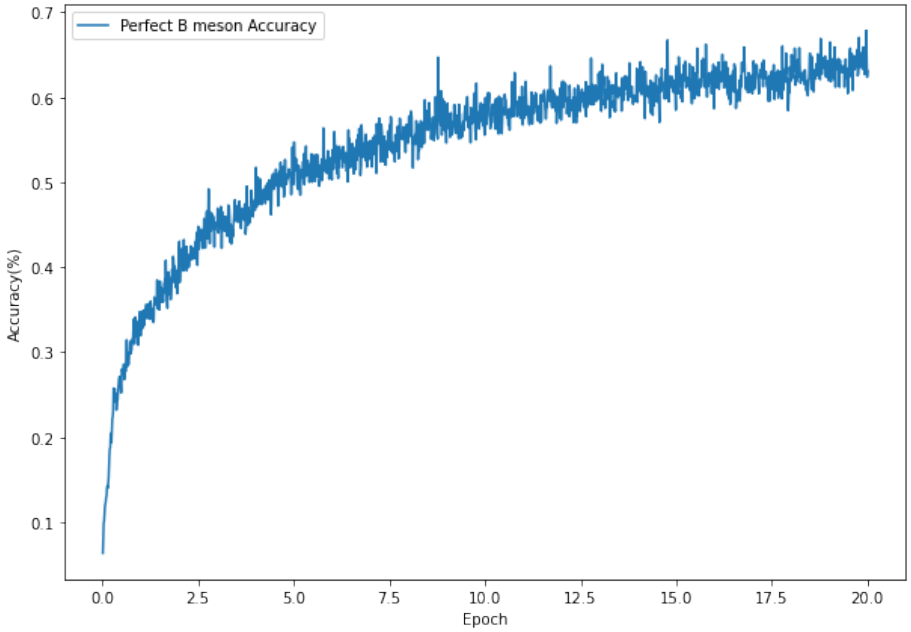

Figure 6.4: Training of transformer model on reconstruction level data - Average accuracy of reconstructing a complete B meson

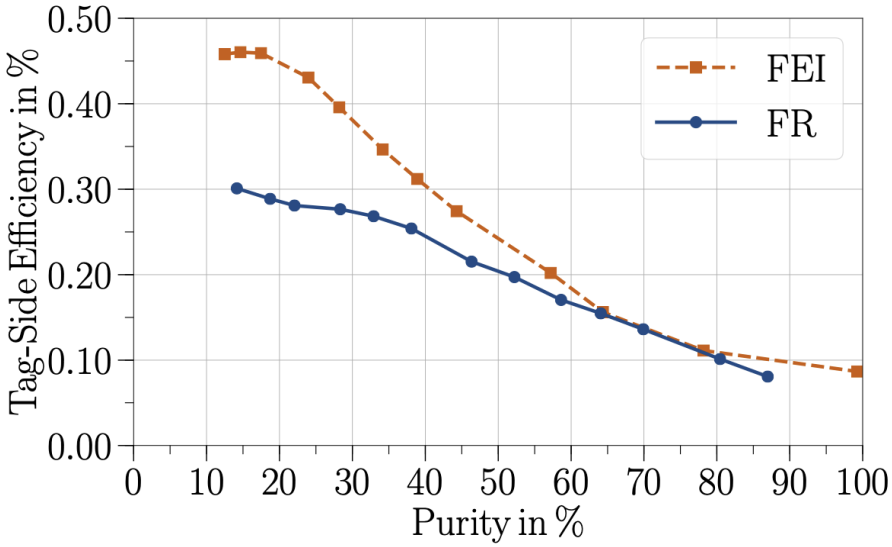

Figure 6.5: Purity and efficiency of FEI and the FR algorithm (predecessor of FEI) when altering selection criteria [15]

6.3 Comparison to FEI

6.3.1 FEI Benchmarks

To compare the results of the Transformer network with FEI, the FEI algorithm has to be discussed in more detail, by showing benchmarks of how it performs and which metrics are used.

Even though FEI was covered in chapter 2.5, here is a quick recap:

- FEI takes detector-level information and tries to reconstruct FSPs
- It then reconstructs decay trees from these FSPs, only considering an exclusive list of decay channels
- Depending on how stringent the criteria of FEI are, most events are discarded and only very easily identifiable tag-sides are fully reconstructed

The two main metrics that are used to calculate how well a tagging algorithm operates, are the tag-side efficiency and the tag-side purity [15]:

Tag-side efficiency - This measures the fraction of $\Upsilon(4S) \to B\bar{B}$ events that are still included after the FEI algorithm runs over all the data and also are correctly classified. For example, if the original data consists of 100 events and 10 events are selected by FEI,

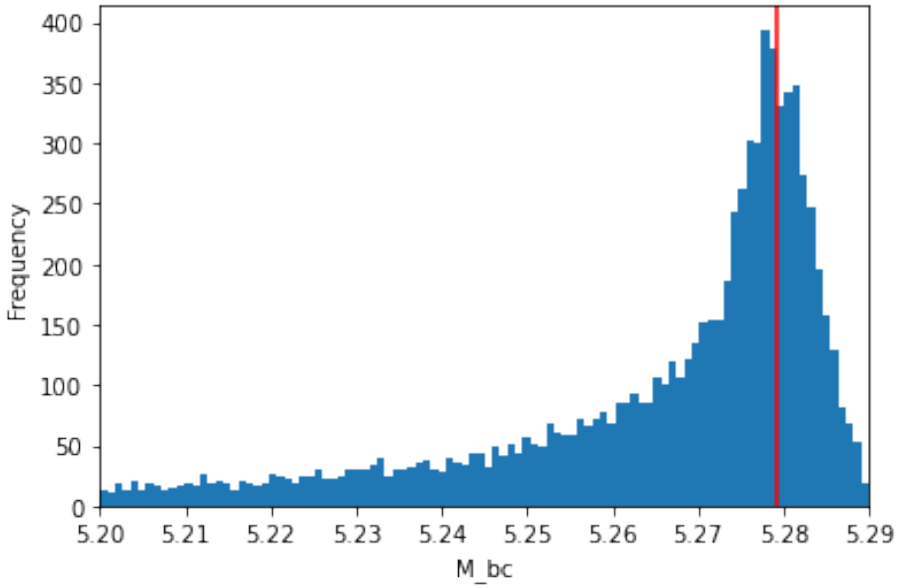

Figure 6.6: Beam-constrained mass of reconstructed FSPs for clusters of one B meson per event after ΔE cut is performed ($\Delta E \leq 0.1$GeV). The red line indicates the B-meson mass at 5.279 GeV.

where 5 of these have a correct tag-side reconstruction, then the overall efficiency is 5%.

Tag-side purity - This is a measure of quality for the reconstructed data by FEI. It measures the fraction of correctly reconstructed tag sides of the events that passed the FEI criteria. Using the same example as above, when having 100 input events and 10 of them are selected by FEI, where 5 have a correct tag, the purity is 50%.

Figure 6.5 displays how well FEI performs. With different selection criteria for FEI, we can see how the purity and efficiency change. FEI can impressively perform a tag-side purity close to 100% but only at an efficiency of 0.1%. When increasing the efficiency, so letting more data pass, the purity decreases dramatically.

6.3.2 Limitations of performance due to reconstruction inefficiencies

As FEI provides a full reconstruction of the tag side, as well as the clustering of the FSPs, relying on the correct reconstruction of the events is needed. The classification task will be done by the trained transformer network, but the reconstruction of the FSPs is done as explained in Chapter 6.1.2.

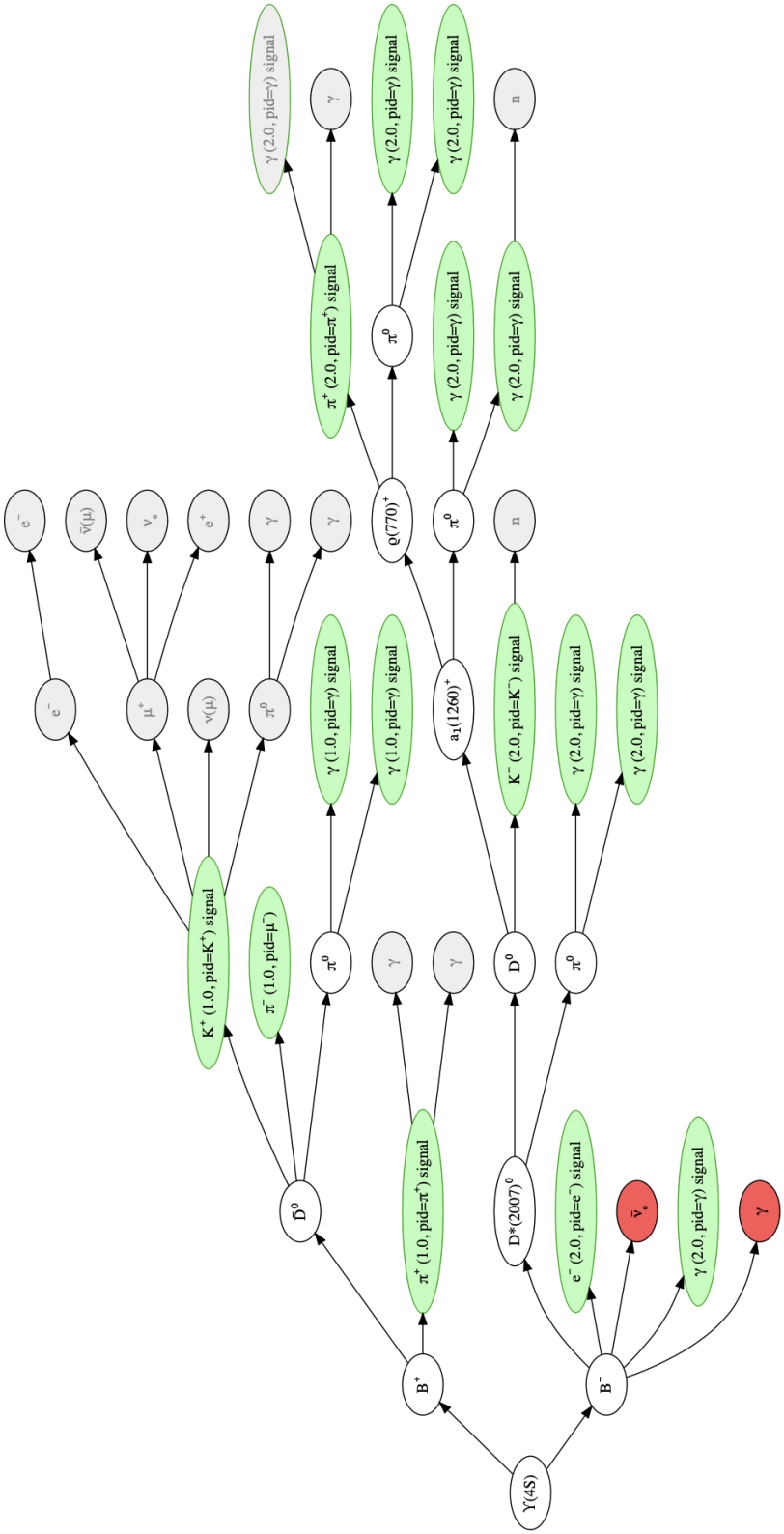

Figure 6.7: Schematic overview of correctly reconstructed B meson. A filled-in green circle indicates a correctly reconstructed primary particle, the red indicates particles that were not reconstructed and the green border around the grey circle indicates reconstructed secondary particles.

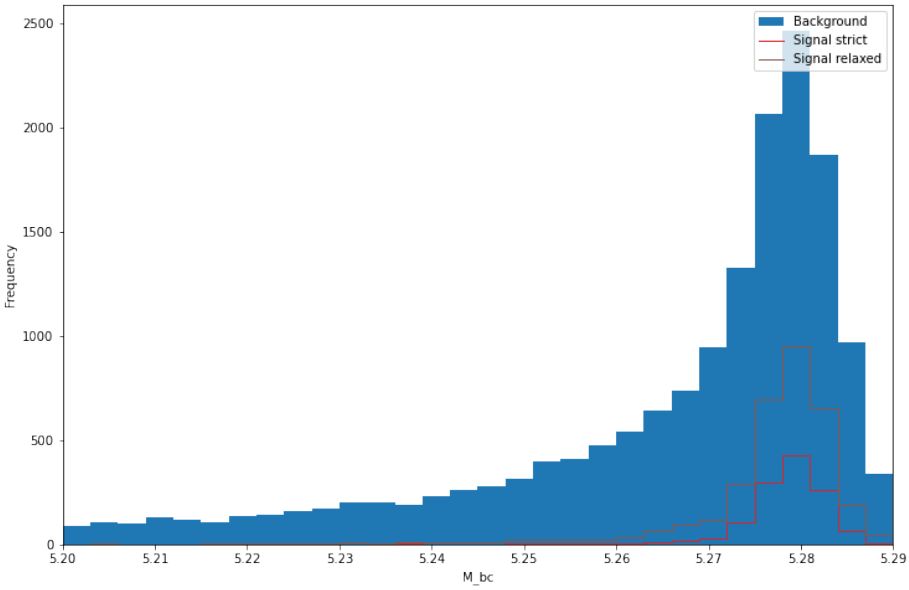

Figure 6.8: M_{bc} for all clusters, with relaxed and strict signal highlighted.

This means that the only events that can be considered to be tagged correctly by the network, are those where no particles (except neutrinos) are missing. Performing a selection on the data filters out the undesirable events (with missing signal particles).

There will be two global selections performed on the data. The first will be performed on the absolute difference between the beam energy and the total reconstructed energy of a cluster, which is called deltaE (ΔE) and is calculated in the following way:

$$\Delta E = |E_{cluster} - E_{beam}| \tag{6.4}$$

Here, $E_{cluster}$ represents the combined CMS energy of each particle cluster. A cut of $\Delta E \leq 0.1$ GeV is applied to each cluster. This means that if the event does not have at least one of the two clusters within these ΔE constraints, the event is excluded. As in Chapter 6.1.2, the beam-constrained mass of each cluster is calculated, clustering each of the input particles coming from the same B meson together. Figure 6.6 illustrates the beam-constrained mass after applying the cut, revealing a much sharper peak compared to Figure 6.2, where no such cut was applied.

The second selection criteria performed on the clustered candidates is to remove all clusters with $M_{bc} \leq 5.20$. As the range of Figure 6.8 only covers the M_{bc} of over 5.20, this is not visible in the Figure.

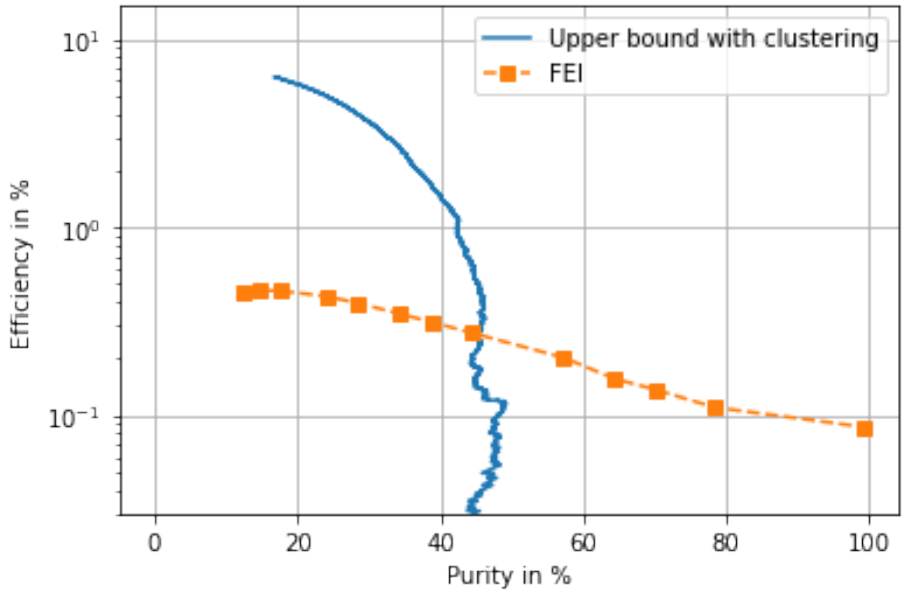

Figure 6.9: Efficiency versus purity of correctly reconstructed events by cutting on the beam constrained mass for relaxed signal. FEI results shown as comparison

Primary particles are the particles that are reconstructed as input for the model and are produced in the decay chain of the B meson. Secondary particles are those that appear from the interaction of the primary particles with the detector. Sometimes these secondary particles are detected and reconstructed. If both the primary particles and its secondary particles are clustered to the same B meson, this would be considered double counting. Therefore these secondary particles are also considered to be background and have to be predicted as such by the network.

As mentioned earlier, a successfully reconstructed event is when all primary particles, except for neutrinos, for one of the B mesons, are found by the reconstruction algorithm. These are referenced as relaxed signal events in later parts of the thesis. A strict signal event is when the criteria of the relaxed signal events are fulfilled and additionally, the reconstructed PID corresponds to the true particle type.

Figure 6.7 displays an example event where all the primary daughter particles for one B meson were found. This is an example of a relaxed signal event, as not all PIDs of the primary daughter particles for the B^+ were correctly identified. The other B meson cannot be classified as correctly reconstructed, as a primary photon is not reconstructed.

Figure 6.8 shows a plot of the beam-constrained mass for all event candidates after the global selection criteria are applied (blue). Of those candidate clusters, the relaxed and strict signals are also shown. Most of the correctly reconstructed clusters are centered

around the B mass peak. After the global selection criteria were applied to the data, the following table shows the efficiency and purity for both the relaxed and the strict signal.

Type	Efficiency (%)	Purity (%)
Relaxed	1.84	20.87
Strict	0.70	7.93

Table 6.2: Efficiency and Purity of the Relaxed and Strict signal with global selection criteria

Since the signal data is centered around the B meson mass, it is possible to construct an efficiency versus purity graph. This is done by selecting the clusters that are close to the B-meson mass ($M_{bc} \geq 5.27$ GeV) and progressively tuning the ΔE selection criteria. For each step, the corresponding purity and efficiency are calculated and compared to FEI. A plot for the relaxed signal is shown in Figure 6.9. This plot shows the upper bound of how well the network can perform when limited by the reconstruction efficiency of FSPs. Therefore the network can only reach a maximum purity of $\sim 45\%$ at low efficiency and a maximum efficiency of $\sim 5.5\%$ for low purity. This issue may be resolved by future studies on this topic.

6.3.3 Applying the clustering on relaxed signal events

Figure 6.10 shows the M_{bc} distribution of the predicted signal. The predicted signal refers to all the clusters, where the network perfectly classified the FSPs to the correct B-meson after applying the global selection criteria. The M_{bc} of almost all the clusters is centered around the B-meson mass. The orange outline shows how much of this predicted signal, is also classified as a relaxed signal, so was perfectly reconstructed within the relaxed criteria. A perfectly classified B-meson cluster is when each FSP network output (rounded to the most probable label) matches the training label for the FSPs of one of the B-mesons.

The purity and efficiency of the network output considering the global selection criteria for relaxed and predicted data are 0.07% for the efficiency and 0.08% for the purity. Although this is very low, it was expected after seeing the very low efficiency and purity of the base reconstruction. When only considering events with a successful reconstruction, 3.8% of them were classified correctly by the network.

It might also be interesting to mention that through these selection criteria, most of the semileptonic decay channels where removed from the data. Figure 6.11 shows the M_{bc} distribution of the relaxed signal split up into hadronic and semileptonic decays. The semileptonic clusters do not peak around the B mass as their neutrinos cant be detected resulting in missing energy in the cluster.

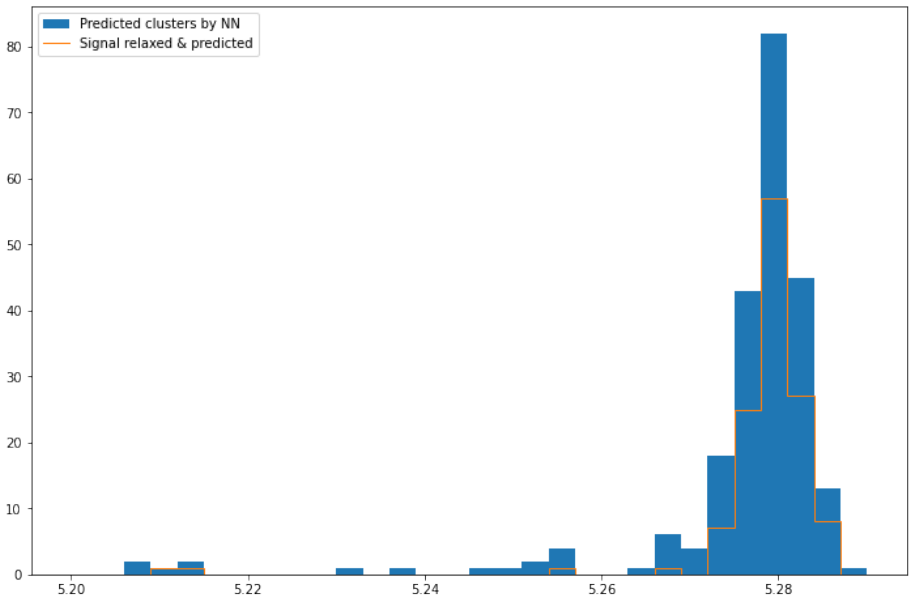

Figure 6.10: M_{bc} distribution of events with perfect B-meson prediction and candidates with perfect prediction and relaxed signal

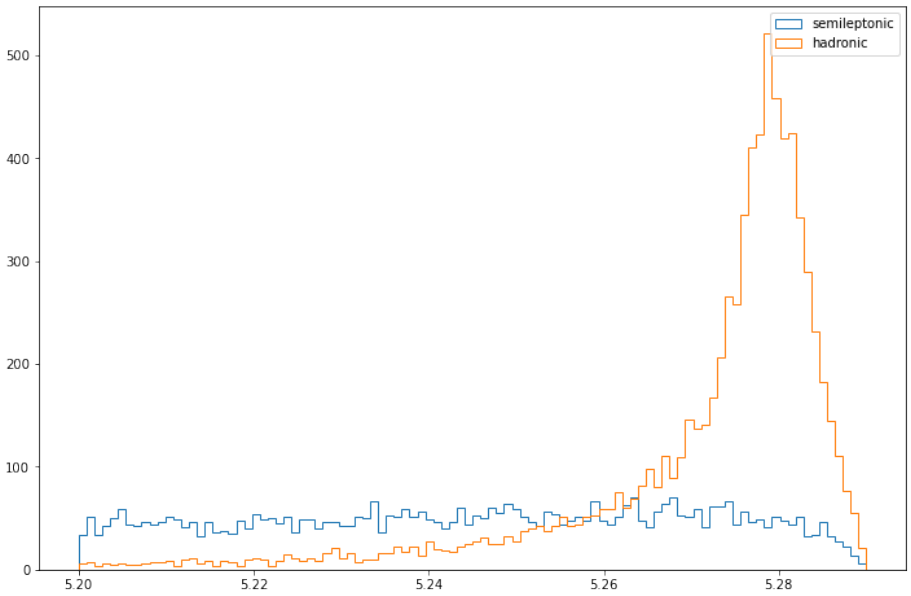

Figure 6.11: M_{bc} distribution of relaxed signal slipt up in hadronic and semileptonic decays

Chapter 7

Conclusion and outlook

At Belle II, B-tagging plays a crucial role, as information from the tagged B meson helps constrain rare signal decays involving neutrinos. The current B-tagging algorithm, FEI, employs a step-by-step process using boosted decision trees to reconstruct the tag sides, but it suffers from low efficiency. This thesis aimed to explore an alternative approach to tagging using modern deep-learning techniques. Instead of reconstructing entire events based on detector data, the proposed method focuses on conventionally reconstructing the final state particles (FSPs) of an event and then clustering these particles into two groups, corresponding to the two B mesons.

As a proof of concept, initially MC truth data was used, were the final state particles are already perfectly reconstructed, to train the network. The first approach involved using a Graph Neural Network. Various architectures, message-passing algorithms, and input features were tested to optimize the model. The best-performing GNN achieved a perfect event accuracy of 9.8%. Noticing similarities between the modified GNN and a Transformer model, the transformer architecture was explored. After further tuning the model's hyperparameters, a perfect event accuracy of 21.5% was achieved, which is very high for events without any selection criteria.

This proof of concept provided sufficient justification to move forward with reconstruction-level data. Without access to MC truth data, the first step was to reconstruct all FSPs. However, not all particles are easily identifiable by the detector or its software, and background particles often mix with the signal. A particular challenge was the reconstruction of the π^0, which led us to use the γ instead. This significantly increased the number of input particles per event for our network, rising from approximately 16 to 35, making it harder to achieve perfectly classified events. This led to the perfect B-meson accuracy to drop to 0.67%. Additionally, the purity of reconstructed events was constrained by the limitations of the detector and its software, which made it challenging to outperform FEI, even with near-perfect classification. The only potential area where the network could improve compared to FEI was in the low-purity tagging region.

Due to the low reconstruction efficiency and purity, the overall performance of the algorithm compared to FEI was poor, with an efficiency and purity of 0.07% and 0.08%, respectively. The main area for improvement is the reconstruction of the FSPs. Below are some ideas that could be explored in future work to address this:

- Develop a method for using π^0s instead of photons as input particles. In this thesis, double-counting issues led to the use of photons. However, with stricter selection criteria and a method to limit the number of π^0s in an event, it may be possible to reduce the particle count per event. This could make it easier for the network to perfectly classify all FSPs for a B-meson.
- In this thesis, the selection criteria were limited to ΔE and M_{bc} of an event. These cuts restricted the network's purity to a maximum of 45%, whereas FEI can achieve purities close to 100%. Future work could optimize these cuts or incorporate additional parameters into the selection criteria to enhance the quality of the reconstruction-level data. Another potential approach would be to use the probability score of correct FSP reconstruction to exclude events where perfect classification of all FSPs seems highly unlikely.

The performance of the network on MC Truth level data shows the potential of using deep learning techniques for B-tagging. Many issues still need to be addressed before becoming a viable replacement for FEI especially when it comes to the reconstruction of the FSPs.

All the code used in this thesis can be found at the following GitLab repository:
https://gitlab.desy.de/niklas.heckmann/master_thesis_niklas.

Bibliography

[1] A. Abdesselam et al. Search for dark matter at belle ii in events with a single visible photon and missing energy. *arXiv preprint arXiv:2003.12466*, 2020. Accessed: 2024-08-19.

[2] CERN. The higgs boson. `https://home.cern/science/physics/higgs-boson`, 2024. Accessed: 2024-08-19.

[3] Belle II Collaboration. Analysis fundamentals. `https://software.belle2.org/development/sphinx/online_book/fundamentals/05-analysis.html`, 2024. Accessed: 2024-08-19.

[4] Belle II Collaboration. Reconstruction fundamentals. `https://software.belle2.org/development/sphinx/online_book/fundamentals/04-reconstruction.html`, 2024. Accessed: 2024-08-19.

[5] Belle II Collaboration. stdcharged.stdmostlikely. `https://software.belle2.org/development/sphinx/analysis/doc/StandardParticles.html#module-stdCharged`, 2024. Accessed: 2024-08-19.

[6] The Belle II Collaboration. *The Physics of the Belle II Experiment*, chapter 3. Belle II Collaboration, 2018. Accessed: 2024-08-19.

[7] The Belle II Collaboration. *The Physics of the Belle II Experiment*, chapter 4. Belle II Collaboration, 2018. Accessed: 2024-08-19.

[8] S. Cristina. The transformer model. `https://machinelearningmastery.com/the-transformer-model/`, 2023. Accessed: 2024-08-19.

[9] DGL. Egatconv documentation. `https://docs.dgl.ai/en/0.8.x/generated/dgl.nn.pytorch.conv.EGATConv.html`. Accessed: 2024-08-19.

[10] A. Ceccucci et al. The ckm quark-mixing matrix. *Particle Data Group*, page 083C01, 2022. Accessed: 2024-08-19.

[11] H. Qu et al. Jet tagging via particle clouds. *arXiv preprint arXiv:2202.03772*, 2020. Accessed: 2024-08-19.

[12] Hendrycks et al. Gaussian error linear units. https://paperswithcode.com/method/gelu. Accessed: 2024-08-19.

[13] P. Kingma et al. Overview of the ckm matrix. *arXiv preprint arXiv:1112.1984*, 2011. Accessed: 2024-08-19.

[14] R. Pascanu et al. On the difficulty of training recurrent neural networks. https://arxiv.org/pdf/1211.5063, 2024. Accessed: 2024-08-19.

[15] T. Keck et al. The full event interpretation. *arXiv preprint arXiv:1807.08680*, 2019. Accessed: 2024-08-19.

[16] Vaswani et al. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. Accessed: 2024-08-19.

[17] GeeksforGeeks. Artificial neural networks and its applications. https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/, 2024. Accessed: 2024-08-19.

[18] IBM. What are neural networks? https://www.ibm.com/topics/neural-networks, 2024. Accessed: 2024-08-19.

[19] P. Khosla. Supervised contrastive learning. https://arxiv.org/pdf/2004.11362, 2021. Accessed: 2024-08-19.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[21] ML Mastery. A gentle introduction to cross-entropy for machine learning. https://machinelearningmastery.com/cross-entropy-for-machine-learning/, 2020. Accessed: 2024-08-19.

[22] Margarete Mühlleitner. The standard model of particle physics. https://www.itp.kit.edu/~maggie/icise/standardmodel.pdf, 2020. Accessed: 2024-08-19.

[23] Huilin Qu, Yutaro Iiyama, Benno Krohn, Jean-Roch Vlimant, Maria Spiropulu, and Daniel Whiteson. Particle transformer for jet tagging. *arXiv preprint arXiv:2202.03772*, 2024. Accessed: 2024-08-19.

[24] Lea Reuter. Full event interpretation using graph neural networks. Master's thesis, Karlsruhe Institute of Technology (KIT), 2022. Accessed: 2024-08-19.

[25] Scikit-learn. Agglomerativeclustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html, 2024. Accessed: 2024-08-19.

[26] SLAC National Accelerator Laboratory. Accelerator science: Intensity frontier. https://www.slac.stanford.edu/econf/C1307292/docs/Intensity-2.pdf, 2013. Accessed: 2024-08-19.

[27] TensorFlow. Mnist dataset. https://www.tensorflow.org/datasets/catalog/mnist, 2024. Accessed: 2024-08-19.

[28] Arthur Thaller. grafei: Full event interpretation using graph neural networks at belle ii. https://indico.in2p3.fr/event/22938/contributions/93158/attachments/63073/86647/graFEI_16_03_at_v2.pdf, 2021. Presentation at IN2P3, Accessed: 2024-08-19.

[29] Mark Thomson. *Modern Particle Physics*. Cambridge University Press, Cambridge, UK, 2013. Chapter 4: Gauge Bosons and the Standard Model.

[30] Wikimedia Commons. Standard model of elementary particles. https://upload.wikimedia.org/wikipedia/commons/thumb/0/00/Standard_Model_of_Elementary_Particles.svg/1390px-Standard_Model_of_Elementary_Particles.svg.png, 2024. Accessed: 2024-08-19.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Dr. Nikolai Hartmann and Prof. Thomas Kuhr for their invaluable supervision and mentorship throughout my thesis. When I began this project, I was relatively new to practical machine learning tasks, but their guidance helped me learn and grow immensely in the first few months. I am especially thankful to Dr. Hartmann for providing fresh ideas and perspectives. I am also grateful to the entire Flavor Physics group at LMU for consistently offering new insights and ideas during our weekly meetings, particularly when I encountered challenges.

Erklärung/Declaration

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt zu haben.

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis.

München, 13.09.2024
Niklas Heckmann