Improved message passing for Graph Neural Network based Full Event Interpretation in proton-proton collisions

Master's Thesis in Physics

Azusa Uzuki

Supervised by
Prof. Dr. Nicola Serra
Dr. William Sutcliffe
Dr. Julián García Pardiñas
Dr. Jonas Eschle

November 18, 2024

Abstract

In high energy physics, graph neural networks (GNNs) are widely studied especially for reconstruction and identification since graphs are often preferred when representing particle collision data. Collision data at the LHCb is highly complex, which is further amplified by increases in the proton-proton collision multiplicity for the detector upgrades. This motivates continuous developments of algorithms that can perform tasks while effectively handling the complexity and use of GNNs seems a suitable approach.

Starting from an idea of the Deep-learning based Full Event Interpretation (DFEI) proposed by J. G. Pardiñas, *et al.*, this thesis explores advantages of a new GNN with weighted message passing to reconstruct the hierarchical decay chains of heavy hadrons from the reconstructed particles using dataset simulated for the LHCb Run 3 condition. With future improvements, this work could contribute to developments of an inclusive trigger system or offline physics analysis at the LHCb.

Contents

Acronyms

BCEL	binary cross entropy loss.
CEL	cross entropy loss.
DFEI	deep-learning based full event interpretation.
GNN	graph neural network.
LCA	lowest common ancestor.
LHC(b)	large hadron collider (beauty).
ML	machine learning.
MLP	multilayer perceptron.
NN	neural network.
NP	new physics.
SM	standard model.

1 Introduction

The theoretical description of matter and its interactions at the smallest scale is known as the standard model of particle physics (SM). Over the past sixty years, the model has successfully explained and predicted a number of observed phenomena and experimental results, however, considerable issues are still present. Most importantly, it is unable to explain cosmological and astronomical evidence for a dark matter and the imbalance of matter and anti-matter in the universe. In addition, it does not provide mechanism for neutrino masses and not include the force of gravity. This motivates the development of theories beyond the SM or new physics (NP), which is one of the most researched areas in particle physics today.

The LHCb at the Large Hadron Collider (LHC) at CERN in Geneva is designed to perform experiments of the SM with enhanced precision and hence search for hints of NP in the form of potential deviations of measurements from their SM predictions. It is specialised in studies of heavy-flavour hadron decays produced in proton-proton (*pp*) collisions at high energy and luminosity, which are currently 13.6 TeV and 9.8 fb^{-1}. The use of hadron collisions enables it to achieve a high centre of mass energy, however, such proton collisions have a large particle multiplicity of O(100) at the current luminosity condition. Furthermore, the energy and momenta of interacting partons within the protons are unknown limiting the reconstruction of particle properties from the energy and momentum conservation. These cause challenges of interpreting the collected data. Innovation of machine learning (ML) brings significant advantages to analyse such large and complex data at various steps in the experiment such as object reconstruction, event selection, and simulation. In general, ML provides a system that is automatically adjusted based on data or experience to solve given tasks without explicit instructions.[1] A number of ML algorithms, especially, neural networks (NNs) have been developed and used in the LHC and the emergence of deep learning has further boosted the ability.[2] A NN allows to derive high level features, which are then used for instance to identify particles, from the raw information of the detector readouts more effectively than traditional approaches that required collections of analysis specific for smaller features.[3]

The need of efficient algorithms is growing due to scheduled increase in luminosity at the LHCb in the near future. One prospective approach is a Graph NN (GNN) based decay reconstruction, which profits relational structure of final state particles measured in the detector. This thesis proposes a novel GNN architecture with improved message passing drawing upon a recent research the Deep-learning based Full Event Interpretation (DFEI) [4]. The model is trained and tested with datasets simulated under the Run 3 condition at the LHCb.

This thesis begins by providing the motivation in a broader picture in Section 2, and the basic idea of NNs and GNNs in Section 3. The following sections focus on contents more specific to this thesis. Section 4 and 5 explain the algorithm and dataset and Section 6 is dedicated for the results. Finally, Section 7 concludes this work with discussion of possible extension of this study for future research.

2 Overview of High Energy Physics

2.1 The Standard Model of Particle Physics

The SM provides theoretical descriptions of the fundamental particles in the universe and their interactions. The SM particles are grouped into half-integer spin particles called *fermions* and integer spin particles called *bosons* as shown in Figure 1. For every SM particle, there exist anti-particles, although some of anti-particles are their own particles. The three fundamental forces, strong, electromagnetism, and weak forces are described by quantum field theories and are mediated by corresponding bosons. The fermions make up matter and are further classified into *quarks*, which carry colour charges, and *leptons*, which do not carry colour charges. All fermions interact with the weak force, while only particles that carry colour/electric charges interact with the strong/electromagnetic forces. All the elements of the SM were confirmed by experiments, concluding with the discovery of the last remaining element, the Higgs boson, which was discovered at the LHC in 2012. Although the SM describes numerous experimental data with great precision, various unexplained observations and unanswered questions indicate its incompleteness and needs for a NP model. Some of the most notable phenomena that the SM does not address are astrophysical observations of dark matter, neutrino masses, the matter-antimatter asymmetry, and fermion generations (flavour problem).

In the SM, the asymmetry between matter and antimatter is partially explained by Charge-Parity (CP) violation in the Cabibbo–Kobayashi–Maskawa (CKM) matrix, however, with a much smaller rate than that required to explain the observed matter dominated universe. The CKM matrix enables one to calculate the likelihood of a quark transitioning to another quark and the complex matrix elements are responsible for CP violation in the quark sector. Therefore, measurements of flavour changing in hadron decays provide precise tests of the SM and probe potential NP that explains the matter-antimatter asymmetry. Experimentally, CP violation was first discovered in neutral kaon decays [5] and a large number of studies have been conducted in hadrons containing b or $\bar{b}$ quarks (collectively called beauty hadrons) such as the Belle collaboration [6] and the LHCb experiment [7].

The experimental tests of the SM relies on statistical analysis of data as the predictions are fundamentally given by the probabilities due to the quantum mechanical nature of the theory. Therefore, large samples are required in order to precisely quantify certain CP violating quark transitions. Additionally, the heavy SM particles such as ones with b quarks are unstable and decay shortly to lighter particles in the standard condition, setting out the need for particle colliders with high energies.

2.2 LHC and LHCb

The Large Hadron Collider (LHC) [9] is a powerful particle accelerator and collider at the CERN accelerator complex in Geneva, Switzerland. The machine was periodically upgraded since the first operation in 2008 to increase the statistics for investigating small effects due to NP. The collision rate is determined by an instantaneous luminosity, which is a measure of the particle flux with respect to the collider geometry. The higher luminosity, the greater number of heavy particles of interest can be produced in the collisions if a sufficiently high center of mass energy is provided.

The LHCb is one of the four large experiments at the LHC as well as ALICE, ATLAS, and CMS, being carried out by a large number of international collaborators. The LHCb is primarily focused on precision measurements of beauty and charm hadrons (b- and c- hadrons) produced in the pp collisions. Using collected data of those hadron decays, a broad range of

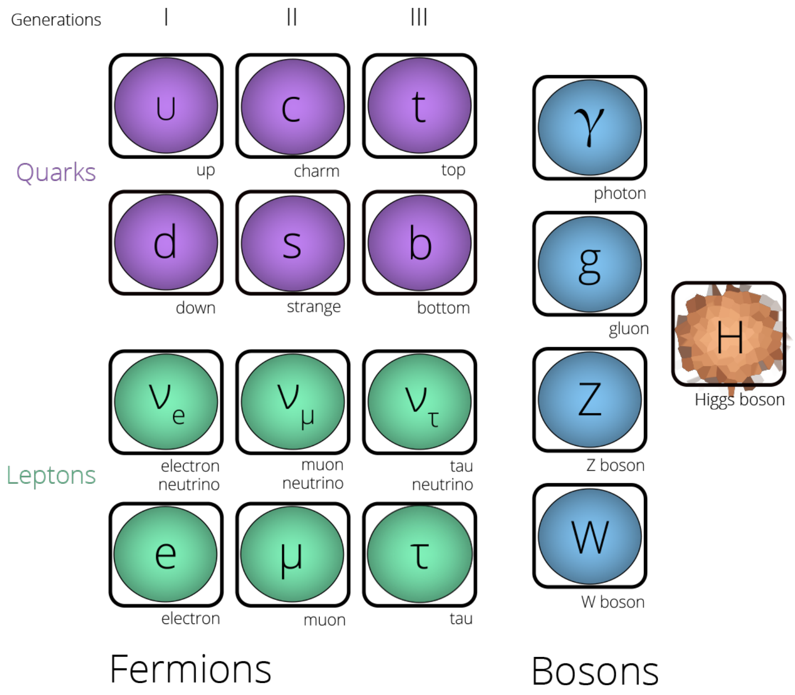


Figure 1: Table of the SM of particle physics [8].

physics programs have been carried out, including measurements of CP violation and search for NP. In particular, rare semileptonic decays such as $b \to sl^+l^-$ are studied extensively since new particles might allow couplings that are not possible in the SM and be observed as deviating rate or kinematics of the decays. [10]

During Run 1 and Run 2 operated between 2010 and 2018, the LHCb collected a total of $9fb^{-1}$ pp collision data at an instantaneous luminosity of $\mathcal{L} \sim 4 \times 10^{32} cm^{-2}s^{-1}$.[11] Recently, a major upgrade (Upgrade I) of the LHCb detector has been completed on both the hardware and software sides for effective data collection during Run 3 and 4. A further upgrade (Upgrade II) is planned for the High-Luminosity (HL) LHC. The instantaneous luminosity is raised to $2 \times 10^{33} cm^{-2}s^{-1}$ for Upgrade I and further tenfold increase is expected for Upgrade II.[12] The following description of the detector is applied to the Upgrade I condition, which is documented in ref. [11] in detail.

In high energy collisions, b-hadrons are known to be produced predominantly at small angles along the beam axis in forward and backward directions. For this reason, the detector was constructed with a forward angular coverage of approximately $10 < \theta < 300$ mrad opening from the particle interaction point. As indicated in Figure 2, a right-handed coordinate system is usually defined as the z axis is parallel to the beam and the x-y plane lays perpendicular to it with the y axis aligning vertically. b- and c- hadrons that are produced at the collision point, which is located on the leftmost of the figure, decay shortly into lighter and more stable particles. Those particles travel through the LHCb detector while interacting with its materials. Data collected at various points in the detector are combined and associated to the particles that are most likely to produce the readouts. This process is called reconstruction of particles.

The detector consists of several subsystems including tracking, particle identification, and trigger systems. The tracking system measures electric signals called hits created by charged particles passing through tracking stations and reconstructs their trajectories (tracks). Its main components are the Vertex Locator (VELO), the Upstream Tracker (UT), and the Scintilating Fibre tracker (SciFi). The VELO is located close to where *pp* collisions occur and measures the position of the *pp* interaction points (primary vertices) and displaced decay points (secondary vertices) of relatively long-lived hadrons. The UT measures the tracks of charged particles, which are combined with the VELO tracks to give a rough estimation of the momentum. It also makes important contributions to increase the processing speed and reduce the fake track rate.[13] The SciFi further provides measurements of the charged particle tracks and their momentum. Between the UT and the SciFi, a 4 Tm dipole magnet is installed to bend the charged particles so that their momentum can be measured. The particle types are identified by two ring imaging Cherenkov detectors (RICH1 and RICH2), electronic and hadronic calorimeters (ECAL and HCAL), and muon chambers (M2-5). The information collected by those systems is processed through the trigger system to reduce the size of data. It is fully software based event selection built with two successive triggers, high level trigger (HLT) 1 and HLT2.

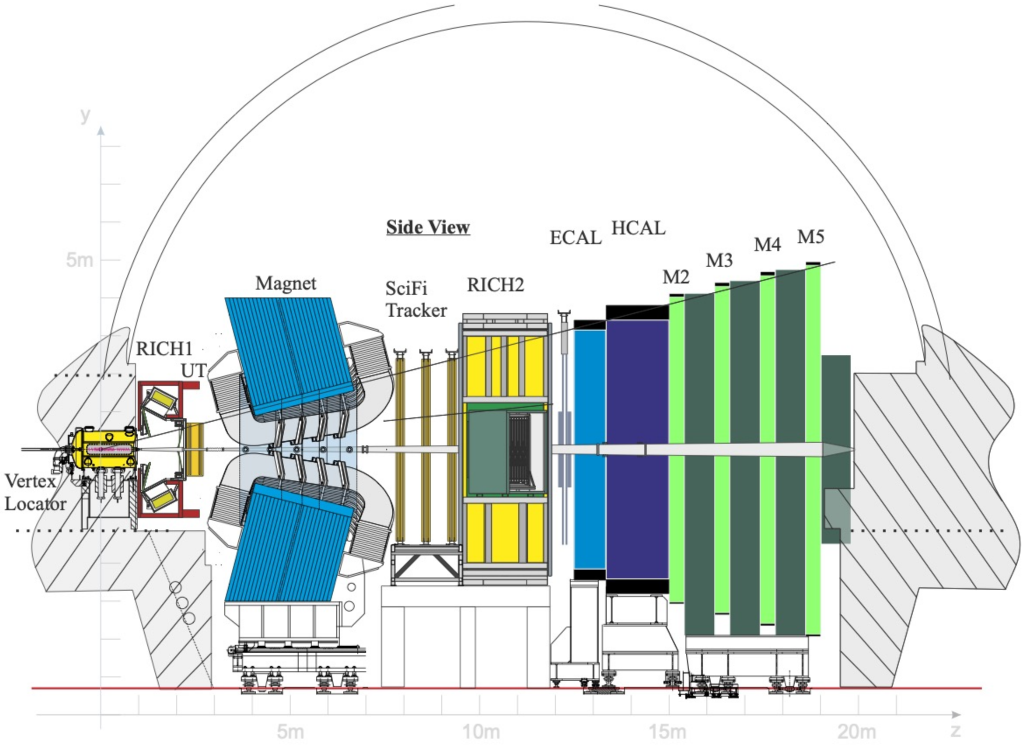


Figure 2: Schematic view of The LHCb detector after the Upgrade I, the individual components are explained in detail in ref. [11].

2.3 Trigger system at LHCb

The trigger system is required to reduce the data to be stored due to the limited hardware capacity, while keeping the events containing interesting signals for offline physics analysis. It has a significant role for the LHCb detector as the rejected events are permanently lost and cannot be further considered in the offline analysis.

The amount of data stored on a disk is proportional to the running time [s], the trigger output rate [kHz] and the average event size [kB]. In a condition where interesting events occur frequently, the trigger rate is necessarily high and the event size should be reduced to keep the data size manageable within available computational resources. This is further emphasised by the LHCb detector upgrades mentioned above. They led to an increase in the number of pp collisions per event from approximately 1 during Run 1 and 2 to ~ 5 during Run 3 and 4, and an expectation of ~ 50 after the Upgrade II. This results in datasets with greater multiplicities of tracks and b-hadrons, which further complicates reconstruction. In Run 3, effective data filtering is possible as a result of extended time frame available for processing events, which is realised with the fully software based trigger system. HLT1 decreases the event rate to a size small enough to be processed in HLT2, under a short time constraint using information of charged particle tracks. A buffer is installed between HLT1 and HLT2 and enables HLT2 to perform offline-quality reconstruction and selection algorithms. At this stage, the event size is also handled based on the signal candidates that executed the trigger. Specified subsets of the events as well as the trigger candidates can be saved using selective persistence, which is an essential part of the reduction.[14]

This event level reduction becomes more crucial in higher luminosity or pileup conditions in the HL-LHC era as almost all events will contain decays of interest, with each of the events being larger in size. Therefore, algorithms are required to decide which parts of the event are relevant to physics search rather than which events are relevant.[4]. This motivates the need for sophisticated ML approaches including the DFEI, which will be introduced in Section 4.1.

3 Deep learning

3.1 Neural Networks

A neural network (NN) is a class of non-linear learning methods and has been developed for various range of applications. Comprehensive introductions of NNs can be found elsewhere for example, ref. [15]. This section explains supervised learning with NNs focusing on an example of classification with a multilayer perceptron (MLP). Other widely used NNs are a convolutional neural network (CNN)[16], recurrent neural network (RNN)[17], and Transformers[18].

In a supervised classification task, a given dataset includes entities which are categorised into some classes and a NN model aims to reproduce the correct output classes from the input features by learning an appropriate function to estimate the probabilities for the various classes. The process of learning involves adjusting or *training* the model using a training dataset. The key metrics of the training procedure are a *loss* and its gradient, which are commonly computed through forwards and backwards propagation [19] methods. A loss measures the differences between the outputs (predictions) from the model and the correct labels (targets) and is often defined by a cross entropy loss (CEL) as described shortly. The objective of training is to minimise the loss by gradient descent, in which the learnable parameters are adjusted.

A MLP is one of heavily used NN architectures and consists of an input layer, one or more hidden layers, and an output layer. Each layer contains neurons storing values and feeds the following layer with corresponding non-linear functions. In the forward pass of a MLP, the values are processed in a single direction from the input layer to the output layer as depicted in Figure 3. A value y_{i+1} of a neuron in a hidden or output layer $(i+1)$ is determined by the neurons (1 to N) of the previous layer (i) with Equation 1 where w_{ij} and b are the learnable parameters called *weight* and *bias* respectively and f is an activation function responsible for the non-linearity. A sigmoid ($\sigma(x) = (1+e^{-x})^{-1}$) and a rectifier ($\text{ReLU}(x) = max(0, x)$) are popular choices for such function.

$$y_{i+1} = f(\sum_{j=1}^{N}(w_{ij} \cdot x_{ij}) + b) \tag{1}$$

Once all the layers are fed for entire batches of samples in a given dataset, the model performance is evaluated by the average loss over the samples. The back-propagation is used to determine the associated gradients of the loss with respect to the parameters operating from the output layer to the input layer. The parameters are decreased by the gradients multiplied by a small step called *learning rate* in order to reduce the loss. This training process is repeated until the desired loss is achieved or maximum number of training steps or iterations over all batches of data (*epochs*) is reached.

Usually, the activation function for the output layer is a softmax $\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ for multi-class and a sigmoid $\sigma(x)$ for binary classification to convert the outputs to numbers between 0 to 1. Those functions as well as aforementioned ReLU are differentiable so that the gradients can be calculated. After completing the training phase, the final prediction is the class with the highest probability, which can be computed by a non-differentiable argmax function.

The CEL can be used for multi-class classification and calculates the mean deviation between the model predictions and the targets by Equation 2. S and C are the number of samples and classes and $p_{i,c}$ is the predicted probability of a sample i belonging to a class c. $y_{i,c}$ is 1 if the target class is c for the sample i, and it is 0 otherwise, therefore, this equation implies that the loss is smallest when the prediction for the target is correct.

$$\text{CEL} = -\frac{1}{S}\sum_{i=1}^{S}[\sum_{c=1}^{C} y_{i,c} log(p_{i,c})] \tag{2}$$

For binary classification particularly, this turns to Equation 3, which is called binary cross entropy loss (BCEL), as the prediction can be defined by a single number p_i between 0 and 1, which is closer to 0 for the class 0 and closer to 1 for the other class.

$$\text{BCEL}(y_i, p_i) = -\frac{1}{S}\sum_{i=1}^{S}[y_i log(p_i) + (1 - y_i) log(1 - p_i)] \tag{3}$$

The CEL and BCEL can be adjusted for unbalanced datasets by scaling the log terms by weights of the classes, with a weight of a class A (w_A) calculated by

$$w_A = (N_{\text{samples}})/(N_{\text{classes}} \cdot N_{\text{samples of A}}) \tag{4}$$

where N_{samples}, N_{classes}, and $N_{\text{samples of A}}$ are the numbers of samples in total, of classes, and samples belonging to the class A in the dataset.

While the learnable parameters are modified during training, the learning rate, epochs, the number of hidden layers, neurons in each hidden layer and in the output layer define the model architecture prior to the training and are called hyperparameters. Hyperparameters that yield a minimum loss cannot be determined by the gradient method but need to be chosen by users from experience or optimization search methods. Finding a set of optimal hyperparameters is called hyperparameter optimization or tuning [20] and discussed in later sections.

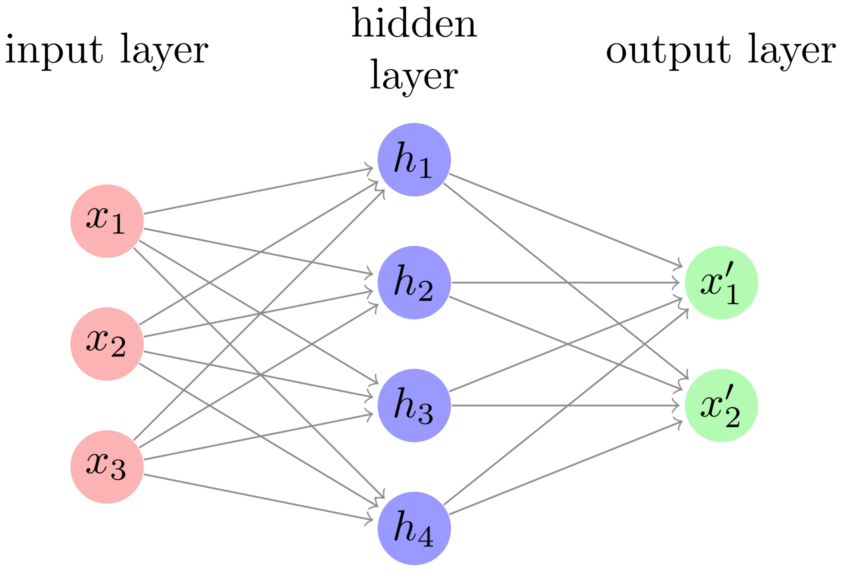


Figure 3: An architectures of a simple MLP.

3.2 Graph Neural Networks

In various domains, data naturally forms a complex relational structure, for example, social networks and connections in a brain. In high energy physics, particle collision data such as relations between tracker hits or reconstructed objects in none uniform detector geometries can be better represented by graphs than vector or grid structures.[21]

These types of data can be represented as a graph with a set of objects called nodes ($v \in V$), and a set of edges ($e \in E$), which refer the pairwise relations between nodes. Each graph, node and edge can be represented by a vector of features $\boldsymbol{u}$, v_i and e_k respectively. In a directed graph, each edge e_k connects a sender node v_{sk} to a receiver node v_{rk}, thus the edge contains the indices of v_{sk} and v_{rk} in addition to its other edge features e_k. Graphs are flexible in their size and nodes are primarily unordered or permutation invariant unlike other common data representation such as sequences and images. Many successful ML models including MLPs and CNNs expect a simple array-like data structure as the input, and therefore they process graphical data as flattened vectors and do not have a built in permutation invariance of the geometric representation.

A graph neural network (GNN) is a class of neural networks, which was proposed by ref. [22] to process graphical data effectively. The rich graph representations allow GNNs to perform classification tasks in the edge, node, or graph level. For example, node classification can be used for an analysis of sentiment in language with the nodes representing each word in a sentence.[23] Variations of GNNs have been developed such as the graph convolutional network (GCN) [24], graph attention network (GAT) [25] and more general message-passing network with permutation invariant aggregation functions.[26]

Ref. [27] proposed a GNN architecture called graph network (GN), which generalises and extends a range of networks and the core block, GN block, allows a graph-to-graph computation. It processes an input graph $G = (\boldsymbol{u}, V, E)$ in three sequential sub-blocks called edge, node, and global blocks, and gives an output graph with new features $G' = (\boldsymbol{u'}, V', E')$ as illustrated in Figure 4. The edge block updates features of each of the edges by a concatenation of the existing edge features, the features of connecting sender and receiver nodes and the global features of the graph with an update function ϕ^e. The following node block applies another update function ϕ^v to the input node and global features and the previously updated edge features that are aggregated to corresponding nodes with a message passing aggregation function $\rho^{e\rightarrow v}$. Finally, the global block uses the input global features and the updated edge and node features aggregated with separate functions $\rho^{e\rightarrow u}$ and $\rho^{v\rightarrow u}$, and an update function ϕ^u to provide new global features. Mathematically, this is expressed in Equation 5. The first equation indicates that, for each edge e_k, the updated feature vector ($\bar{e_k}'$) is computed by the function ϕ^e given its edge features, sender and receiver nodes' features, and the global features. Then, $\bar{e_k}'$ is aggregated to each sender/receiver node by the function $\rho^{e\rightarrow v}$ where $e \rightarrow v$ implies an *edge-to-node* aggregation, and each node is given its *edge* features $\bar{e_i}'$. The other equations work similarly with the corresponding update functions (ϕ^v and ϕ^u), and *edge-to-global* and *node-to-global* aggregation functions ($\rho^{e\rightarrow u}$ and $\rho^{v\rightarrow u}$). Example update and aggregation functions are MLPs and summation respectively.

$$\begin{aligned} \bar{e_k}' &= \phi^e(\bar{e_k}, \bar{v_{rk}}, \bar{v_{sk}}, \bar{u}) & \bar{e_i}' &= \rho^{e\rightarrow v}(\{\bar{e} \text{ for all edges connected to the node } i\}) \\ \bar{v_i}' &= \phi^v(\bar{e_i}', \bar{v_i}, \bar{u}) & \bar{e}' &= \rho^{e\rightarrow u}(\{\bar{e} \text{ for all edges}\}) \\ \bar{u}' &= \phi^u(\bar{e}', \bar{v}', \bar{u}) & \bar{v}' &= \rho^{v\rightarrow u}(\{\bar{v} \text{ for all nodes}\}) \end{aligned} \tag{5}$$

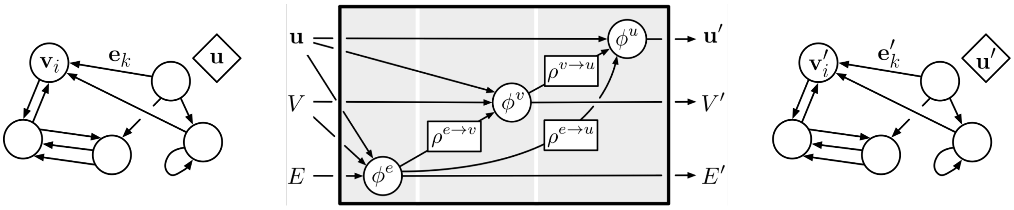


Figure 4: The graphical representations of an input data and the output (left and right) and the full GN block configuration (centre) presented in ref. [27]. It transforms the input graph $G = (\boldsymbol{u}, V, E)$ into the output graph with updated features $(\boldsymbol{u'}, V', E')$.

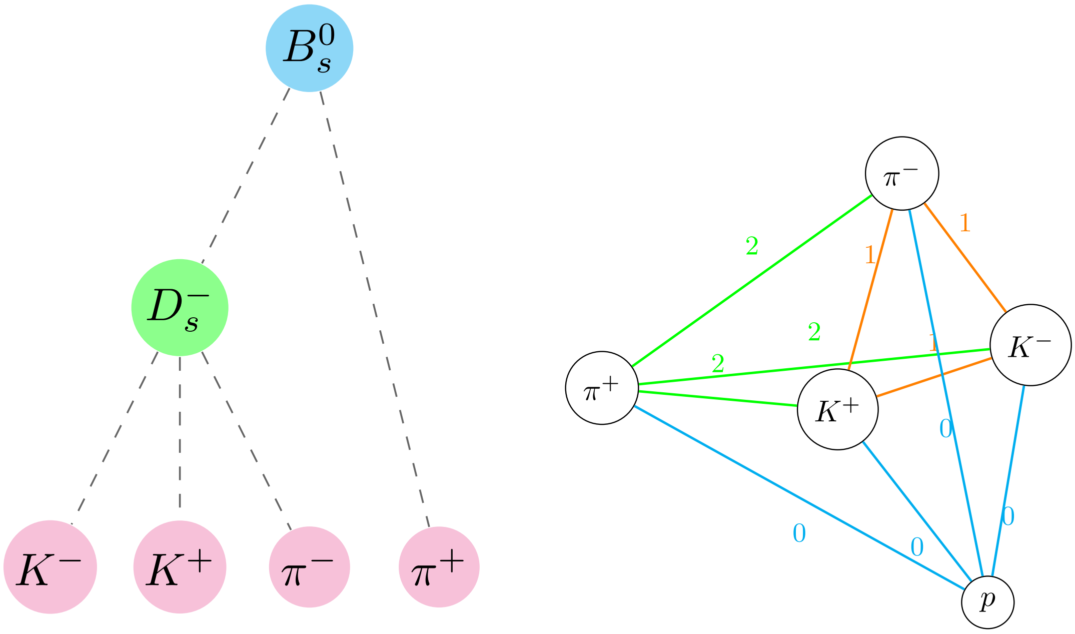
4 Method

4.1 DFEI

The Deep-learning based Full Event Interpretation (DFEI) is a new approach to analyse the entire event at the LHCb inclusively from the reconstructed particles with GNNs as introduced by ref. [4]. Prospective applications are selection at the trigger system and signal reconstruction or background rejection in offline analysis. The current trigger at the LHCb determines which particles in an event should be retained relative to the signal particles as described in Section 2.3. However, this requires tuned selections of several specific decay modes, which potentially limits the search of NP as events in which undiscovered particle decays exist might be rejected due to their unique decay topologies. The DFEI on the other hand provides an inclusive selection of any interesting decays systematically, whilst handling the growing multiplicity of particles in the event with the LHCb upgrades. The DFEI aims to reconstruct the hierarchical decay chains of heavy hadrons without pre-defining individual decay channels, thus if applied, it enables the trigger to identify clusters of particles associated with the interesting physics processes and discard the rest of the event. This capability to learn decay topologies is also beneficial in offline analysis where knowledge of its decay mode and assumptions such as the mass of the NP particle are needed for reconstruction of the decay processes.

As mentioned previously, the particles of interest which are produced in pp collisions can be observed only as the final decay products, for which lifetimes are long enough to travel through and interact with the detector. Therefore, the available input data for the NN model contains information of the stable particles and the task is to reconstruct the unobserved decay hierarchy from the stable final state decay products. The DFEI deals with this complex problem by exploiting an effective representation called the Lowest Common Ancestor (LCA), which is first presented for phase space simulation of particle decays in ref. [28] as the LCA Generation matrix. A LCA value is defined between a pair of the final state particles in the following manner. If the two particles do not originate from the same decay, this means no *common ancestor* particle exists and the LCA is 0. Otherwise, the LCA indicates the depth of the tree from the common ancestor which is closest to the leaves. A decay of B_s^0 is shown in Figure 5 as an example. The left tree diagram illustrates the hierarchical decay chain to be reconstructed by the DFEI. The leaves (K^-, K^+, π^-, and π^+) can be detected and the other nodes (B_s^0 and D_s^-) and the connections (dashed lines) are not visible at the detector level. In the input data, this decay is seen as a part of an event with the nodes of the reconstructed leaves as the right graph depicts. The LCA values as the edge labels represent the decay structure, for instance, the LCA of π^- and K^+ is 1 since the common ancestor is D_s^-, while that of π^- and K^- is 2 as the common ancestor is B_s^0. The proton (p) is either a random background or produced in another b-hadron decay ($\Lambda_b^0 \to \Lambda_c^+(\to pK^-\pi^+)\pi^-$), thus the LCAs between p and the other particles in the considered decay are 0. In practice, there are more of such particles of O(100) in an event and graphs are larger.

With the LCA definition, the problem of learning the decay chain can turn to predicting the LCAs of reconstructed particles. This is naturally adopted to a GNN edge classification as the input data can be represented as a fully connected graph for each event with reconstructed particles as the nodes and LCAs defining ground truth labels for edges in the graph. To handle this complexity, DFEI has developed a three step GNN architecture as Figure 6 schematically shows. Particle event data is first processed in a node pruning (NP) GNN, which employs a binary classification of nodes coming from b-hadrons or not and aims to remove most of the particles belonging none of the b-hadron decays. The remaining particles are the input of the second edge pruning (EP) GNN. It further reduces the graph size by performing a binary classification on edges. Edges which connect particles from the same b-hadron decay are classified as 1 and otherwise 0. The final, LCA inference (LCAI) module predicts the LCAs of edges using a multi-class classification with LCAs of up to 3. The core layers of the GNNs are comprised of the GN blocks (refer to Section 3.2) with independent MLPs as the update functions ϕ and sum aggregation functions ρ. The DFEI currently only considers charged particles.



(a) decay tree (b) input sub-graph

Figure 5: (a) an example b- hadron decay at the LHCb and (b) a part of an input event for the DFEI. In (a), only the leaves (K^-, K^+, π^-, and π^+) are visible at the detector. The numbers on the edges in (b) are the ground truth LCAs. The target of the DFEI is to reconstruct the decay tree by representing it as the graph and performing an edge classification.

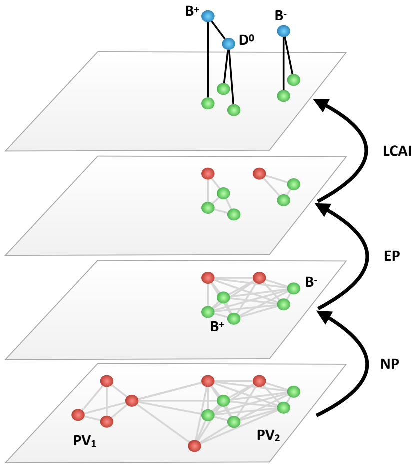


Figure 6: Illustration of an event processed through the three GNNs of the DFEI algorithm [4]. The background particles shown as the red nodes are largely removed in the NP GNN and most edges which do not connect particles from the same b-hadron decay are removed in the EP GNN. Finally, the b-hadron decays producing the particles which are represented as the green nodes are reconstructed by the LCAI. The particles in blue are the reconstructed ancestors and not visible in the data.

4.2 Model Architecture

The primary aim of this thesis is to extend the idea of the DFEI by developing a model capable of the NP, EP, and LCAI in a single GNN using an improved message passing paradigm. This new GNN is implemented in `PyTorch` [29] instead of `TensorFlow` for more flexibility in data loading and model structures, and current prevalence in the research community. Accordingly, the GN from ref. [27] is implemented in `PyTorch` as the initial step of this work.

4.2.1 Nominal GNN

The full model architecture is illustrated in Figure 7. The input node, edge and global features are first learned independently in the encoder GNN with no message passing layer and the outputs are passed to a sequence of multiple GN blocks. The skip connections between the encoder and the GN blocks allow the model to preserve some features in the original graphs and combine them with the outputs of the message passing GNN layers. Such skip connections also prevent oversmoothing, which is when multiple message passing updates cause features of the graph to become homogeneous. After several GNNs with message passing layers, the features of the updated graphs are decoded to final representation via a decoder GNN, which is structurally same as the encoder GNN. The final layer makes the predictions of LCA classes using the decoded edge representation, followed by the CEL to learn the model parameters. The CEL function in Pytorch (`torch.nn.CrossEntropyLoss` [30]) involves a log softmax within, hence no extra softmax activation is employed in the final layer in the GNN. The update functions corresponding to ϕ in Equation 5 in the encoder, GN blocks, decoder and final classifier are separate MLPs with ReLU activation functions from `torch_geometric.nn.models` [31]. The model is trained using Adam optimizer [32], which enables the parameter learning computationally efficient for gradient decent. This GNN is a nominal architecture for this thesis and referred as GNN(1) in the following sections.

4.2.2 GNN with weighted message passing

Further upgrades to the model are explored to improve the performance and the computational speed. The NP and EP in the original DFEI algorithm can be integrated into the single GNN by introducing learnable *weights* for edges and nodes in the graph, which are provided by new functions ψ^e and ψ^v in each GN block.

The function ψ^e takes the updated edges $\bar{e_k}'$ in Equation 5 or 6 as its inputs and assigns a single value between 0 and 1 (weight) to each edge. Similarly, ψ^v considers the updated nodes ($\bar{v_i}'$). The weights are then used in the aggregation of messages during message passing and Equation 5 is modified to 6 where w_e and w_v are the corresponding edge and node weights computed with the functions ψ^e and ψ^v. Schematically, this weighted message passing is shown in Figure 8. In an ideal scenario, the weights are exactly 1 if the edge is in non-zero LCA class and the node has at least one of those edges, and 0 otherwise. Regarding the nodes in the bottom graph of Figure 6 as an example, the model should predict the weights of 0 for the nodes in red and the weights of 1 for the nodes in green. This can be further enforced using BCEL terms for the weights, $\mathrm{BCEL}(y_e, w_e)$ and $\mathrm{BCEL}(y_v, w_v)$, thus they would estimate the probability of each edge being a non-zero LCA edge and the probability of each node connected to any non-zero LCA edges. The functions ψ^e and ψ^v are parametrized with their own independent MLPs and a final sigmoid layer in this thesis. The model is trained either with or without the BCEL constraints, which are referred as GNN(2) and GNN(3) respectively in the later sections.

In order to predict both the LCA structure and the node and edge message passing weights during the same gradient decent optimization, the CEL and BCEL need to be combined into a single loss. However, the two losses can be at different orders of magnitude, which might prevent the model learning the 4 LCA classes properly if a simple sum of the losses is taken. This is handled by introducing a hyperparameter α, which is any positive value and scales the BCEL terms. The total loss becomes $\mathrm{CEL}(y_{\mathrm{LCA}},\, p_{\mathrm{LCA}}) + \alpha \cdot \mathrm{BCEL}(y_e, w_e) + \alpha \cdot \mathrm{BCEL}(y_v, w_v)$ and this joint loss is to be minimized though the training process.

$$
\begin{aligned}
\bar{e_k}' &= \phi^e(\bar{e_k}, \bar{v_{rk}}, \bar{v_{sk}}, \bar{u}) & \bar{e_i}' &= \rho^{e\to v}(\{\bar{e}\cdot w_e \text{ for all edges connected to the node } i\}) \\
\bar{v_i}' &= \phi^v(\bar{e_i}', \bar{v_i}, \bar{u}) & \bar{e}' &= \rho^{e\to u}(\{\bar{e}\cdot w_e \text{ for all edges}\}) \\
\bar{u}' &= \phi^u(\bar{e}', \bar{v}', \bar{u}) & \bar{v}' &= \rho^{v\to u}(\{\bar{v}\cdot w_v \text{ for all nodes}\})
\end{aligned}
\tag{6}
$$

The weighted message passing does not alter the core structure of the GNN in Figure 7, but refines the message passing in each of the GN blocks. Each single GN block predicts message passing weights for nodes and edges with their own associated BCEL terms. As a consequence, the total loss is naturally larger for a deeper model where the BCEL terms become the sum of $\alpha \cdot \mathrm{BCEL}_i(y_e, w_e) + \alpha \cdot \mathrm{BCEL}_i(y_v, w_v)$ over all GN blocks i.

4.2.3 Extension to pruning

Ultimately a selection on the weights during inference after a given layer of the network would allow for a pruning effect similarly to the NP and EP GNNs in the DFEI and accelerate the computational time. Pruning of edges or/and nodes is implemented within a GN block in such a way that entities whose predicted weights are below a given threshold are removed from the graph. It can be activated on a GN block specified by the user during or after training and the shrunk graphs are fed through to the remaining layers. Once pruning is applied, the input graphs are also necessarily reduced as they are concatenated in the skip connection. Technically, pruning edges is more straightforward than nodes as edges connected to those removed nodes also need to be treated properly. Therefore, this thesis only examines edge pruning with node pruning left for future work due to time limitation. Further developments including node pruning is discussed in Section 7.

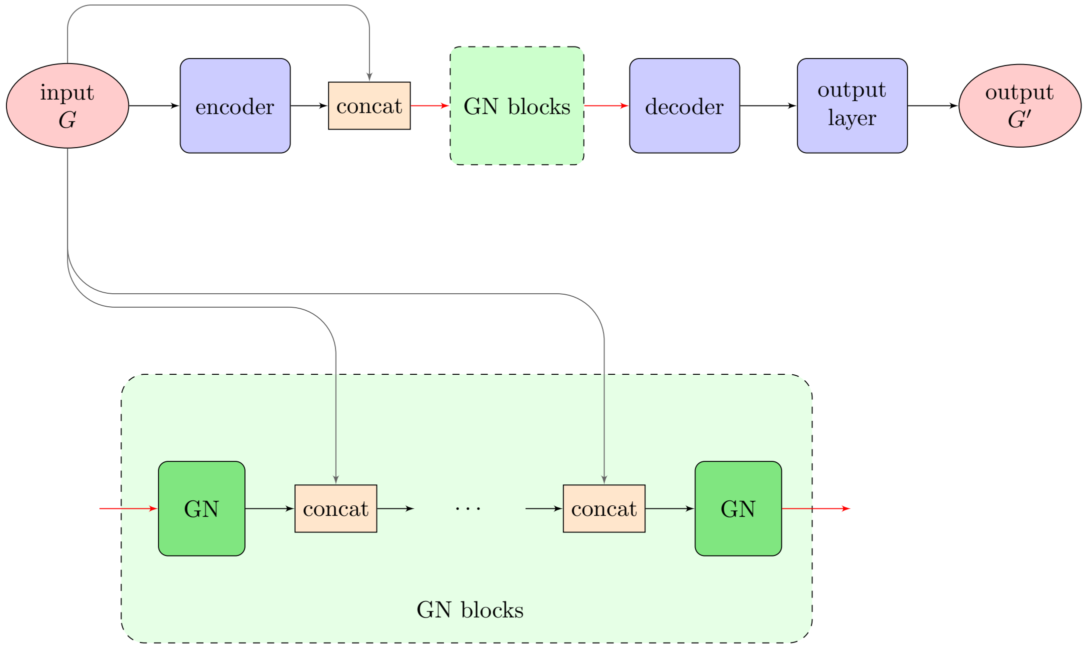


Figure 7: The full GNN architecture. The top flowchart depicts the main algorithm. The green rectangle labelled as "GN blocks" below is intended to show the details of the smaller green box in the main flowchart. The dots (. . .) implies that the number of GN blocks can vary.

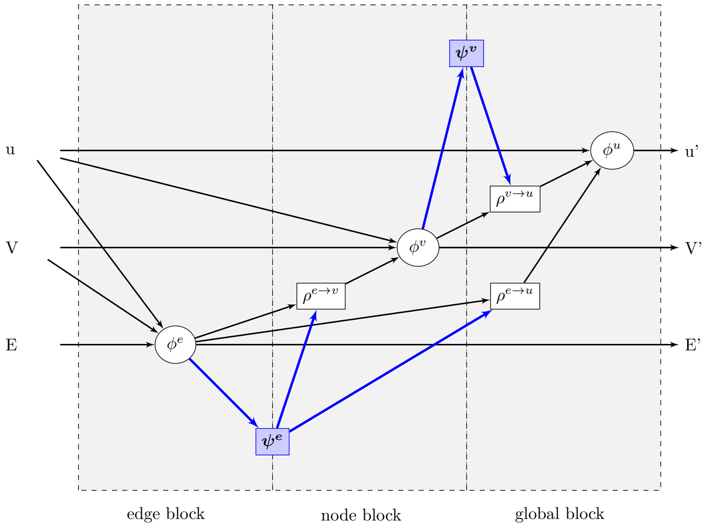


Figure 8: Figure 4 from ref. [27] is extended to include the new functions ψ for weighted message passing, which are represented in blue in this diagram. ψ^e and ψ^v are responsible for edge and node weights respectively. In this thesis, they are equal sized MLPs.

5 Training Setup

5.1 Dataset

Supervised learning requires true labels for model training. In high energy physics, simulation is an essential part of the physics analysis, and simulated samples can provide the ground truth of particle collision events. This work uses datasets that are produced with PYTHIA 8 [33] and EvtGen [34], which are commonly used programs for generating pp collisions and heavy hadron decays. The datasets contain pp collision events simulated in the expected environment during LHCb Run 3, likewise (but not identical to) the publicly accessible datasets on [35]. Each generated event includes at least one pp collision resulting in the production of a $b\bar{b}$ pair.

For the GNN model, an input graph represents an event in the dataset and its nodes are the reconstructed charged particles. Neutral particles such as photons and π_0 are not included in the graphs. A pair of nodes shares an edge only if they are associated to the same primary vertex or if their opening angle is smaller than 0.26 rad. The angle requirement is chosen so that 99% of edges of which the particle pair originating from a common b-hadron decay are kept in the dataset. Each graph has 10 node and 4 edge features as listed below and a global feature, which is the number of nodes in the graph.

Node features

- x_o, y_o, z_o: x, y, and z coordinates of the point from which the particle track originates
- x_{PV}, y_{PV}, z_{PV}: x, y, and z coordinates of the primary vertex associated to the particle
- p_x, p_y, p_z: the three-momentum of the particle
- q: the charge of the particle, limited to either positive or negative in this analysis

Edge features

- a boolean value containing information whether the particle pair shares the same associated primary vertex
- θ: the opening angle between the three-momentum of the two particles
- $d_{\perp\vec{O}}$: the distance between the origins of the two particles on a plane transverse to the vector combining their three-momentum
- Δ_z: the distance between the origins of the two particles in the z axis

The training and test datasets include 39488 and 9280 simulated events respectively. The datasets are highly imbalanced and dominated by the background processes (edges with LCA=0). On average, a given event has roughly 3700 , 5 , 15, and 3 edges with the true LCA=0, 1, 2, and 3, and 95 nodes among which around 10 nodes originate from b-hadron decays. This imbalance of classes is accounted for by the weighting of classes in the CEL as described in Section 3.1.

5.2 Model Hyperparameters

Hyperparameters involved in a supervised learning problem fall into two categories, those specific to the model architecture and more general parameters related to gradient decent optimization during the training process. The hyperparameters for this work are listed below. The GNN model specific parameters include the sizes of the MLPs and the number of GN blocks, while more general parameters are the learning rate, batch size, and number of epochs. As described in Section 3.1, the size of the MLP is determined by the numbers of hidden neurons or channels in each layer (C_{mlp}), of hidden layers (L_{mlp}), and of output neurons (O_{mlp}). Upon inclusion of the edge and node weight MLPs in the message passing, additional hyperparameters need to be considered. The two MLPs are generated with another set of C_{mlp}, L_{mlp} and O_{mlp}, and the binary classification on the weights includes a parameter α for the BCEL term. Finally, the threshold on the edge weights during edge pruning is given as k_w^e ranging from 0 to 1.

Hyperparameters defining gradient decent:

- $\boldsymbol{lr}$: Learning rate
- $\boldsymbol{\alpha}$: Coefficient to scale the contribution of the BCEL to the total loss
- Number of epochs
- Batch size

Hyperparameters defining the model architecture:

- $\boldsymbol{N_{blocks}}$: Number of the GN blocks
- $\boldsymbol{L_{mlp}}$: Number of hidden layers of each MLP in the GN blocks
- $\boldsymbol{C_{mlp}}$: Number of hidden neurons (channels) in each layer of the MLPs in the GN blocks
- $\boldsymbol{O_{mlp}}$: Number of output neurons of each MLP in the GN blocks
- $\boldsymbol{L_{mlp}^{e,v}}$: Number of hidden layers of each MLP for the message passing weights
- $\boldsymbol{C_{mlp}^{e,v}}$: Number of hidden neurons in each layer of the MLPs for the message passing weights
- $\boldsymbol{O_{mlp}^{e,v}}$: Number of output neurons of each MLP for the message passing weights
- $\boldsymbol{k_w^e}$: Edge pruning threshold

Through out the training presented in this thesis, the batch size is fixed to 32 and the other hyperparameters mentioned above are adjusted to seek an optimal performance. The sizes of the MLPs are the same within the encoder, decoder, and GN blocks but can be different in each complete training, meaning that, for instance, once the number of layers is increased from 4 to 8, all the MPLs in the GN blocks have 8 layers instead of 4. The two MLPs for the message passing weights share the same C_{mlp}, L_{mlp} and O_{mlp} values, which can differ from the other MLPs.

5.3 Hyperparameter Optimization

For hyperparameter optimization, an exhaustive search method called grid search is used. The method generates and trains the model with all configurations of a given set of hyperparameters and finds the optimal configuration by simply comparing the results. In this work, the grid search aims to minimize the loss of the test dataset at the last epoch for three model setups, GNN(1) which has unweighted message passing, and GNN(2) and GNN(3), which involve the weighted message passing with and without the BCEL terms. For each GNN, the architecture is first optimized while the number of epochs and the learning rate are fixed to 20 and 0.001 respectively. When applicable, a search for optimal α follows. Then, the architecture is fixed with the determined hyperparameters and a range of learning rates are tested. As Table 1 summarises, in the first step, 54 model configurations are considered and, next, 6 values for the learning rate and 13 values for α are evaluated one after another. The MLPs for message passing weights are set to have a single hidden layer with 16 neurons, which is relatively small and constant through the aforementioned optimization searches in order to reduce the search dimension. A wide range of α values are considered as it is fairly unique to this GNN and no initial assumption would be favourable. After all the searches and training of the optimal model, the value for k_w^e is selected based on the resulting weight predictions.

Hyperparameter	Values
No. of GN blocks	[2, 4, 6]
C_{mlp}	[32, 64, 128]
L_{mlp}	[2, 4, 6]
O_{mlp}	[8, 16]
α	[10, 8, 4, 2, 1, 0.8, 0.4, 0.2, 0.1, 0.08, 0.04, 0.02, 0.01]
learning rate	[0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001]

Table 1: Search for optimal hyperparameter values.

5.4 Evaluation Methods

In general, the number of hyperparameters relates to the number of learnable parameters and the model complexity. Increasing hyperparameters, and thereby the model complexity, enables the model to learn more complicated tasks, however, it can lead to an overfitting problem where the model fits statistical fluctuations in the training dataset and fails to learn the general properties of the data performing poorly on unseen datasets. On the other hand, if the values of the hyperparameters are too low, the model may be too simple, which leads to underfitting where the model does not fully capture the data in any dataset despite the consistent performances for different datasets. In the terms of machine learning, overfitting describes a model with low bias and high variance and an underfitting model is high bias and low variance.

In this thesis, the main methods to evaluate the performance of the model are loss and accuracy. The above argument indicates that a well trained model absent from under or overfitting will have a low loss and high accuracy in both training and test datasets. Accuracy is calculated for each of the LCA classes, as the total number of correctly predicted edges divided by the true number of edges in the class. The loss is the CEL or the combination of CEL and BCEL applied to the final outputs as defined earlier. In order to resolve overfitting problems, the CEL on the test dataset are also monitored even though it does not affect the learning process. The hyperparameter optimization focuses on the loss on the test dataset. As the model complexity largely depends on the number of GN blocks, the optimization includes a range of possible values for the GN blocks, which is helpful assess under and overfitting.

A confusion matrix is another common tool to visualise ML performance in classification tasks. It is a matrix, where the rows and columns correspond to true and predicted classes, and the matrix element of the i-th rows and j-th column is the number of instances belonging to the true class i and predicted to the class j. For an imbalanced data, a confusion matrix is often normalised over the rows for a better interpretability and each element represents the fraction of the predictions per target class. Its diagonal then corresponds to the accuracy.

Finally, for the model with the weighted message passing, the discrimination power of the node and edge weights on each GN block is examined by plotting the normalized distributions of the weights for correct and incorrect nodes and edges. This allows to inspect the effectiveness of the BCEL applied on the weights and the model ability to discriminate the entities from background and those of interest.

6 Results

In this section, the results of the grid search are first presented briefly and the three different scenarios are compared in terms of the losses and accuracies at the last epoch given numerically in a table. For purposes of clarity and ease, short descriptions of the GNN configurations follow.

- **GNN(1)**: The nominal GNN (with unweighted message passing)
- **GNN(2)**: The GNN with weighted message passing using learnable weights of edges and nodes and without constraints on the weights
- **GNN(3)**: The GNN with weighted message passing using learnable weights of edges and nodes and with constraints on the weights

The performance of GNN(3) using the optimized parameters are discussed in detail together with relevant plots of various performance metrics. The plots for the other architectures can be found in Appendix. The edge and node weights are discussed next. Finally, the impact of edge pruning is examined for GNN(3) while taking into account the other results and the efficiency is analysed.

6.1 Grid Search Optimization

The grid search determined the optimized hyperparameters summarised in Table 2. Interestingly, increasing the number of GN blocks is not necessarily promising for the first two scenarios, while employing 6 blocks, which is the maximum value for this search, seems to perform well for GNN(3). This can indicate the possibility of overfitting when the model is made more complex with more GN blocks but is not provided the extra information of the binary classes of edges. Also, learning at each GN layer using true edges and nodes in GNN(3) is likely to improve weighted message passing for the subsequent layer, thereby, the optimization resulted in a deeper model for GNN(3).

On the other hand, the selected value of L_{mlp} is smallest possible value, 2 for all GNNs, which hints that larger MLPs within the GN blocks are not always required to estimate the appropriate functional forms.

model	α	learning rate	N_{blocks}	L_{mlp}	C_{mlp}	O_{mlp}	performance
GNN(1)	N/A	0.001	4	2	128	8	A.1
GNN(2)	N/A	0.0005	4	2	128	16	A.3
GNN(3)	0.04	0.0005	6	2	128	16	6.3

Table 2: Sets of optimal hyperparameters determined by the grid search.

6.2 Comparison of different GNNs

Each of the GNNs is constructed with the corresponding optimal hyperparameters shown in Table 2 and trained for 30 epochs. 10 more epochs than the grid search are used as experiments found the CEL decreasing around epoch 20, while running 30 epochs is computationally expensive and too time consuming for a full hyperparameter optimization. Table 3 provides a simple comparison of the performances of the three GNNs as accuracies at the last epoch on the training and test datasets.

For all the LCA classes on both of the datasets, GNN(3) achieves higher accuracies than the other GNNs, with especially remarkable results for the LCA of 2. Further investigation also confirms that GNN(1) cannot reach to the same level of accuracy even if it is trained longer, which is motivated since A.1 indicates the accuracies are still increasing around epoch 30. Comparing to the nominal GNN, implementing weighted message passing seems to improve the performance, especially with BCEL, it results in the best performance as shown in the accuracies of GNN(3). In particular, GNN(3) produces 18% and 10% better accuracies for LCA 2 and LCA 3 respectively than GNN(1). Although the improvement in LCA 0 of 3% is small, it is still remarkable considering the fact that very large fraction of edges are LCA 0. Boosting this could ultimately lead to higher reconstruction performance as the model can separate particles from b-decays from the rest and focus more on learning the hierarchy structure of those interesting particles.

Looking at individual GNNs, some overfitting is concerned as some of the accuracies on the test set are slightly lower than those on the train set. However, there are also classes for which the test performances are better. These might indicate the trade off between different classes. In order to reduce the variance, standard regularization techniques such as dropout and weight penalties could be used in the future developments.

	Accuracy (%)							
	training set				test set			
LCA	0	1	2	3	0	1	2	3
GNN(1)	94.7	67.9	45.3	80.3	94.7	70.0	42.9	83.0
GNN(2)	95.5	71.7	46.7	80.5	95.3	72.5	44.1	81.1
GNN(3)	97.4	76.0	53.8	85.5	97.8	77.2	51.0	84.2

Table 3: Result summary of different GNN performances as the accuracy at the last epoch.

6.3 Loss, Accuracies and Confusion Matrix

As presented above, the grid search using GNN(3) determined the optimal hyperparameters of {learning rate, GN blocks, L_{mlp}, C_{mlp}, O_{mlp}} = {0.0005, 6, 2, 128, 16} and α=0.04. The curves in Figure 9 and 10 show the CEL and accuracies on the training and test datasets over 30 epochs. A slight overfitting is seen in the loss plot from approximately 20 epochs onwards, and in the notable differences between the training and test accuracies of the LCA 1 class. On the other hand, the LCA 2 and 3 accuracies on the test set are still increasing at the last a few epochs, therefore training it longer could improve the performances if the overfitting can be addressed by regularization techniques as discussed earlier.

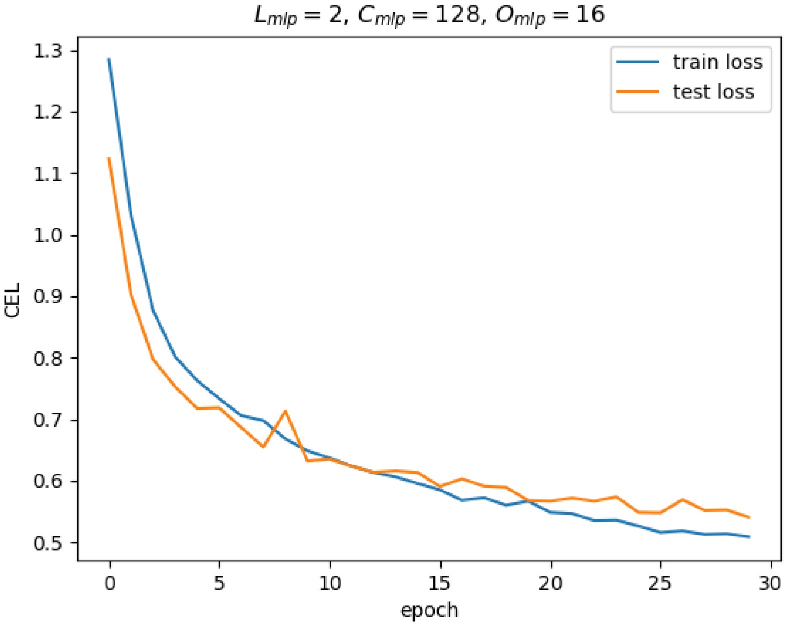


Figure 9: epoch vs CEL

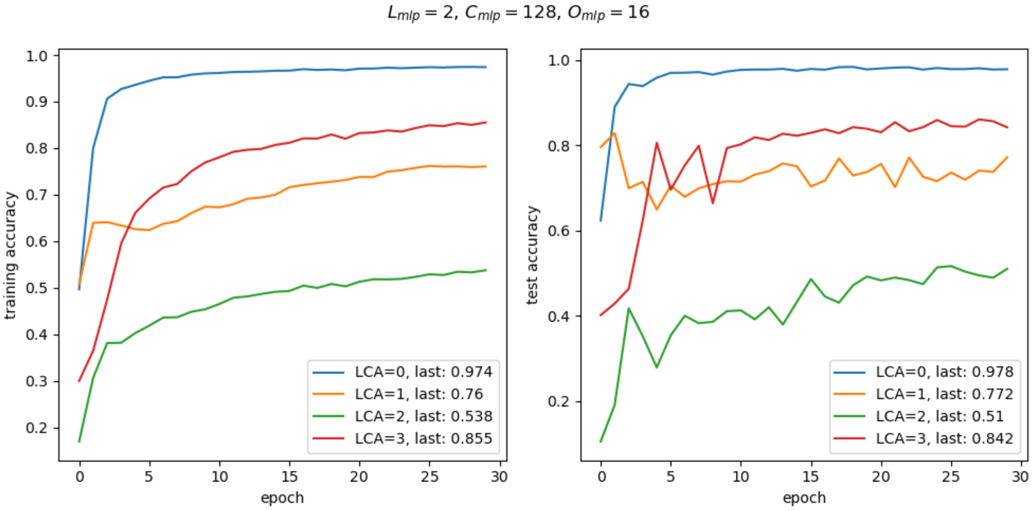
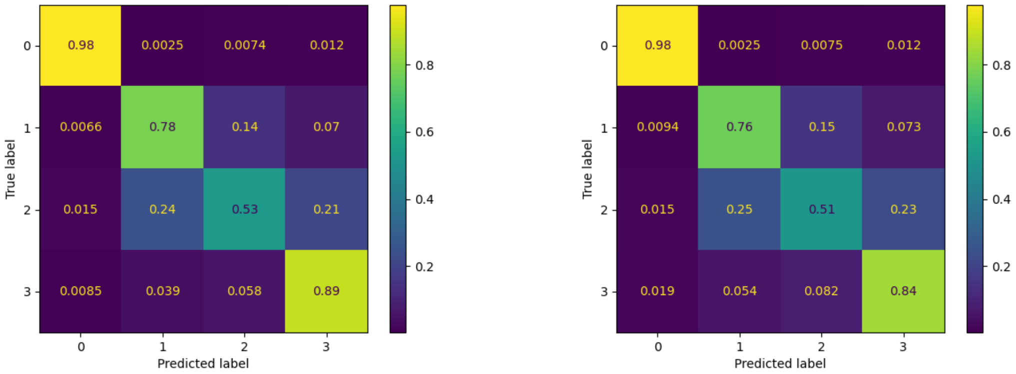


Figure 10: epoch vs accuracies

The trained model is used to produce normalized confusion matrices for the training and test datasets, which are displayed in Figure 11. It should be noted that the model parameters were updated regarding the training loss at the last epoch, therefore the performance on the training dataset shown in the matrix is slightly different from the previous training accuracies. (It might be considered as the training loss at epoch 31.) Comparing the two matrices, no remarkable discrepancy is observed, which confirms that generally the model provides the consistent performance across the different datasets. Overall, it is fairly successful with identifying edges of LCA 0, which need to be removed to have only particles originating from b-hadron decays. This is very important for the entire reconstruction performance as discussed above. The LCA 2 class seems the most challenging as the simple accuracy plots also indicated, and roughly half of the entities are misclassified to either LCA 1 or 3.



(a) Training dataset

(b) Test dataset

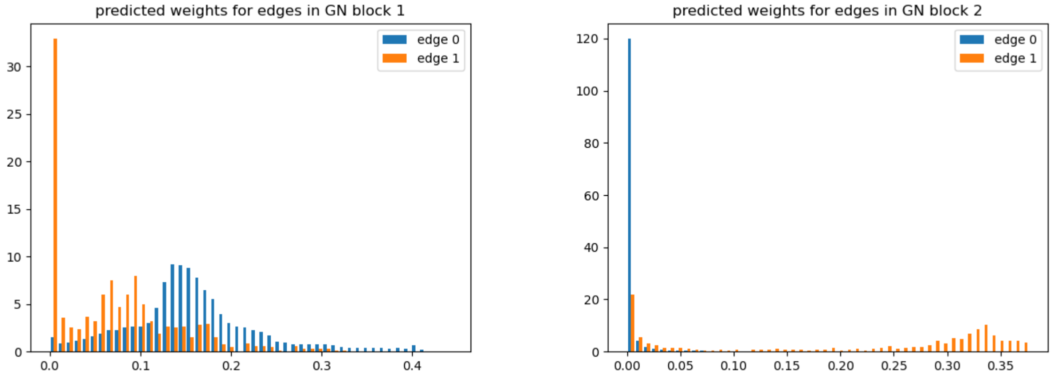
Figure 11: Confusion matrices for GNN(3) trained for 30 epochs. They are normalized over the rows and the diagonal corresponds to the accuracies. The rows and columns represent the true and predicted LCA classes respectively.

6.4 Edge and Node Weights

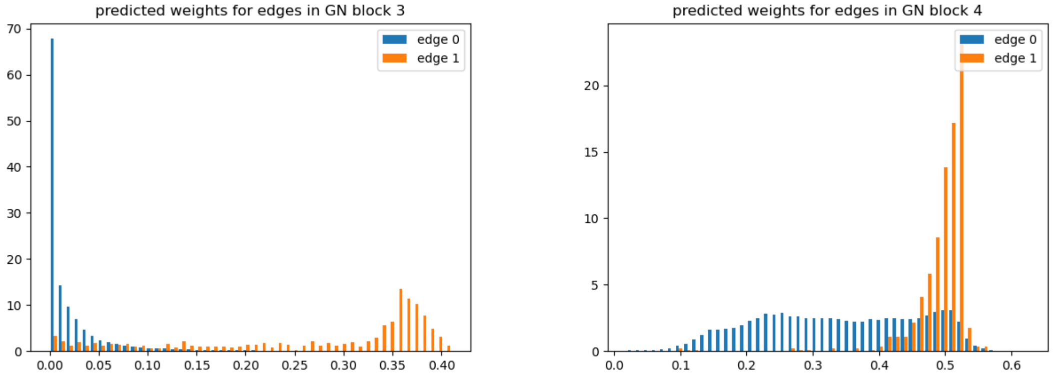
The optimized GNN(2) and GNN(3) provide weight predictions for edges and nodes respectively as well as the accuracies in Table 3. The weights are presented here in the form of histograms showing their distributions at each GN block for the test samples on the last batch. Therefore, the histograms can be understood as averaged distributions of the weights of 32 graphs on the batch. The two colours, blue and orange, in the figures are used to denote the true binary classes of edges and nodes as described in Section 4.2. It should be noted that the numbers of samples belonging to each class are normalized for convenience as the class 0 contains significantly more entities.

Figure 12 and 13 are for GNN(2), where no information about the true binary classes is provided and the only restriction on the edge and node weights is the softmax during the training. The MLP for the edge weights seems to differentiate the two classes in some degree, while for the node weights, it does not have a clear discrimination except on the block 3. However, no explanation for the behaviour of the block 3 can be inferred from those histograms.

Contrarily to GNN(2), GNN(3) involves the BCEL to constrain the weights in message passing, and thus performs a binary classification of edges and nodes in each GN block. This attempt is obviously successful as seen on both edge and node weights in Figure 14 and 15. Looking at them in more detail, notably more class 0 edges are predicted to be 0 than class 1 edges although non negligible amount of class 1 edges are also given to be or close to 0 for the first two blocks. The discrimination seems improved for deeper blocks. This is likely because the model learns more meaningful representations of the nodes, edges and graph as a whole through subsequent GN blocks. The improving discrimination sensitivity can be used to boost the pruning effects with increasing degrees of severity after each GN block is possible.

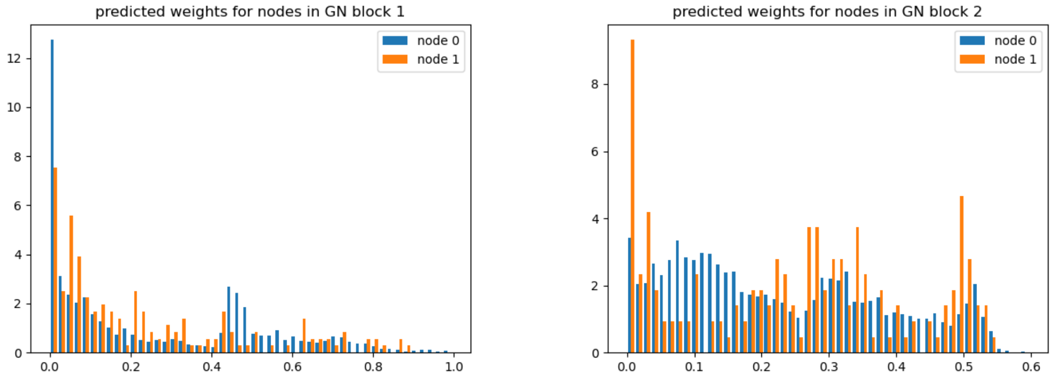


(a) block 1 (b) block 2



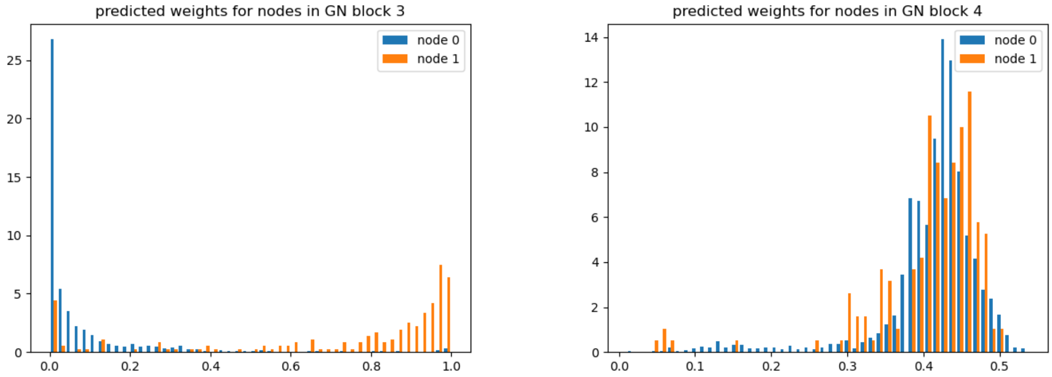
(c) block 3 (d) block 4

Figure 12: Normalised distribution of edge weights on each GN block in GNN(2).



(a) block 1

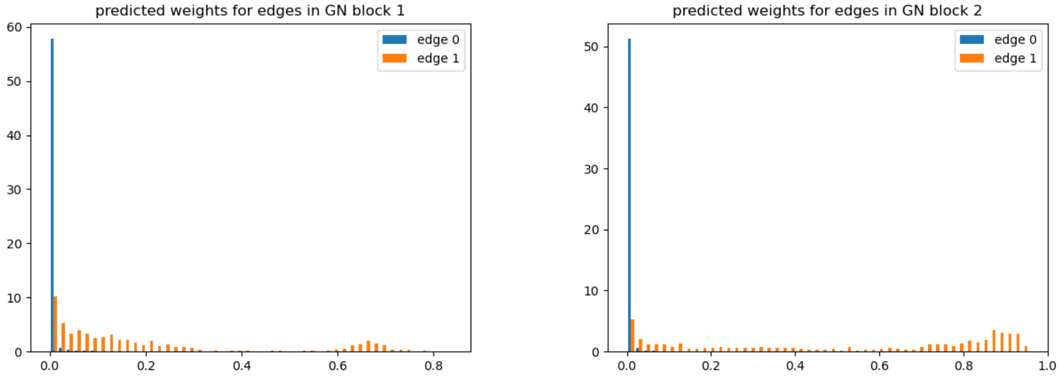
(b) block 2



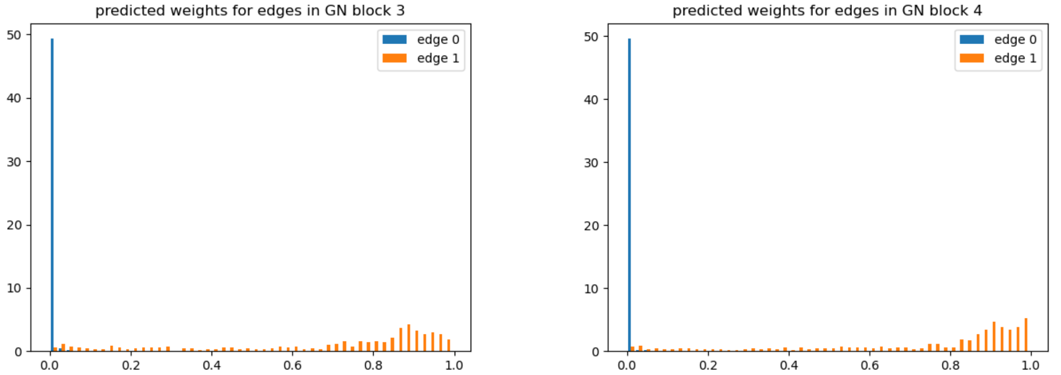
(c) block 3

(d) block 4

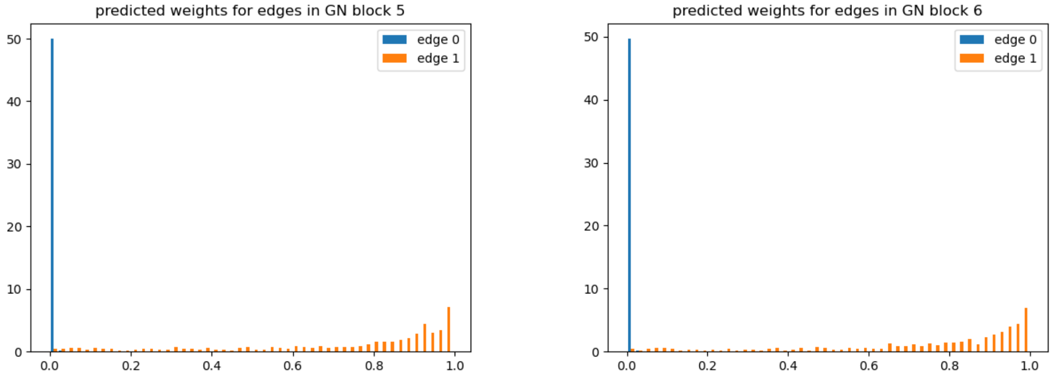
Figure 13: Normalised distribution of node weights on each GN block in GNN(2)



(a) block 1 (b) block 2

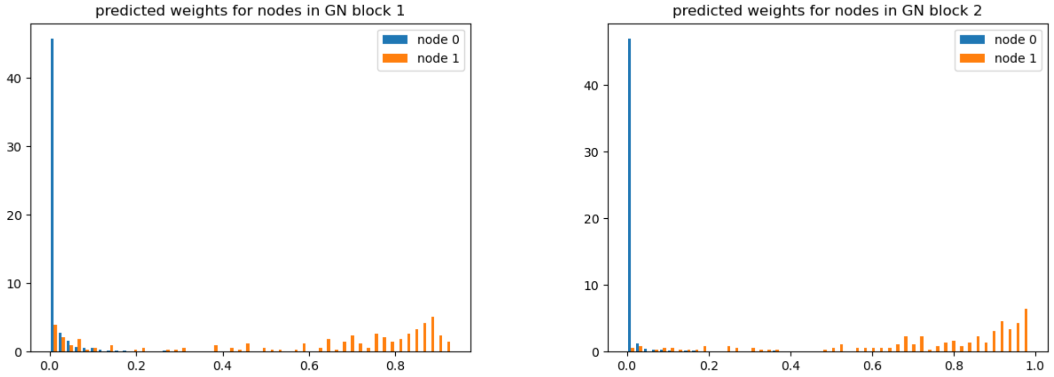


(c) block 3 (d) block 4

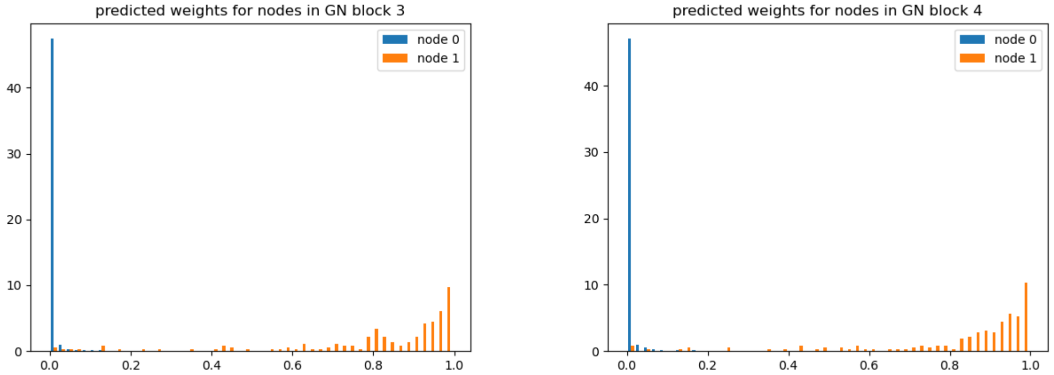


(e) block 5 (f) block 6

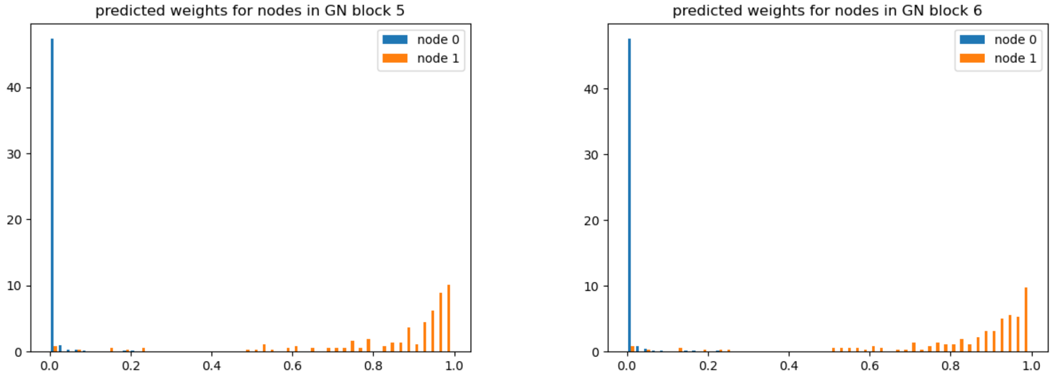
Figure 14: Normalised distribution of edge weights on each GN block in GNN(3).



(a) block 1 (b) block 2



(c) block 3 (d) block 4



(e) block 5 (f) block 6

Figure 15: Normalised distribution of node weights on each GN block in GNN(3)

6.5 Edge pruning

In order to ensure that the performance is not greatly lowered by pruning, the model needs to make sufficient predictions on the weights. As seen in Figure 14, the binary classification is not efficient in first a few GN blocks, therefore, pruning should be activated only in the later blocks. Here presents performance of edges pruning in the 4th GN block using the trained GNN(3) and the test dataset.

As a result of pruning, on average over 90% of edges are excluded, which is a huge reduction of the data size. Table 4 shows the performance of pruning with the previous results. The pruned edges are still accounted for in the accuracy calculation and are assumed to be predicted as LCA 0, otherwise, the accuracy of the LCA 0 class would be lower since only LCA 0 edges that are difficult to be separated from the edges of the other classes remain after pruning. The table shows that pruning does not significantly affect the performance in terms of the accuracies. A simple timing comparison determined the computational time is about 108% faster with pruning in testing. Although the time is not dramatically reduced, the consistent accuracies imply this approach is promising and further studies are desired. In particular, further speed up is expected through applying pruning on nodes as well as edges through earlier GN blocks at possibly a tighter threshold. It is equally possible to prune nodes and their associated edges, however due to time restrictions this is left for future work.

	Accuracy (%)							
	no pruning				with edge pruning			
LCA	0	1	2	3	0	1	2	3
GNN(3)	97.8	77.2	51.0	84.2	97.8	77.3	51.0	84.1

Table 4: Comparison of the accuracies between GNN(3) with and without edge pruning on the test dataset.

7 Conclusion and Outlook

This thesis presents a GNN model as an extension of the DFEI algorithm in the context of b-hadron decay hierarchy reconstruction at the LHCb experiment. The key idea of the GNN architecture and target definition for classification, Lowest Common Ancestor (LCA), in the DFEI are inherited to this work. Additionally, the new concept of learnable weights of edges and nodes is studied in order to implement the node pruning and edge pruning GNNs in the DFEI into the single GNN. Three scenarios of GNN are studied, a standard message passing GNN equivalent to that of the DFEI, a weighted message passing with learnable weights and weighted message passing with intermediate binary targets for weights using a binary cross entropy loss. For each scenario, the hyperparameters of the GNN are optimized with the grid search method. In general, the weighted message passing improves the performance of the model, however, the most significant increase of performance is seen when this is coupled with intermediate targets for the weights. While the message passing weights with no intermediate targets shows some discrimination between class 0 and class 1 edges (or nodes), only using intermediate targets gives an effective discrimination power for the two classes. Furthermore, the first attempt to apply edge pruning based on the predicted edge weights is presented in this thesis and the result confirms its functionality as it leads to no remarkable decrease in the accuracies. However, further studies including implementation of node pruning are required to conclude the usability.

Due to time constraints, there are several rooms for improvement left for future work in addition to the aforementioned implementation of node pruning. Further improvements in the model's performance are expected by increasing the number of training samples or providing more features in each event. For instance, if information of particle identification is included to each node, it can help the model as some decay topologies are restricted to particular final states. Besides, adding neutral particles in consideration is interesting as discussed in ref. [4] for the DFEI as well, but it is certainly more challenging due to growth in possible decay modes. Regarding the pruning function, there are various direction to test and refine its effectiveness. To start with, a range of the thresholds on the weights should be tested and optimized in terms of the accuracies and computational time. Other selection methods can be considered, such as a topk function, which extracts k largest elements of the given set.

Regardless of the need for continuing studies, overall, the thesis already demonstrates the weighted message passing GNN achieves an improved LCA reconstruction accuracy and potential inference speed up given the possibility of pruning edges and nodes in the graphs. The developments could eventually contribute to the trigger systems or various physics analysis at the LHCb.

A GNN performances

A.1 GNN(1)

The grid search for GNN(1) determines the optimal hyperparameters of {learning rate, GN blocks, L_{mlp}, C_{mlp}, O_{mlp}} = {0.001, 4, 2, 128, 8}. The model with those values is trained for 30 epochs and the performance is shown below as the plots of the CEL and accuracy.

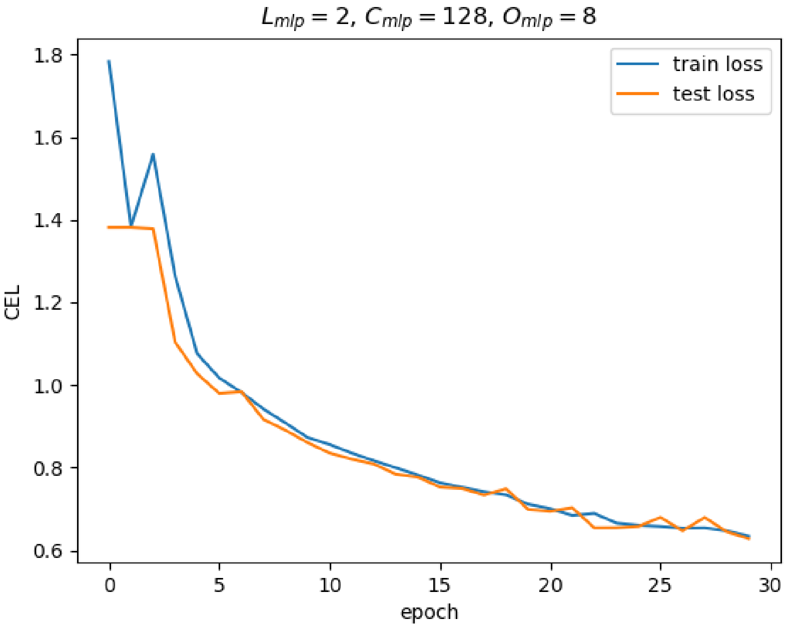


Figure 16: epoch vs CEL

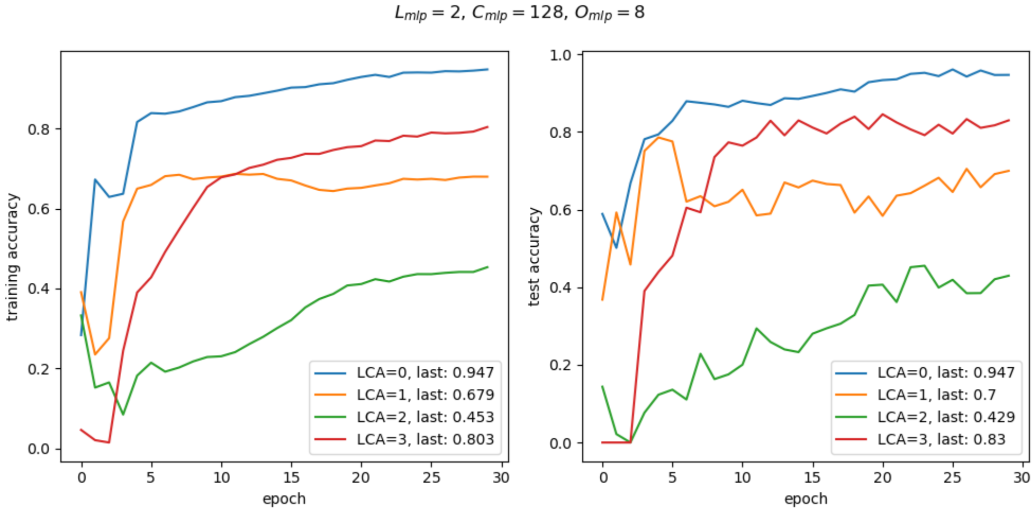


Figure 17: epoch vs accuracies

A.2 GNN(1) for longer training

GNN(1) with the same hyperparameters as above is trained for 60 epochs to investigate if the performance improves. The test loss seems to reach a plateau at epoch 40 or earlier and no dramatic improvement in the accuracies is achieved compared to the previous result of epoch 30.

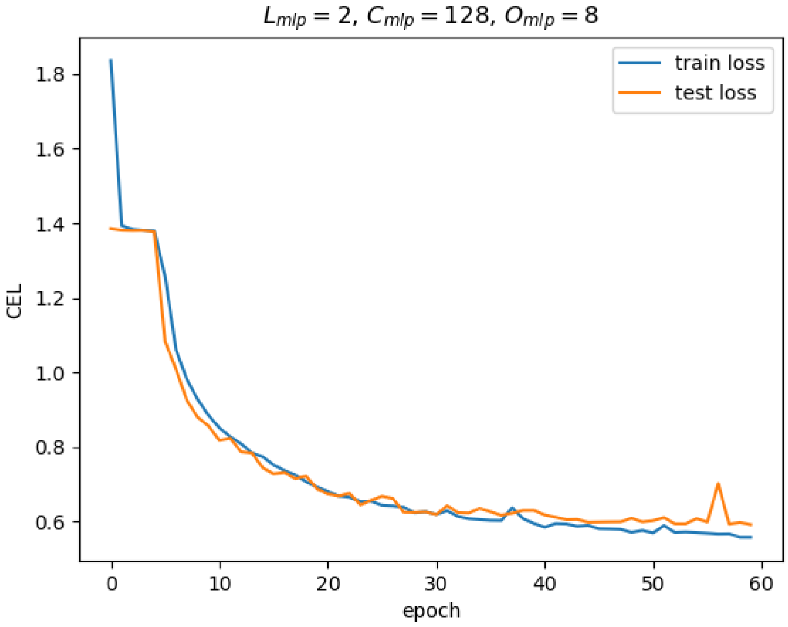


Figure 18: epoch vs CEL

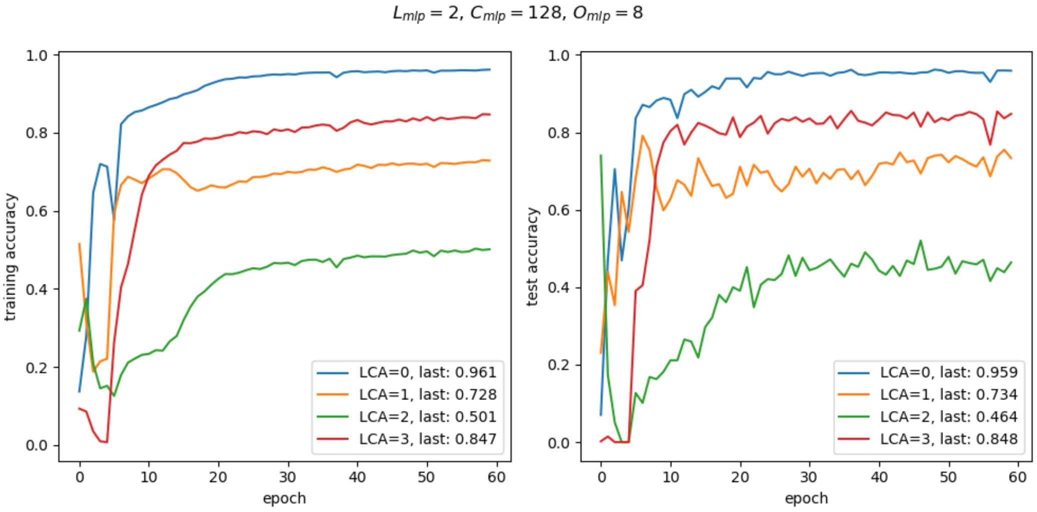


Figure 19: epoch vs accuracies

A.3 GNN(2)

The grid search for GNN(2) determines the optimal hyperparameters of {learning rate, GN blocks, L_{mlp}, C_{mlp}, O_{mlp}} = {0.0005, 4, 2, 128, 16}. The model with those values is trained for 30 epochs and the performance is shown below as the plots of the loss and accuracy.

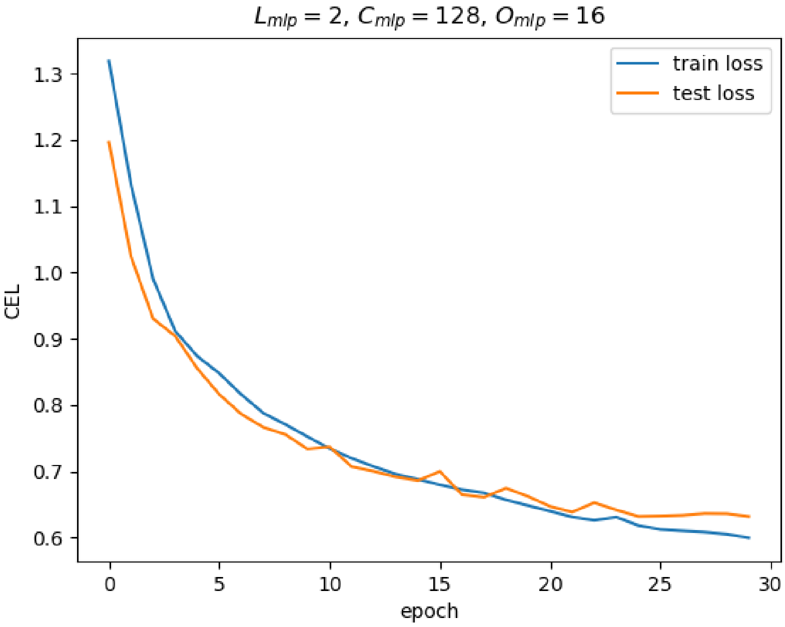


Figure 20: epoch vs CEL

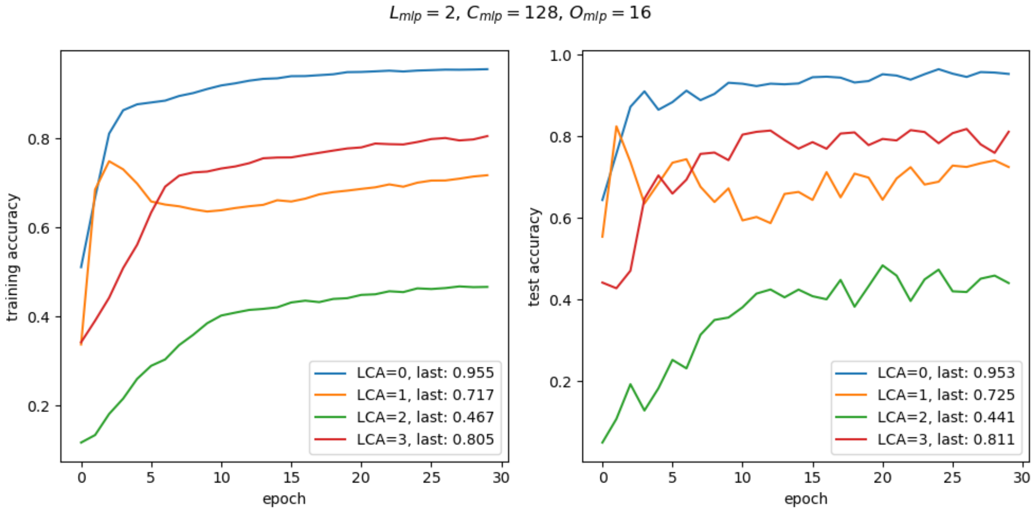


Figure 21: epoch vs accuracies

References

[1] I. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN COMPUT. SCI.*, vol. 2, no. 160, 2021. DOI: `https://doi.org/10.1007/s42979-021-00592-x`.

[2] K. Albertsson, P. Altoe, D. Anderson, *et al.*, *Machine learning in high energy physics community white paper*, 2019. arXiv: `1807.02876 [physics.comp-ph]`. [Online]. Available: `https://arxiv.org/abs/1807.02876`.

[3] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics," *Annual Review of Nuclear and Particle Science*, vol. 68, no. 1, pp. 161–181, Oct. 2018, ISSN: 1545-4134. DOI: `10.1146/annurev-nucl-101917-021019`.

[4] J. G. Pardiñas, M. Calvi, J. Eschle, *et al.*, *Gnn for deep full event interpretation and hierarchical reconstruction of heavy-hadron decays in proton-proton collisions*, 2023. arXiv: `2304.08610 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/2304.08610`.

[5] J. H. Christenson, J. W. Cronin, V. L. Fitch, and R. Turlay, "Evidence for the 2π decay of the K_2^0 meson," *Phys. Rev. Lett.*, vol. 13, pp. 138–140, 4 Jul. 1964. DOI: `10.1103/PhysRevLett.13.138`.

[6] J. Brodzicka, T. Browder, P. Chang, *et al.*, *Physics achievements from the belle experiment*, 2012. arXiv: `1212.5342 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/1212.5342`.

[7] T. L. Collaboration, A. A. A. Jr, L. M. A. Filho, *et al.*, "The lhcb detector at the lhc," *Journal of Instrumentation*, vol. 3, no. 08, S08005, Aug. 2008. DOI: `10.1088/1748-0221/3/08/S08005`. [Online]. Available: `https://dx.doi.org/10.1088/1748-0221/3/08/S08005`.

[8] E. M. Le Boulicaut, "The Standard Model - ATLAS Physics Cheat Sheet," General Photo, 2023. [Online]. Available: `https://cds.cern.ch/record/2851985`.

[9] L. Evans and P. Bryant, "Lhc machine," *Journal of Instrumentation*, vol. 3, no. 08, S08001, Aug. 2008. DOI: `10.1088/1748-0221/3/08/S08001`.

[10] T. Hadavizadeh, *Rare leptonic and semi-leptonic decays at lhcb*, 2024. arXiv: `2406.11108 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/2406.11108`.

[11] L. collaboration, R. Aaij, A. S. W. Abdelmotteleb, *et al.*, *The lhcb upgrade i*, 2023. arXiv: `2305.10515 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/2305.10515`.

[12] L. collaboration, I. Bediaga, M. C. Torres, *et al.*, *Physics case for an lhcb upgrade ii - opportunities in flavour physics, and beyond, in the hl-lhc era*, 2019. arXiv: `1808.08865 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/1808.08865`.

[13] M. S. Rudolph, "The LHCb Upstream Tracker Upgrade," *PoS*, vol. Vertex2019, p. 013, 2020. DOI: `10.22323/1.373.0013`. [Online]. Available: `https://cds.cern.ch/record/2747954`.

[14] R. Aaij, S. Benson, M. D. Cian, *et al.*, "A comprehensive real-time analysis model at the lhcb experiment," *Journal of Instrumentation*, vol. 14, no. 04, P04006, Apr. 2019. DOI: `10.1088/1748-0221/14/04/P04006`.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009.

[16] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: `1511.08458 [cs.NE]`. [Online]. Available: `https://arxiv.org/abs/1511.08458`.

[17] R. M. Schmidt, *Recurrent neural networks (rnns): A gentle introduction and overview*, 2019. arXiv: `1912.05911 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1912.05911`.

[18] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: `1706.03762 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1706.03762`.

[19] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. DOI: `https://doi.org/10.1038/323533a0`.

[20] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2020.07.061`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0925231220311693`.

[21] S. Thais, P. Calafiura, G. Chachamis, *et al.*, *Graph neural networks in particle physics: Implementations, innovations, and challenges*, 2022. arXiv: `2203.12852 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/2203.12852`.

[22] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, 2005, 729–734 vol. 2. DOI: `10.1109/IJCNN.2005.1555942`.

[23] W. W. L. Nuijten and V. Menkovski, *Node classification in random trees*, 2024. arXiv: `2311.12167 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2311.12167`.

[24] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2017. arXiv: `1609.02907 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1609.02907`.

[25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2018. arXiv: `1710.10903 [stat.ML]`. [Online]. Available: `https://arxiv.org/abs/1710.10903`.

[26] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*, 2021. arXiv: `2104.13478 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2104.13478`.

[27] P. W. Battaglia, J. B. Hamrick, V. Bapst, *et al.*, *Relational inductive biases, deep learning, and graph networks*, 2018. arXiv: `1806.01261 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1806.01261`.

[28] J. Kahn, I. Tsaklidis, O. Taubert, *et al.*, "Learning tree structures from leaves for particle decay reconstruction," *Machine Learning: Science and Technology*, vol. 3, no. 3, p. 035 012, Sep. 2022, ISSN: 2632-2153. DOI: `10.1088/2632-2153/ac8de0`.

[29] A. Paszke, S. Gross, F. Massa, *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: `1912.01703 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1912.01703`.

[30] PyTorch Contributors, *Cross entropy loss*, `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`, Accessed: 21-07-2024.

[31] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric*, 2019. arXiv: `1903.02428 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1903.02428`.

[32] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: `1412.6980 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1412.6980`.

[33] C. Bierlich, S. Chakraborty, N. Desai, *et al.*, *A comprehensive guide to the physics and usage of pythia 8.3*, 2022. arXiv: `2203.11601 [hep-ph]`. [Online]. Available: `https://arxiv.org/abs/2203.11601`.

[34] A. Ryd, D. Lange, N. Kuznetsova, *et al.*, “EvtGen: A Monte Carlo Generator for B-Physics,” May 2005.

[35] J. G. Pardiñas, M. Calvi, J. Eschle, *et al.*, *Dataset of paper ”GNN for Deep Full Event Interpretation and hierarchical reconstruction of heavy-hadron decays in proton-proton collisions”*, version v1.0.0, Zenodo, Apr. 2023. DOI: `10.5281/zenodo.7799170`.

[36] T. Keck, F. Abudinén, F. U. Bernlochner, *et al.*, “The full event interpretation: An exclusive tagging algorithm for the belle ii experiment,” *Computing and Software for Big Science*, vol. 3, no. 1, Feb. 2019, ISSN: 2510-2044. DOI: `10.1007/s41781-019-0021-8`.

[37] A. Mathad, *Recent highlights from the lhcb experiment*, 2023. arXiv: `2308.02390 [hep-ex]`. [Online]. Available: `https://arxiv.org/abs/2308.02390`.

[38] S. I.H, “Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions.,” *SN COMPUT. SCI.*, vol. 2, no. 420, 2021. DOI: `https://doi.org/10.1007/s42979-021-00815-1`.

[39] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015, ISSN: 0893-6080. DOI: `10.1016/j.neunet.2014.09.003`.

[40] I. Tsaklidis, P. Goldenzweig, I. Ripp-Baudot, J. Kahn, and G. Dujany, “Demonstrating learned particle decay reconstruction using graph neural networks at belleii,” Presented on 19 06 2020, Ph.D. dissertation, Strasbourg, Universitè de Strasbourg, Karlsruhe, Strasbourg, 2020.

[41] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[42] S. R. Qasim, J. Kieseler, Y. Iiyama, and M. Pierini, “Learning representations of irregular particle-detector geometry with distance-weighted graph networks,” *The European Physical Journal C*, vol. 79, no. 7, Jul. 2019, ISSN: 1434-6052. DOI: `10.1140/epjc/s10052-019-7113-9`.

[43] PyTorch Contributors, *Binary cross entropy loss*, `https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html#torch.nn.BCELoss`, Accessed: 25-07-2024.

[44] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.