Faculté de physique et ingénierie
Université de Strasbourg

master Physique Subatomique Astroparticules

Master Science, Mention Physique
Spécialité Physique Subatomique et Astroparticules

Année universitaire **2019-2020**

Ilias TSAKLIDIS

DEMONSTRATING LEARNED PARTICLE DECAY RECONSTRUCTION USING GRAPH NEURAL NETWORKS AT BELLE II

Rapport de stage de Master
sous la direction de
Pablo Goldenzweig
et
Isabelle Ripp-Baudot

01 Mars 2020 au 11 Juin 2020

KIT Karlsruhe Institute of Technology

IPHC Institut Pluridisciplinaire Hubert CURIEN STRASBOURG

Belle II

To my lovely sister,
and the bright future ahead of her.

Abstract

The clean environment within Belle II, with decay processes originating from non-composite electron-positron pairs, allows for the reconstruction of the entire collision event. Having precise information about the initial state kinematics gives a unique advantage to the Belle II experiment in that allows for direct measurements of decay processes involving neutrinos or few detectable particles in the final state, due to conservation of energy and momentum. This does, however, require a catch-all reconstruction algorithm which is able to determine which particles are not associated to the signal B-meson and reconstruct a second B-meson from them. The current full event reconstruction algorithm at Belle II requires the reconstructed sub-decay processes to be hard-coded, and the careful selection of which kinematic variables to exploit. This both restricts the total branching fraction coverage of the algorithm and relies on intuition to decide which decay processes to reconstruct and how to reconstruct them. This work introduces a method for learning which processes to reconstruct and how to reconstruct them from example using simple kinematic variables with graph neural networks. The efficiency of the proposed method is demonstrated by reconstructing decays from generic and Belle II phasespaces in numerous kinematic scenarios of increased complexity.

Abstract

L'environnement claire du Belle II, avec tous les processus de désintégration qui proviennent de pairs électron-positron non composite, permet la reconstruction d'évènement de la collision entier. La possession des information precises pour la cinématique de l'état initial, donne un avantage unique à l'experience de Belle II qui autorise la mésure direct de désintegration qui contient des neutrino ou peu particules détectable à l'état final, en raison de la conservation d'énergie et de l'impulsion. Ce processus demande un algorithm de reconstruction, qui prend en compte tout les particules détectés , capable à déterminer quelles particules ne provient pas du B-méson du signal et reconstruit un deuxième B-méson avec eux. L'algorithm de reconstruction d'évènement complet actuel à Belle II nécessite que les processus de sous-désintégration reconstruit sont explicitement programmer cas par cas et soigneusement choisir les variables cinématique à exploiter. Ce processus d'un part il restreint le rapport de branchement total du range de couverture de l'algorithm et d'autre part la décision pour quelle désintégration vont se reconstruire et comment se fait intuitivement. Dans cet projet nous introduisons une méthode pour apprendre quel processus à reconstruire et comment le reconstruire par l'exemple, en utilisant des simples variables cinématique à l'aide de graphe des resaux neuronal. Nous avons démontré l'effficacité de notre méthode en reconstruisant des désintégration par des éspaces de phase generique et par le Belle II dans plusieurs scenario cinématique avec une complexité croissante.

Acknowledgments

I would like to express my gratitude firstly to both my supervisors, Dr Pablo Goldenzweig and Dr Isabelle Ripp-Baudot. To Pablo for being one of the nicest and most supportive people I've ever met in my life and to Isabelle for believing in me and helping me since the very beginning. My sincere thanks goes to Dr James Kahn, for being the best mentor a student could ever ask for. James introduced me to the beautiful world of deep learning, spent endless hours explaining and correcting my mistakes, and showed me how to approach every single aspect of my work. This thesis could not have been written without James's guidance. I would also like to thank Dr Giulio Dujany for all our interesting discussions and for always reminding me to question every single step I made. A big thanks goes to Dr. Markus Goetz and Dr Oskar Taubert for their technical support and useful insights on deep learning. Lastly I would like to thank all the members of the Belle II group at KIT, and especially Tobias Boeckh for warmly welcoming me from the first day, and the members of the Belle II group in Strasbourg for our excellent remote collaboration the last 3.5 months.

This project could not have been completed without the love and support of many dear friends, who I'm extremely lucky to have in my life. To begin with, I'd like to thank my friend and flatmate Achilleas for being the person who motivated me to move on and made me believe in myself during difficult times, and for always supporting my decisions even though sometimes they were against his own interest. I'd also like to thank my friend Kai simply for making life more interesting just by being around. A big thanks goes to my friends Clémence and Ioanna for their love and support. Lastly I'm grateful to my dear friends Jonas, Gerrit and Simon for their incredible hospitality and the good times we had together.

Last but not least I'd like to thank the QMat Graduate school and its members in Strasbourg for supporting financially all my projects during this year.

Acronyms

SM Standard Model

QFT Quantum Field Theory

HEP High Energy Phycics

FEI Full Event Interpretation

ML Machine Learning

DL Deep Learning

NN Neural Network

MLP MultiLayer Perceptron

FFN Feed Forward Network

GNN Graph Neural Network

FSP Final State Particle

LCA Lowest Common Ancestor

NRI Neural Relational Inference

MC Monte Carlo

HPO HyperParameter Optimization

Contents

1. Introduction

Physics is the branch of science that describes nature by creating mathematical theories to predict physical phenomena. The two fundamental theories that allow us to understand the Universe are General Relativity, which describes Physics on a large scale, and the Standard Model of Particle Physics, which describes the Universe at an elementary level. In recent decades there has been an outstanding experimental verification of the predictions the Standard Model [1]. However there are still open questions that demand answers, such as the origin of the CP violation in the Universe or the nature of dark matter [2].

The Belle II experiment investigates Physics at an elementary level by colliding electrons and their anti-particles, positrons, at the SuperKEKB collider in Tsukuba, Japan. The main goal of the Belle II experiment is to make precision measurements related to flavour Physics and CP violation. The particles that are produced in the collisions at Belle II are primarily B mesons which decay very fast. The state-of-art of detector, hardware, and software technologies of Belle II are needed to study these decays with high precision.

This thesis focuses on the development of novel software tools necessary for the precision measurements that Belle II is aiming for. Specifically, a new Deep Learning algorithm is developed to learn how to reconstruct particle decays from example, using the kinematic information of the particles that have been detected, as a from of B-tagging. Efficient B-tagging in Belle II is necessary to reconstruct the signal B-mesons and study the Standard Model and its extensions in detail.

This thesis is divided in the following sections: The scientific context and the deep learning motivation is discussed in chapter 2. The outline of this thesis is presented in chapter 3. The results are discussed in chapter 4, while a conclusion and the outlook of this work is given in chapter 5

This thesis has been written under the exceptional conditions that everyone has experienced during the last few months. Luckily the outcome of this work has been only slightly affected by the pandemic situation. The first part of this work (3 months) has taken place at the Karlsruhe Institute of Technology (KIT) while it is currently being continued at the University of Strasbourg and the Institut Pluridisciplinaire Hubert CURIEN (IPHC). The initial targets of this thesis as they have been set by its supervisors were: the modification of data production procedures to make them appropriate for a graph-network-based approach, the implementation from the literature of an appropriate Graph Neural Network architecture for particle reconstruction, the evaluation of the capacity of the network in the context of particle reconstruction, and the extension of the training data to more complex reconstruction requirements.

2. Scientific Context

2.1 The Standard Model of Particle Physics

The Standard Model (SM) of Particle Physics is a Quantum Field Theory (QFT), that describes the nature of the elementary particles of the Universe and their interactions. Matter in the Universe, as described by the SM, consists of 12 elementary particles of spin $\frac{1}{2}$ which are called fermions and their anti-particles. Interactions between these elementary particles manifest themselves as particles of spin 1, the gauge bosons, which mediate the Electroweak and the Strong Force. Massive gauge bosons acquire mass from the Higgs scalar field. Figure 2.1 shows the particle content of the SM. The foundations of the SM are elegant ideas about symmetries of Nature, that lead to conservation of associated quantities and the fusion of Quantum Mechanics and Special Relativity. The remarkable consistency with almost all the available experimental data and the exquisite unified picture it provides, make the SM one of the greatest triumphs of modern Science [3].

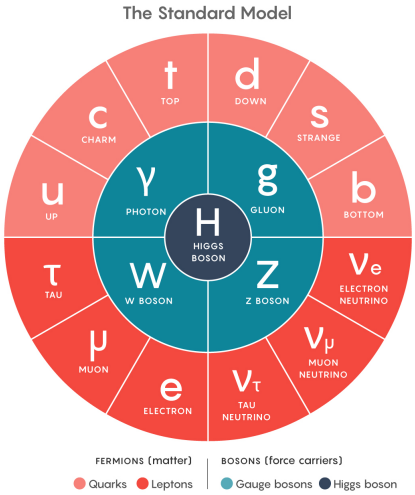


Figure 2.1: The particle content of the SM [4]. There are 8 gluons that mediate the Strong Force. Two W bosons (with positive and negative charge) and a Z boson that are carriers of the Weak Force. There are twice as much fermions since every particle has an anti-particle as well. The Higgs boson is a spinless and neutral particle.

Even though scientists managed to answer some of the very fundamental questions about the Universe in the last decades, there's still further inquiry as to the details of the SM. For instance the theory has 18 free parameters, that cannot be predicted, including structure constants, the three mixing angles and the CP violating phase δ_{13}, or the two Higgs potential parameters. There are further conceptual questions such as the origin of CP-violation in the Universe, the nature of Dark matter, or whether there is an energy scale that the 4 fundamental forces were unified, that cannot be explained within the current version of the SM. These open problems drive the pursuit of New Physics (NP) by studying and trying to identify physical processes beyond the SM. [5]

The tremendous progress in accelerator technology gives hope that the answers should not be far away. Addressing these questions is possible at two main frontiers.

1. *The Energy Frontier* where the highest possible energy is reached in order to produce -unseen yet- heavier particles. Undoubtedly the biggest achievement in this domain is the discovery of the Higgs boson in 2012 at the LHC.

2. *The Intensity Frontier* where ultra-high luminosity gives the opportunity to compare experimental data with precise theoretical calculations. Major discrepancies due to quantum fluctuations indicate New Physics (NP) in specific energy scales. The confirmation of the existence of the CP violation, as described by the CKM matrix, was possible only after exhaustive and precise measurements at SLAC and KEK during the first decade of the 21st century.

2.2 The Belle II Experiment

SuperKEKB is a particle collider located at KEK in Tsukuba, Japan and is an upgrade of the KEKB-B-Factory. It is an asymmetric-energy electron -positron collider that runs on the $\Upsilon(4S)$ resonance with a 7 GeV electron and a 4 GeV positron beams and aims to deliver 55 billion $B\bar{B}$ pairs (50 ab^{-1}) by 2027. One of the main advantages of electron-positron colliders such as the SuperKEKB is that they don't suffer from pile-up or any underlying event induced background like hadron colliders[6]. This gives the opportunity to know the 4-momentum of one B meson and the tracks associated with it, as soon as the other B-meson originating from same $\Upsilon(4S)$ resonance is fully reconstructed. Figure 2.2 shows the principle of B-tagging schematically. If the signal B meson candidate is reconstructed, then all the remaining final state particles of the event must be attributed to the tag B meson and vice-versa. Both of the decay chains need to be valid and the B mesons need to be of opposite flavour. This reduces significantly the number of eligible possible decays trees [7]. Furthermore the admittedly ingenious asymmetric setup establishes a large enough Lorentz boost for the e^+e^- system, allowing the B and D mesons to travel trackable distances before they decay inside the detector. The aforementioned configuration is very convenient for conducting precision measurement of important quantities, such as CP violation parameters or lifetimes, even on the rarests processes that are predicted by the SM [8].

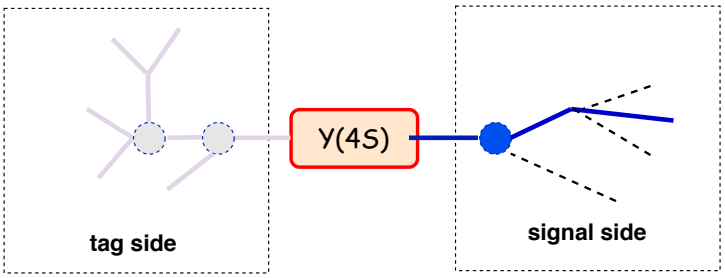


Figure 2.2: The $\Upsilon(4S)$ resonance decays exactly into a $B\bar{B}$ pair with a branching fraction of nearly 100%, providing a very clean experimental environment. The final state particles of the signal and the tag sides are overlapping spatially making the task of separating the two sides non-trivial[7]

.

The Belle II detector at SuperKEKB is the state-of-the-art experimental tool to study the Physics of B-mesons. The main parts of the Belle II detector that endorse the detection of the decay products of the B-mesons are namely:

1. The **Vertex Detector (VXD)** that consists of two devices; the Silicon Pixel Detector (PXD) and the Silicon Vertex Detector (SVD) and grants the efficient reconstruction of vertices very close to the collision point.

2. The **Central Drift Chamber (CDC)** that serves as the main tracking device of the detector. The CDC is a drift chamber composed of sense wires within a gas filled compartment , placed in a 1.5 T magnetic field that is produced by a superconducting solenoid.

Charged final state particles are in principal reconstructed as tracks in the CDC and the VXD.

3. Two Cerenkov detectors, a **time-of-propagation (TOP)** counter and the **Aerogel Ring Imaging CHerenkov (ARICH)** provide particle identification in the barrel and end-cap regions respectively.

4. The **Electromagnetic Calorimeter (ECL)** consists of CsI(Tl) scintillation crystals. The ECL is used for gamma detection and the separation of electrons from pions. Furthermore neutral final state particles deposit energy in the ECL, formatting clusters, that are used to detect and identify them.

5. K_L and μ are identified by resistive plate chambers that are placed in the **K_L-Muon Detector (KLM)** in the barrel (BKLM) and end-cap (EKLM) regions.

More information about the detector system can be found in official documents of the Belle II collaboration such as [9]. The Belle II detector is depicted in Figure 2.3

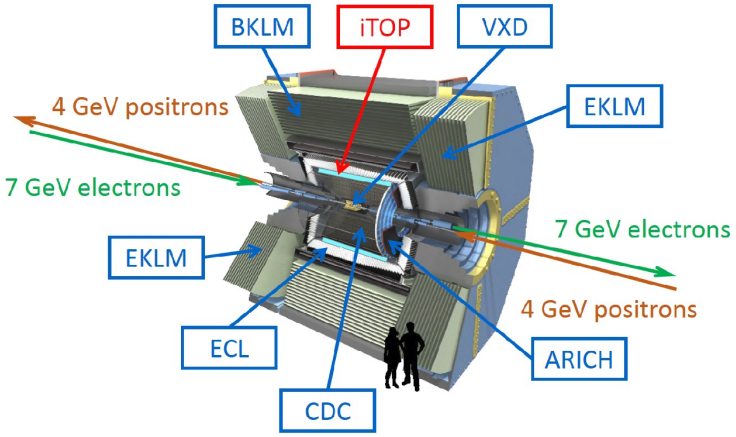


Figure 2.3: The Belle II detector with its main components.

2.3 Full Event Interpetation

In order to exploit the relatively clean experimental environment in Belle II, the Full Event Interpretation (FEI) algorithm is widely used for event reconstruction. The FEI algorithm uses multivariate classifiers (Boosted Decision Trees specifically [10]) to reconstruct decay chains in a hierarchical manner. Beginning with clusters, tracks, and vertices FEI first reconstructs final state particles (FSP), then proposes intermediate particle candidates, and finally predicts potential B meson candidates. In the end a probability is assigned to each B meson candidate based on the reconstruction efficiency of intermediate particles and the B-meson with the highest probability is picked. A detailed description of the FEI can be found in [11].

The performance of the FEI is outstanding compared to its very successful predecessors [12] that were used in Belle, however it is likely only the first out of a generation of future Machine Learning (ML) algorithms, that will be developed for full decay tree reconstruction in B-factories. The decay channels reconstructed by the FEI are hard-coded explicitly: 100 decay processes have been specified. This leads to the efficient reconstruction of $\mathcal{O}(10000)$ B-meson decay chains, but restricts the total branching fraction coverage of the algorithm to around 15% and relies on the developers' intuition to decide which decay processes to reconstruct. Moreover, the FEI's approach includes 6 distinct stages, that each runs only after completion of the last. Therefore the final results rely heavily on the performance of each stage.

2.4 Deep Learning motivation and elements of Neural Networks

Deep Neural Networks (NNs) are techniques used in various learning tasks that mimic the learning mechanisms in biological organisms. The human brain consist of billions of neurons connected to each other in complicated ways. The strength of these connections is adaptive and changes in response to external stimuli, defining all human physical and mental activity. The same basic concept is used in NNs. External stimuli correspond to data, that is propagated from input neurons to output neurons. The propagation happens by multiplying the values stored in the initial neuron, with the corresponding value of the weight that connects it with the final neuron. Learning is based on correctly adjusting these weights in order to make correct predictions. Data is stored in the form of tensors, so the fundamental mathematical operation that an NN is built on is tensor multiplications.

It is highly anticipated that Deep Learning (DL) will revolutionise Particle Physics [13], in the same way it has revolutionised numerous aspects of industry and everyday life (image-speech-text recognition [14] [15] [16], protein attributes prediction [17], self-driving cars [18], etc.). Traditional computing has proven to be very efficient in performing arithmetic calculations very fast and obeying a strict, user-defined, list of orders. Decision and prediction making, without the need for explicitly programming instructions, but learning from experience, was only possible through ML. However, the user still needs to be very active in the learning procedure, in the sense of breaking down bigger problems into smaller ones and combining the results, or creating the appropriate features that rely heavily on intuition and expertise [19]. The sub-field of DL has arisen due to large amounts of available training data and enormous computational power in the form of Graphics Processor Units (GPU). One of the main advantages of DL is the end-to-end learning approach, which means that the learning does not occur in distinct stages. A DL model can predict outputs, given a list of inputs in one step, which significantly speeds up the application time of the algorithm and provides a holistic manner of training [20]. Furthermore an end-to-end approach gives rise to the so-called *representation learning* which is the task of learning representations of low level data in order to extract useful information, without relying on human ingenuity [21]. These are two very important elements, especially in High Energy Physics (HEP), where the time of acquisition and analysis of the data is vital and the search for NP makes the choice of the appropriate variables very challenging.

The basic block of modern DL architectures is the Multilayer Perceptron (MLP). An MLP consists of an input layer, an output layer and multiple hidden layers, the number of which is defined by the user. MLPs are often called Feed Forward Networks (FFN) as each layer feeds the next one in a single direction. An activation function is deployed after each layer in order to decide which neurons are triggered or not. This is the same logic as an ON and OFF function according to the values stored in each neuron. The values stored in the neurons lie in range $(-\infty, +\infty)$, so without any sort of activation these values are meaningless. Mathematically the forward pass from one layer to another is expressed as: $z = f((\overline{W} \cdot \overline{X} + b))$, $\overline{W}$ are the weights connecting the two layers, z are the values of the following layer, $\overline{X}$ are the values of the previous layer b is an optional bias and f is a particular activation function. In the end one or several predictions are made and they are compared to some target values, defined before the training. A loss scalar is calculated as $L = \Phi(\hat{y} - y)$, where Φ is a specific loss function, $\hat{y}$ are the values of the output layer and y the target values. All this happens during the forward pass of the data from the NN and what follows is the actual learning. NNs use the back-propagation algorithm [22] that inversely calculates the gradient of the loss function with respect to the weights. These gradients are utilised to update the weights in order to minimise the loss. Starting from the output layer and moving backwards, the gradient $\frac{d(loss)}{d(w_i)}$ is subtracted by the current weights, in order to minimise the loss. It is apparent that the backpropagation takes place in a multidimensional space, thus training a neural network is a computationally expensive procedure. This forward-backward procedure happens recursively and stops after a

certain amount of iterations, that are called *epochs*, or when a satisfying performance is achieved [19]. All the above are shown schematically in Figure 2.4

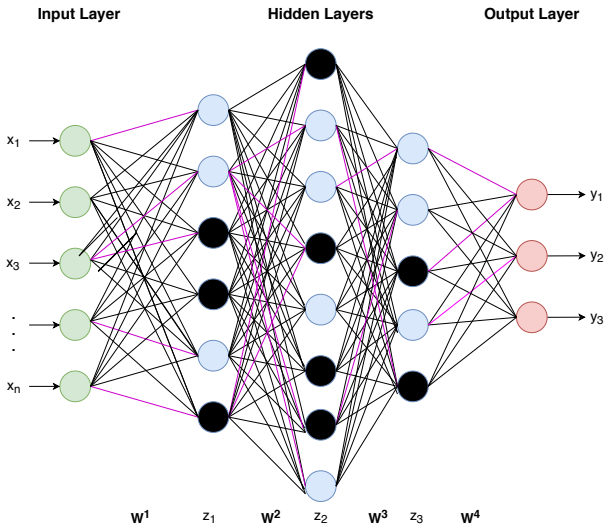


Figure 2.4: A typical MLP. The green nodes are the inputs and the red ones are the outputs of the MLP. Every edge between two nodes represents a weight that is updated during the training. The nodes with black are the nodes that have been turned off by the activation function, while the purple edges are the ones that have been randomly dropped out due to regularization.

An NN model consist of two types of parameters the *learnable parameters* and the *hyperparameters*. The learnable parameters of the model are typically the aformentioned weights. The hyperparameters of the model are the ones that define the architecture itself. Some of the key hyperparameters are the number of layers, the number of nodes per layer, the learning rate, the number of epochs (iterations), and the the dropout rate. The learning rate is the size of the steps made during the search for the minimum of the loss scalar in the multidimensional space. The learning rate is multiplied by the gradient $\frac{d(loss)}{d(w_i)}$. Higher learning rates change the weights drastically, while lower learning rates slow down the search of a minimum. The number of epochs is a hyperparameter of the model, however the user typically deploys an *early stopper* to terminate the training when the loss starts to increases. Dropout is a regularization technique such as the early stopping and it is the percentage of random neurons within the model to be deactivated. All regularization techniques are used to avoid the overtraining of the model. The overtraining occurs when the model learns the statistical noise of the training sample and misses the important information, making it useless for unknown samples.

Normally the data is splitted into a training and a validation set to monitor the level of overtraining. The *training set* is the one that is used when updating the weights. At the last stage of an epoch the *validation set* is passed through the updated model and the output is compared to the target to compute the validation loss and other metrics. Since the training and the validation sets are statistically independant, the similarity in their performance exhibits the ability of the trained model to generalise. Any unknown sets that the model is applied on after the training is referred to as *testing set*.

Usually the user performs a hyperparameter optimization in order to achieve the best possible performance. Techniques for hyperparameter optimization differ from simple trial-and-error approaches to more sophisticated Bayesian optimizations or exhaustive grid searches. After the determination of the general structure of the model, the appropriate setting of the hyperparameters is the most important contribution by the user in the learning procedure.

2.5 Graph Neural Networks

A graph is a type of data structure that represents objects (nodes) and the relations between them (edges). These relations can be very complicated and for that reason, graphs are usually referred to as *non-Euclidean data structures*. Graph Neural Networks (GNN) are NNs that are designed for graph structured data and aim to capture both information related to the nodes themselves and relational information described by structure. [23] GNNs have some significant advantages over other techniques which arise from the nature of graphs. Conventional DL techniques such as Convolutional Neural Networks or Recurrent Neural Network that are very successful in other domains fail to handle properly graph structured data. One reason for that is that the nodes within a graph are permutable: there is no natural order to represent the neighbours of nodes in a graph. Furthermore in other Euclidean data structures such as images, the connections are embedded in the objects themselves (i.e. each pixel has a predefined number to adjacent pixels, whereas in graph structured data the number of edges of each node varies) and the edges may have their own features while representing dependencies between nodes. [24] These particularities of graph structured data have made GNNs very popular among different fields, where they are used for node or graph classification tasks, link prediction or edge labelling tasks [25][26][27]. In Particle Physics GNNs have been used in the recent years in Calorimetry for tower or jet reconstruction [28][29], as they are suitable for capturing relations of 3D structured data.

3. Outline of this work

3.1 Definition of the problem and proposed approach

The overarching goal for this research internship is to demonstrate a means of learning to reconstruct particle decays from examples. The approach proposed is one of graph generation, since a decay tree is naturally represented as a graph. In Graph Theory there are two matrices that fully define a graph. The *adjacency matrix* denotes the structure of a given graph and the *feature matrix* contains the attributes of each node. An entry of the adjacency matrix is 1 if there is a connection between two nodes and 0 if they are not connected. One may also define an *edge feature matrix* to represent features attributed to the edges if they exist. In the case of decay tree reconstruction in Particle Physics, where only the final state particles (FSPs) are detected, the adjacency matrix is unknown. It is only after the successful prediction of the adjacency matrix that one can attribute kinematic features to the intermediate particles. To achieve this, this work uses the *Lowest Common Ancestor Matrix* (LCA). The LCA matrix shows the level of which the nodes share a common ancestor in terms of generations. An entry of 1 means that two particles share the same mother, 2 means that two particles share the same grandmother and so on. The LCA matrix contains all the necessary information to reconstruct the whole adjacency matrix. One can derive the adjacency from the LCA and vice-versa [30]. Figure 3.1 shows an example decay tree and its corresponding matrices.

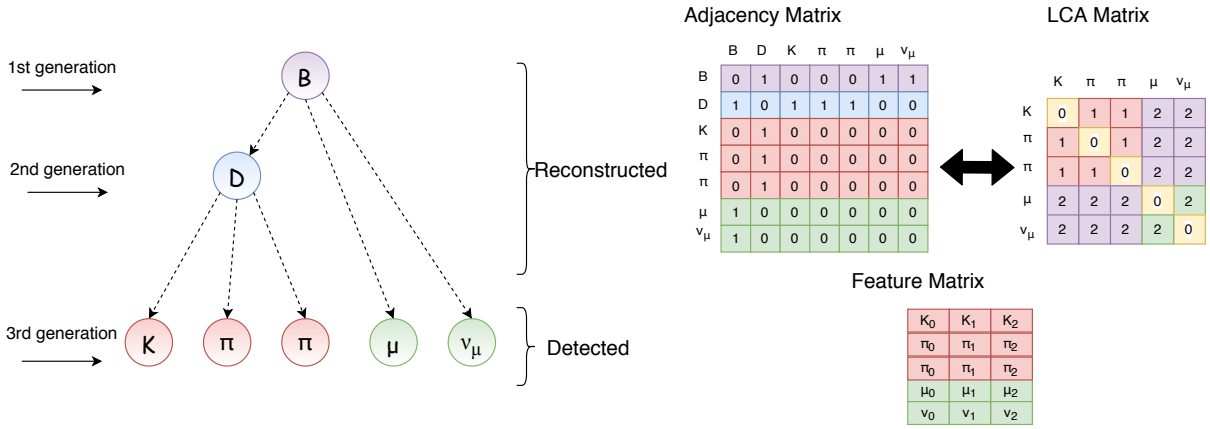


Figure 3.1: *Left:* Example of decay tree showing what is input available to the network (detected) and the decay tree that should be output (reconstructed). *Right top:* The adjacency matrix shows the structure of the graph. *Right bottom:* The feature matrix contains information about the attributes of these nodes. For particles these may be the 4-momentum or any other physical quantity.

3.2 Demonstration of graph-based particle reconstruction

The initial target of this work was to demonstrate a graph-based particle decay reconstruction. As of today there are very few attempts to use GNNs to reconstruct particle decays, which are mainly individual projects still on the level of proof of concept. In the DL literature there is a huge number of GNN applications, although they are always data-agnostic, thus there is no out-of-the-box solution for the very specific problem at hand. Roughly all the first month of this internship was spent on extensive research in the DL literature, in order to find the most appropriate approach to implement.

Several different techniques have been used, without however leading to gratifying results. The common ground among all the different approaches is that a complete graph is used where the nodes are the FSPs as illustrated in Figure 3.2 The advantage of such an initial configuration is that no assumptions or intuition are needed, since all the particles are treated equally. Particles are bound to kinematic laws and all the necessary information to reconstruct the decays is contained in the respective quantities. Firstly it was attempted to use Graph Convolutional Networks to predict the LCA matrix of a given tree [31]. The fully connected configuration and the small size of the graphs made it impossible for the models to learn any useful information. Secondly pooling techniques were studied to select the nodes that share common ancestors [32]. However these techniques always work in a pairwise manner, by finding a first pair that is likely to be linked and then build on top of that. Consequently they depend heavily on this first guess, making them unreliable -or at least inefficient. Later on an edge contraction approach was deployed where the network recursively erases connections leading to classified nodes [33] This technique suffers from the same problem as the graph pooling. Finally the most promising approach was the one of edge label prediction and is described in detail in the following.

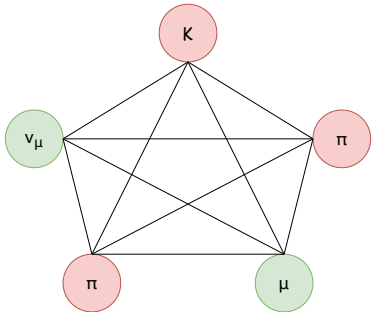


Figure 3.2: In graph theory a complete graph is a fully connected graph, where all the nodes are interconnected.

3.3 Neural Relational Inference for Interacting Systems

The goal of this approach is to assign and learn how to predict the edge labels of the fully connected graph in Figure 3.2. These labels are interpreted as the entries of the LCA matrix. Edge labelling is still one of the emerging subfields of GNNs, so there are not abundant applications in the literature [34]. Probably the most appropriate architecture for the problem of particle decay reconstruction, which also has the advantages of being conceptually simple and relatively not expensive computationally, is the one provided by the *Neural Relational Inference for Interacting Systems*(NRI) [35].

The model presented in NRI is one of a graph autoencoder, used to predict the law of Physics that governs the interaction of classical particles. An autoencoder such as the one presented in Figure 3.3 first encodes input information in some latent space, of typically lower dimension, which is decoded in the second stage in order to reproduce the initial input. The learning happens by penalizing the wrong recreation of the input and so the model manages to find the optimal latent space that contains the minimum essential information to describe the data. For this work only the first encoding part of the autoencoder is needed. The feature matrix of the complete graph is given as an input and the output of the network predicts the labels for each edge which are then compared to entries of the LCA matrix.

The main idea behind the encoder is the interchange between node and edge features and the creation of deeper representations. The basic layer is a 2-layer MLP with ELU activation functions [36] and dropout [37]. The number of hidden nodes and outputs along with the number

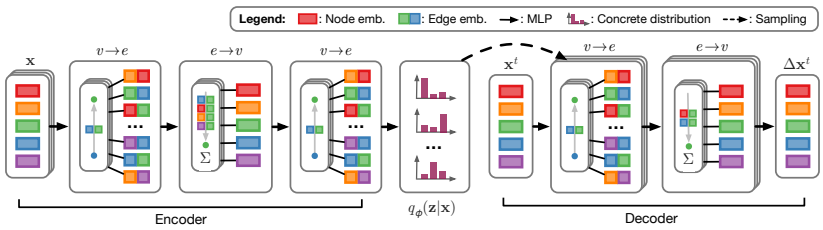


Figure 3.3: The Graph Autoencoder of the NRI.

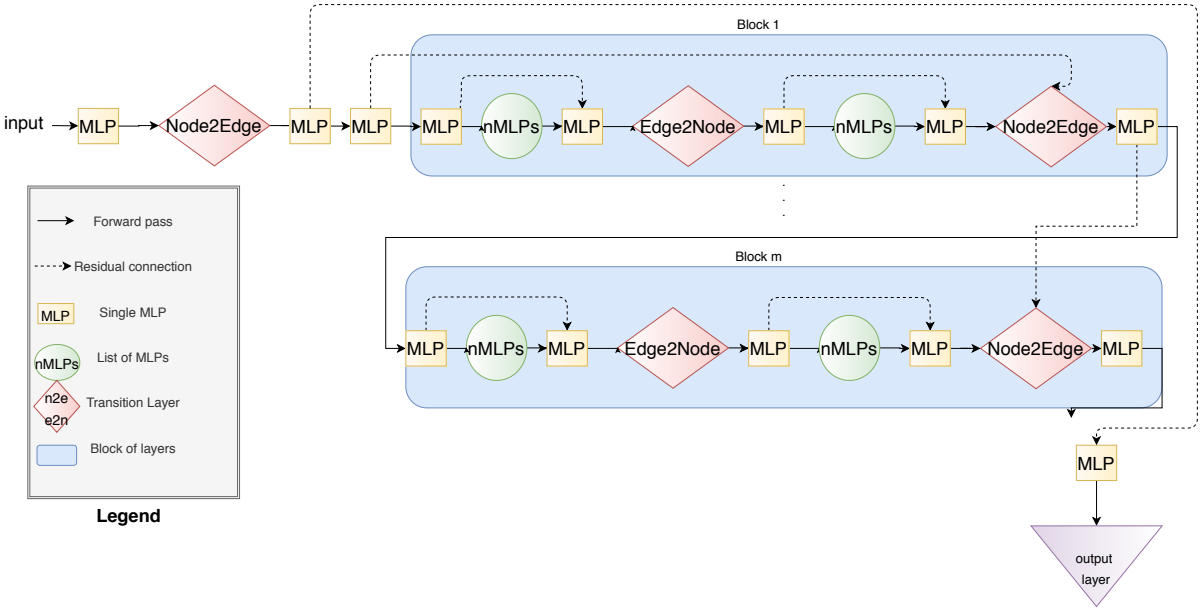


Figure 3.4: The encoder architecture used in this work.

of MLPs are set after the hyperparameter optimization and they are discussed in the following sections. The key components of this model are the Node2Edge and Edge2Node layers. These layers make the transition from node representations to edge representations and vice versa. A detailed description of these layers can be found in the Appendix. All the models in this work have been built using the DL library *Pytorch* [38]. Pytorch is one of the state-of-the-art deep learning research platforms that provides maximum flexibility and speed, and allows the user to take advantage of the computational power of GPUs. The original encoder consists only of 8 layers like the ones described above. It was found that for particle decay reconstruction a way deeper network is needed that scales up to even 200 layers in some cases! The architecture of the model for this work is presented in Figure 3.4 For all the trainings that are presented in the following ELU activation functions were used [36]. The loss utilized was the cross entropy loss [39] along with the Adam optimizer [40].

3.4 Data production

The appropriateness of the encoder was first demonstrated on a generic phasespace decay sample. Beginning with simple, generic phasespace decays, generated with the python *Phasespace* library [41], allows for precise control over the complexity of training data. Once the encoder has been shown to be able to learn this, it can then be progressed to the more complex Belle II simulated decays. Phasespace provides the tools to generate Monte Carlo n-body phasespace decays. It is possible to define the decay tree structure, the masses of generated particles, and

potentially a boost for the initially decaying particle.

The complexity of the experiments was increased step by step. The very first dataset that was used, was the simplest non-trivial case that is showed in Figure 3.5. The daughter particles in a three body decay have a rich range of momenta and energies. For the experiments that are described in the following sections, the initially decaying particle is at rest before decaying and the masses of all the particles are showed in Figure 3.5. With this configuration the energy spectra and the momenta of the final state particles are strongly overlapping as can be seen in Figure 3.5. Evidently this decay chain is a very good first test since there is no trivial way to find the common ancestors of FSPs by just summing up the energies and the momentum components. This graph is referred to as *3o3 dataset* in this document, since two groups of three particles originate from the same mothers. This naming convention is followed in the rest of the document for all the Phasespace datasets. That is if the X particle decays to two FSPs and the Y decays to four, this dataset is called *2o4 dataset*. After the first demonstration numerous datasets were generated to test the robustness of the model with respect to different kinematic phasespaces, increased complexity in the combinatorics and missing information for the FSPs. For all the trainings on phasespace datasets the 4-momenta of FSPs have been used as input features to the network.

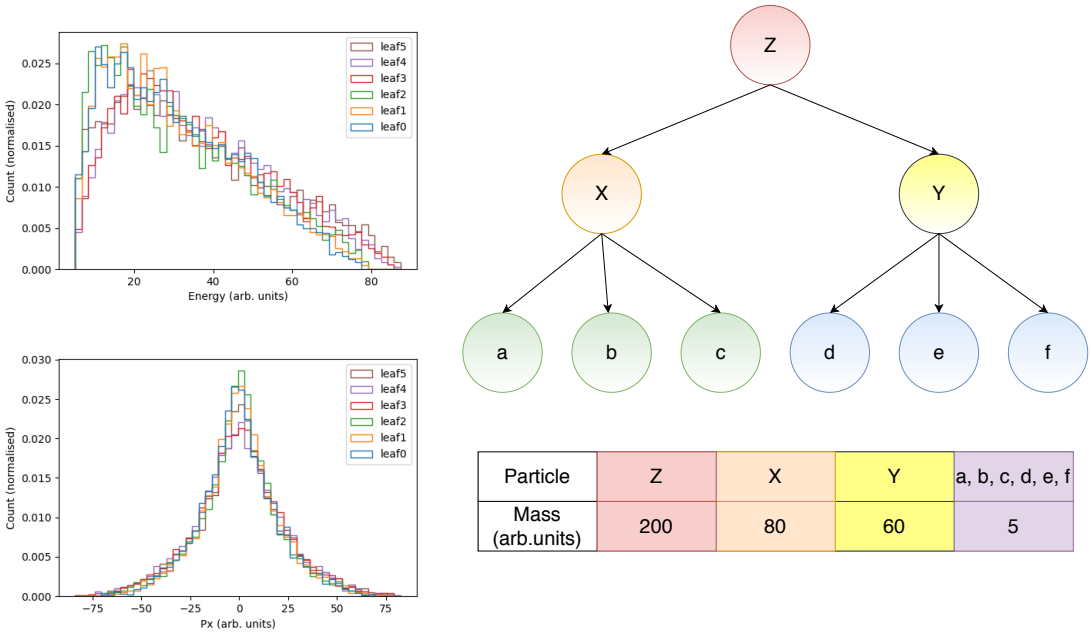


Figure 3.5: *Left top*: Overlapping Energy spectra of the FSPs in arbitrary units. *Left bottom*: x component of the momentum of the FSPs in arbitrary units. Similar situation for the y and z momentum components as well. *Right top*: Graph of the decay chain. *Right bottom*: Table showing the masses of all the particles in the tree.

The following logical step was to test the performance of the model on Monte Carlo (MC) samples, that represent the real phasespace of the decays that take place in Belle II. Table 3.1 shows the six different tag sides from the MC samples that were generated, along with a very short description of the motivation behind the choice of each channel. The signal side was chosen to be $B \to \mu\nu_\mu$, since its a distinct signal, with only one detected particle and therefore can be easily isolated from the tag side during the analysis. For all the trainings on MC Truth datasets the 4-momenta and the charge of FSPs have been used as input features to the network.

Lastly the model was tested on events that were reconstructed after the simulation of the Belle II detector. This series of experiments was of course the most realistic ones, as any proposed algorithm that would be used by the Belle II collaboration would have be applied on real data

from the detector. The same decay channels that were presented in Table 3.1 were utilised for the reconstructed events as well.

Decay Channels generated with the Belle II software		
Decay Channel	N^{o}FSPs	Motivation
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\pi^+$	5	benchmark tag side on T.Keck's PhD thesis on FEI
$B^+ \to D^-(\to \pi^-\pi^+\pi^+)\pi^+\pi^+$	5	two 3-body decays, overlapping spectra, same FSPs)
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\rho(\to \pi^-\pi^0)$	7	resonances not dealt by FEI, includes 4 photons that need to be assigned to the correct π^0
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\omega(\to \pi^+\pi^-\pi^0)\pi^+$	9	Three 3-body decays, resonances not dealt by FEI
$B^+ \to D^-(\to \pi^-\pi^-\pi^+\pi^0)\pi^+\pi^+\pi^0$	9	two 4-body decays
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)e^+\nu_e$	5	semileptonic decay to demonstrate semileptonic tagging

Table 3.1: Decay channels produced with the Belle II software for this work. All the π^0 decay to two photons. All the datasets contain the decay channel presented here and it charge conjugate

3.5 Metrics and Hyperparameter Optimization

One of the most important aspects of all the experiments was the set of metrics that were used to evaluate the performance of the model. The two very basic metrics that one monitors during a typical DL application is the overall accuracy of the predictions per epoch and the average loss per epoch. In the particular case of the LCA matrix prediction an ordinal accuracy was deployed. A simple example makes very clear the need for such a metric. Assuming that the target value of an LCA entry is 1, it is worse to mistakenly predict the value 3 for this entry than the value 2. Thus the ordinal accuracy increases the penalty for an incorrect prediction according to how far away from the actual value it is. Another important metric is the percentage of perfectly predicted LCA matrices, since this corresponds to perfectly predicted decay trees. In order to evaluate the quality of the predictions, the metrics M1P, M2P, M3P, M4P, M5P and BadLCA were used. M1P means 1 pair of wrong predictions, M2P 2 pair of wrong predictions and so on. BadLCA is the percentage of predicted LCAs with more than 5 wrong pairs. These metrics correspond to pairs of mistakes, as the symmetry of the predicted matrix is imposed at the latest stage of the architecture.

In Section 2.4 it was discussed that the hyperparameters of the model are critical for the performance of any experiment. Especially the ones that are related to the architecture of the model, such as the number of layers, or the number of hidden nodes that defines the size of the layers, are of significant importance and their tuning is computationally very expensive. For this work the Hyperparameter Optimization (HPO) software *Optuna* was utilized in order to find the optimal configuration for the model. [42] Optuna tries to maximise/minimise an objective function in a particular hyperparameter space that has been defined by the user. Even though searches conducted with Optuna are not exhaustive they have the advantage of offering a dynamic construction of the search space, an efficient pruning algorithm that terminates unpromising trials and in general flexibility to run optimizations. All the HPOs and the trainings were carried out using four GeForce GTX TITAN X 12 GB GPUs. Only the depth studies discussed in section 4.3 were made with eight Tesla V100 32 GB GPUs

4. Results and Discussion

4.1 Phasespace

4.1.1 Proof of Concept

The first dataset used, introduced in section 3.4 and shown in fig. 3.5, serves as a proof of concept. The model used was shallow [1] and not rigorously optimised. The hyperparameters were chosen instead from reasonable initial values. Figure 4.1 shows the performance of the model on the validation set during the first successful training. The percentage of the perfectly predicted LCAs is relatively low as expected for a first attempt. Moreover the training suffers from overtraining as can be seen in Figure 4.2: after several epochs the model begins memorising the training set, leading to worse predictions on the validation set.

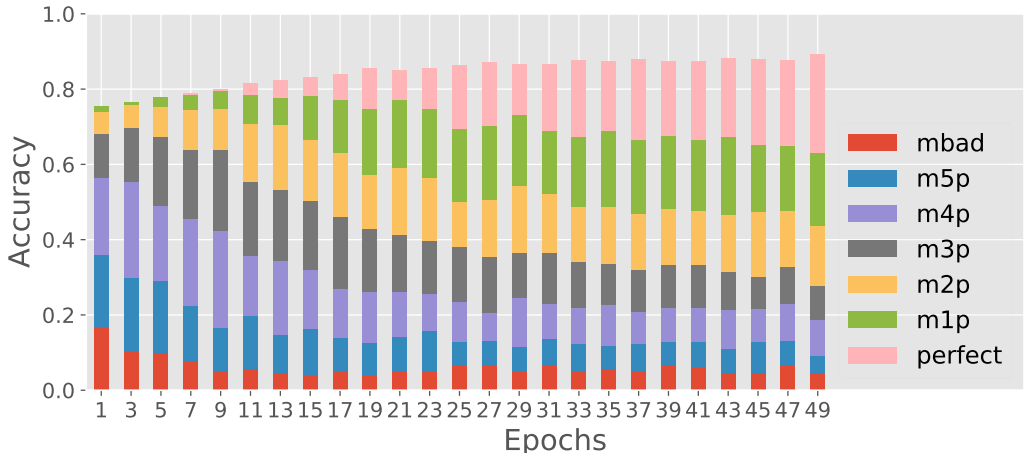


Figure 4.1: Proof of concept training results on the 3o3 dataset

4.1.2 First Level Reconstruction

The next step was to investigate the capacity of the network in performing first level reconstruction. First level reconstruction includes graphs no deeper than 3 generations, like that presented in fig. 3.5. Table 4.1 shows the weighted accuracy, the loss and the percentage of perfect LCAs for the training and the validation set of the numerous datasets that were generated.

The networks trained on these datasets are still shallow. The only hyperparameters that are tuned are the number of hidden nodes, the batch size, the learning rate, and the dropout rate. The performance of the trainings drops as a function of the number of possible combinations of the FSPs and the complexity of the intermediate particles' decays. For example even though the number of FSPs is the same for the datasets 2o5 and 3o4 there is a clear drop in the performance. This is due to the 2o5 dataset containing a trivial 2 body decay which the model learns to recognise quite easily, and the number of FSPs permutations scales according to the equation

$$C(n,r) = {}^{n}C_{r} = \frac{n!}{(n-r)!r!},$$

where n is the number of FSPs and r and $n-r$ are the number of particles that share a common mother. Another evident problem is the overtraining that occurs in all datasets. There are

[1] Shallow in this case means that there's only 1 block of layers and 1 MLP in the list of MLPs as shown in Figure 3.4

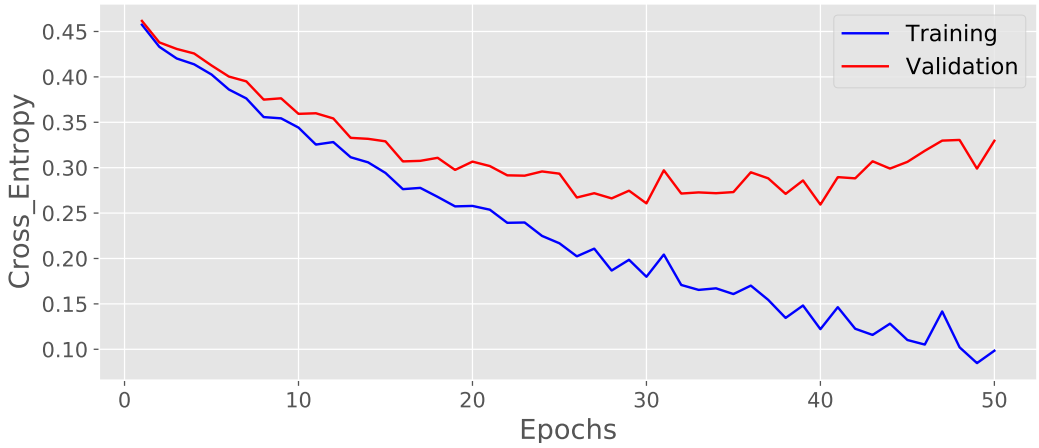


Figure 4.2: Demonstration of overtraining during the proof of concept training. The loss for the training set decreases, but the loss for the validation set eventually increases as the model learns the statistical fluctuation of the training set. The optimal training is one where the loss and all the metrics of the training and validation sets are evolving in the same manner.

First Level Datasets and Respective Results						
Set	OAcc(T)	OAcc(V)	CEL/(T)	CEL/(V)	PrfLCA(T)	PrfLCA(V)
2o2	0.981	0.973	0.021	0.060	0.873	0.840
2o3	0.963	0.938	0.142	0.146	0.712	0.602
2o4	0.991	0.971	0.084	0.079	0.929	0.852
2o5	0.992	0.978	0.036	0.077	0.939	0.857
3o3	0.960	0.894	0.053	0.277	0.598	0.291
3o4	0.936	0.833	0.085	0.483	0.370	0.087
3o5	0.924	0.841	0.151	0.382	0.278	0.090
4o4	0.928	0.805	0.081	0.514	0.272	0.060
4o5	0.910	0.820	0.135	0.421	0.164	0.043
5o5	0.849	0.831	0.280	0.342	0.144	0.112

Table 4.1: OAcc: Ordinal Accuracy, CEL: Cross Entropy Loss, PrfLCA : Perfectly Predicted LCAs. The performance of the model drops as a function of the FSPs combinations. (T) stands for Training set, while (V) for Validation set.

several possible explanations for why the overtraining occurs. The simplest solution to prevent the observed overtraining is to use more typical DL regularization. Another explanation is that the model is not deep enough to learn useful representations of the data, but only manages to learn statistical noise at this point. Evidently, another way to interpret this performance is the need for more training data as the amount of data strongly affects the capacity of any DL model. The findings of this work, presented in the following sections, indicate that the depth of the model played a major role in the overfitting during a training.

4.1.3 Mix of datasets I

Any algorithm developed for Belle II should be applied to generic $Y(4S)$ decays in the long term. This means that the model should be able to recognise and predict different decay trees and decay chains at the same time. The simplest way to demonstrate this is to mix different datasets that contain the same number of final state particles. This limitation arises from the requirement

that all the LCA matrices of the initial DL model must be of the same size. Therefore predicting LCA matrices of different dimensions at the same time needs extra manipulation [2]. Figure 4.3 shows the performance on the validation set for a dataset containing 6 FSPs that emerged from the mixing of the respective datasets used in Table 4.1. The networks built for these trainings are tuned using Optuna and optimise the number of blocks of layers and size of MLPs shown in Figure 3.4. No significant overtraining is reported for these experiments. The key difference between these trainings and the first level reconstruction presented in Section 4.1.2 is the depth of the model. This is a strong indication that the first models had limited prediction capacity due to their smaller size. Similar plots are presented in the Appendix for 7 and 8 FSPs. The training for 8 FSPs is worse than the two other cases, though it appears likely that with a more exhaustive HPO a similar performance could be achieved.

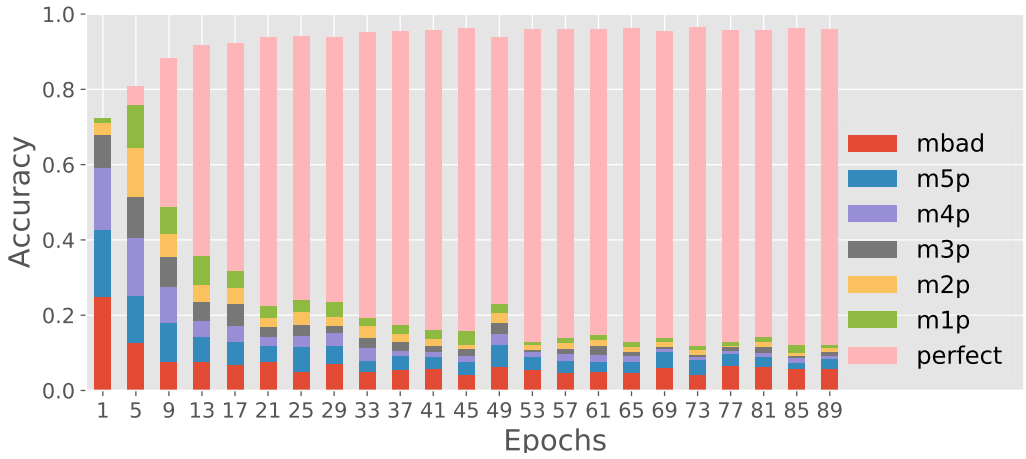


Figure 4.3: Performance on the validation set for dataset contatining 6 FSPs

4.1.4 Noisy data

Verifying the model's robustness on data with noise is critical. For real experiment data the detectors always introduce uncertainties (e.g. resolution effects) [43], therefore conserved quantities such as energy or momentum do not add up exactly. To simulate this random noise was introduced to the data. In the preprocessing stage for every momentum component of every particle in all events a Gaussian distribution is generated with the mean the actual momentum value and with variance of 1%. Note that the smearing factor for the Belle II tracking detectors is only 0.1% [9], so the scenario created in this experiment is more unfavourable [3]. A value is picked up randomly from this normal distribution and then the energy is recalculated as $E = p_x^2 + p_y^2 + p_z^2 + m^2$ since the masses of all the particles are known. Figure 4.4 shows the performance of the network on this noisy data situation for the 6 and 8 particles datasets. No significant discrepancies between the smeared and the unsmeared datasets is seen, which is a straightforward demonstration that the network is robust to random noise. The last step in this series of noisy data experiments was to save a model that was trained on unsmeared data and test it on a sample of smeared data. For 6 FSPs the testing performance of the model is: Acurracy:0.9756, PerfectLCA: 0.8891, performance similar to the trainings presented above. This attempt is mainly driven by random discrepancies that may occur between the MC samples that any DL model is trained on and the real data that it would be applied to.

[2] These techniques are discussed in more detail in section 4.1.6

[3] However this is an over-simplification since the uncertainties in Belle II events are correlated

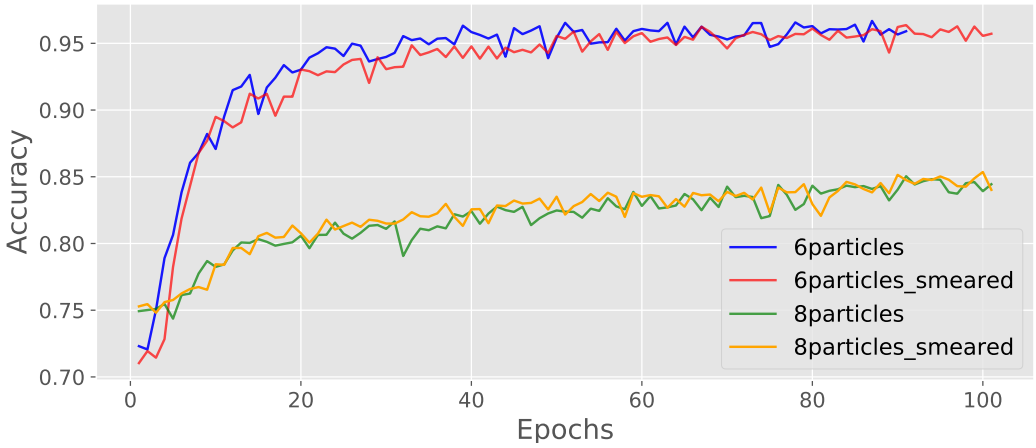


Figure 4.4: Comparison of the model's accuracy on smeared and unsmeared data for the 6 and 8 particles datasets.

4.1.5 Missing Particles

The next step was to demonstrate that the model is able to handle events with missing particles. These events correspond to semileptonic decays, i.e. events where one or several particles have not been detected. There is of course a limit to the least necessary kinematic information, from both signal and tag sides, to reconstruct an event, however a detailed discussion on this topic is out of the scope of this thesis. The following results serve as a demonstration that the model presented in this thesis can handle events with missing kinematic information. Figure 4.5 shows the performance of the model on the validation set for the 3o3 dataset when one daughter particle has been randomly removed from one intermediate particle. In the Appendix results are shown for the cases where two daughter particles have been removed from one intermediate particle and when one daughter particle has been removed from each intermediate particles. It's observed that for the first two cases the performance of the model is similar to the one for the mix of dataset in section 4.1.3. This indicates that the network is robust to missing particles and warrants further investigation.

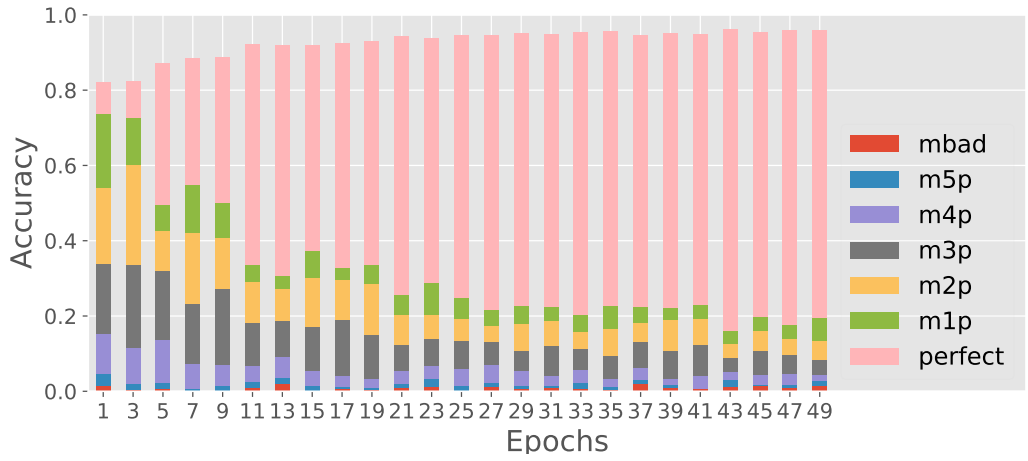


Figure 4.5: Performance of the model on the 3o3 dataset when 1 FSP has been removed.

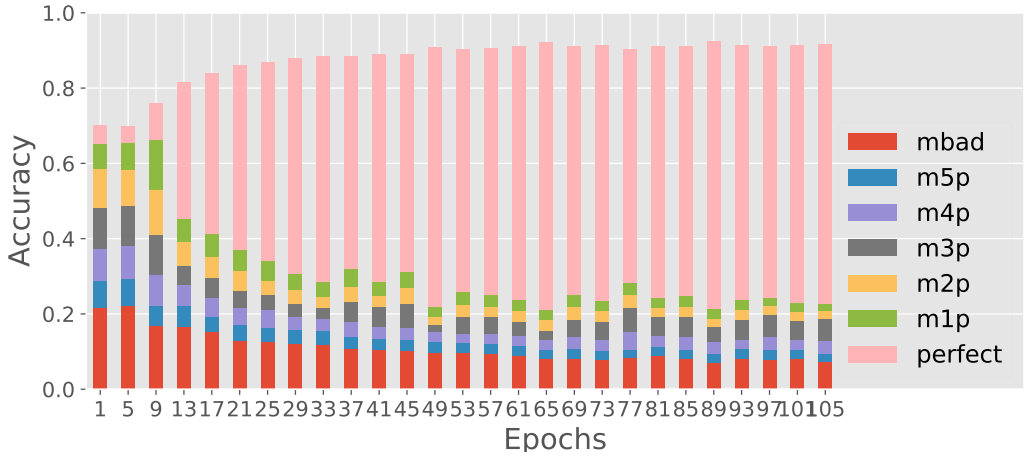


Figure 4.6: Performance on the Validation set for all the datasets.

4.1.6 Mix of datasets II

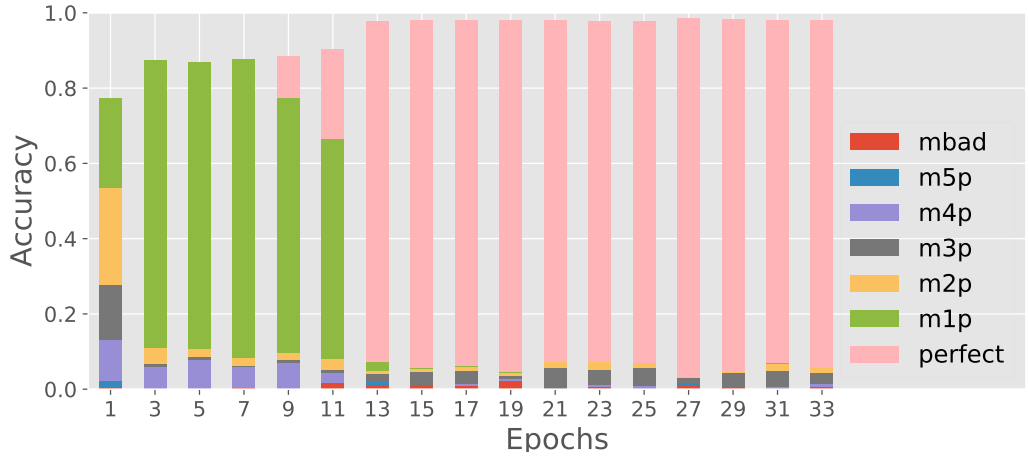
The final experiment that was conducted using the Phasespace datasets was to train on a mix of all the datasets that were presented in Table 4.1 together. In order to do that, the problem of training on datasets with different number of FSPs needed to be overcome. Padding is a technique traditionally used in convolutional neural networks or other ML techniques to artificially increase the size of different sized inputs and fit them in a multidimensional array. Zero padding works by adding rows and columns of zeros when a particular dimension of a tensor needs to be extended. For instance the LCA matrix for the 5o5 dataset is a 10×10 matrix whereas the one for the 3o5 is a 8×8 matrix. For the latter there are two extra rows and columns of zeros added to meet the dimensions of 5o5. These extra zero elements are ignored in the learning procedure using a technique called masking, thus they don't affect the loss calculation and the weight update, nor do they interfere with the validation metrics such as the Perfect LCA. The same principle is followed for all the datasets and for the initial adjacency and feature matrices as well.

Figure 4.6 shows the results for a mix of all the datasets presented in table 4.1. The percentage of the Perfect LCAs is lower than the previous tests, however almost 75% of the predictions are perfect. The various successful tests on the phasespace datasets, including this last one which demonstrates that the model can handle different kind of input datasets with different FSPs, warrant the proceeding to Belle II datasets, with an ultimate goal of training on generic B-meson decays.

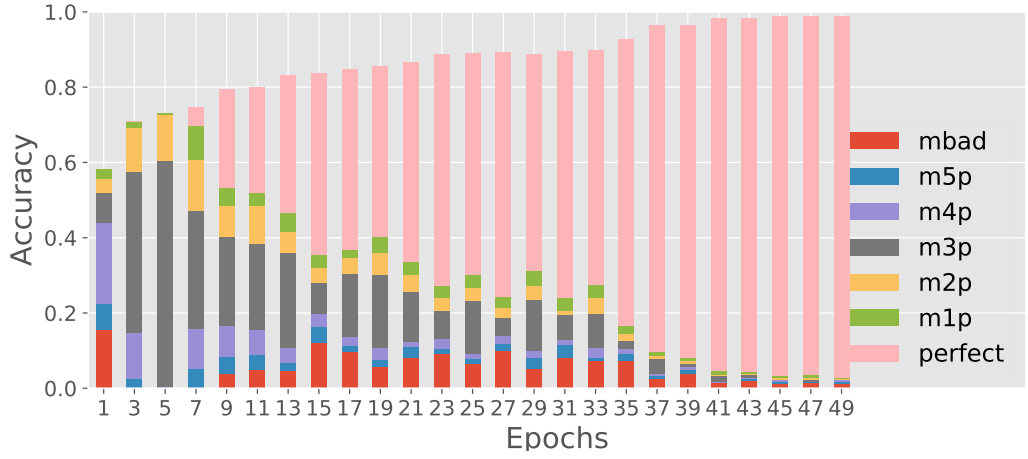
4.2 MC Truth

4.2.1 Demonstration on first MC truth channels

Table 3.1 in section 3.4 shows the decay channels for the tag sides that were produced using the Belle II software. Figures 4.7a and 4.7b show the model performance on the validation set for the benchmark Belle II tag side and the semileptonic B-meson decay channel. The percentage of perfect LCA predictions is almost 95% which is higher than most of the Phasespace datasets. Moreover the network achieves a very good performance, especially for the $B^+ \rightarrow \overline{D^0}(\rightarrow K^+\pi^-\pi^0)\pi^+$, very early during the training. This is well expected as the kinematic scenario created in the Phasespace datasets is more troublesome than the one in the first Belle II datasets. The FSPs in the Belle II datasets differ in their masses in multiple orders of



(a)



(b)

Figure 4.7: Performance on the MC truth validation datasets for the (a) $B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\pi^+$ and (b) $B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)e^+\nu_e$ decay channels.

magnitude for the cases of electrons and photons. This means that there is a lot separation information embedded in the 3-momenta of the FSPs, event though their momentum spectra are overlapping, that leads to efficient LCA predictions. The training performance for the rest of the MC datasets in table 3.1 is given in the Appendix.

4.2.2 Mix of datasets III

Following the same methodology as for the Phasespace datasets a mix of all the different decay channels is used to train the model. Figure 4.8 shows the training performance of the model on a mix of all the B-meson decay channels from table 3.1. The performance is significantly lower than in the case of Phasespace dataset, however this is something well expected. Some of the kinematic scenarios created in the Belle II channels and included in this mix of datasets are more difficult to handle, as they contain 9 final state particles, deeper decay trees and numerous photons, originating from $\pi^0 s$, that probably make the predictions of the model more difficult.

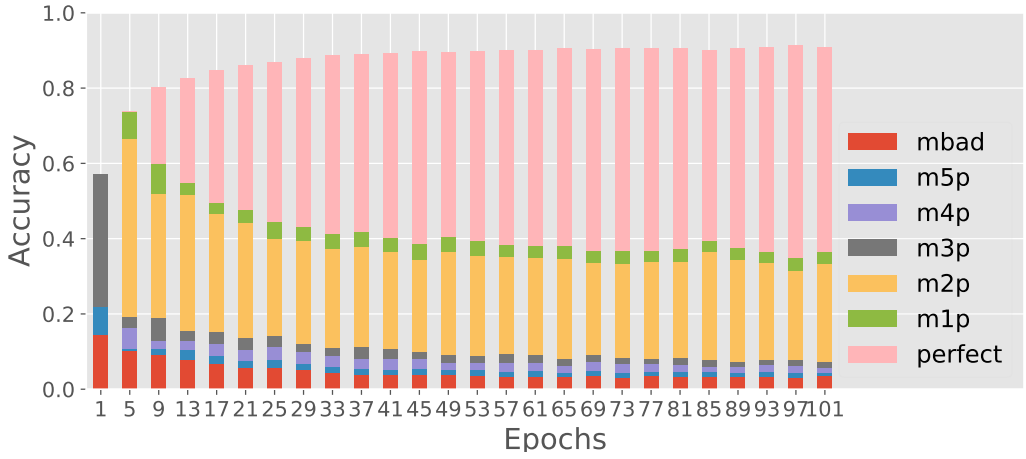


Figure 4.8: Performance on the Validation set for all the Belle II datasets.

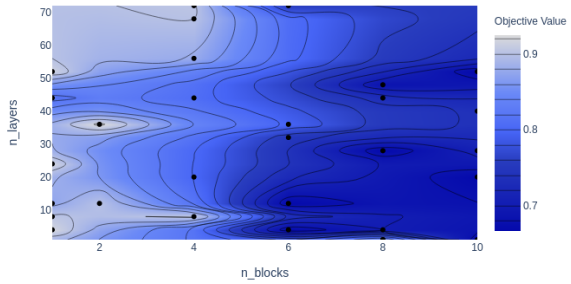
4.2.3 Reconstructed events

Results for the trainings on reconstructed events after the detector simulation using the Belle II software have not been included in this thesis. The firsts results are very promising and no significant discrepancies between these trainings and the ones on MC truth samples have been observed. In addition, the good performance on the noisy Phasespace datasets presented in section 4.1.4 is also an indicator that the model will be able to handle Belle II reconstructed events with detector related noise.

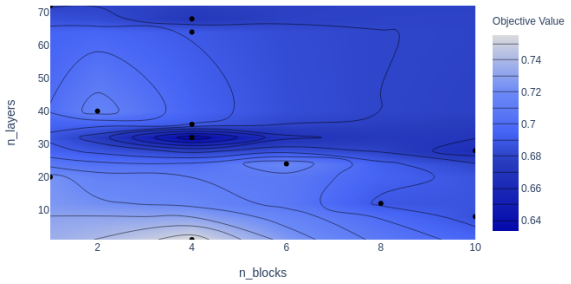
4.3 Depth Studies

The last study conducted for this thesis was aiming to identify potential patterns on how the size of the model scales with respect to the number of FSPs. For that reason optuna was deployed to map the hyperparameter space. The HPO performed was a grid seach to explore the whole hyperparamer space. Contrary to the rest HPOs in this thesis, these experiments used 8 Tesla V100-PCIE GPUs of 32 GB each for a grid search optimization. The hyperparameter space that was mapped included: number of layers : 1 and $2-72$ with steps of 4, number of blocks : 1 and $2-10$ with steps of 2, and dropout rate: $0.25, 0.4, 0.5$. The latter was included in the search since DL models with more learnable parameters have the tendency to overfit more [37]. In fig. 4.9 contour plots are presented, for the number of layers and the number of blocks, for the phasespace datasets. A trend towards lower number of blocks is reported from these experiments. This is something logical as multiple transitions from node to edge representations and vice-versa probably lead to loss of information or at least they don't favour better predictions [4] Besides that no final conclusions can be made regarding the number of learnable parameters of the model. However, since the combinatorics problem is an open problem in both DL [44] and HEP [45], these initial findings warrant further investigation in the future, as a potentially larger network could provide fast and efficient solutions to other HEP related problems too, where the number of possible combinations is large.

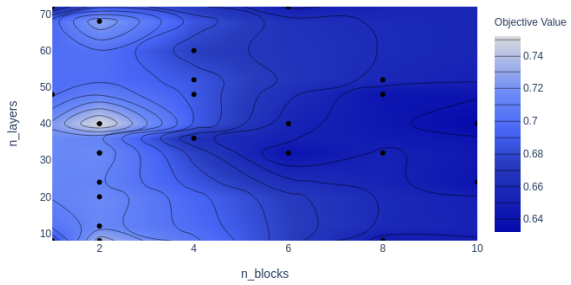
[4]The details about these transitions are presented in the Appendix.



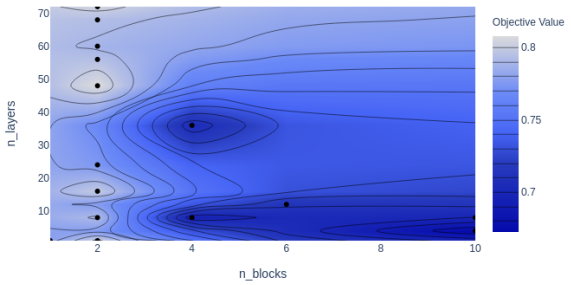
(a) 2o2



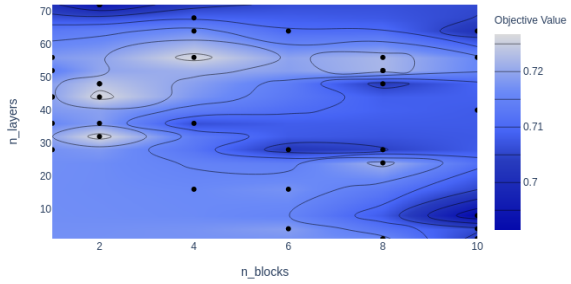
(b) 2o3



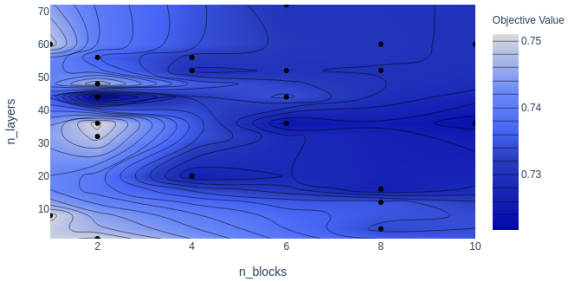
(c) 6 particles



(d) 7 particles



(e) 8 particles



(f) 4o5

Figure 4.9: Contour plots for the number of layers and blocks for the phasespace datasets. The objective value is the validation accuracy and is indicated by the shade of blue.

5. Conclusion and Outlook

This master's thesis has covered two major topics.

Proof of concept of a graph based approach for decay tree reconstruction. This work introduces one of the first attempts to reconstruct particle decays from example using graphs. The contribution of this work on the topic is twofold: an end-to-end Deep Learning solution is proposed for reconstruction of particle decays in Belle II , and it is demonstrated that the Lowest Common Ancestor matrix contains the necessary information to capture the structure of a decay tree, by using as input only the 4 momentum of each particle. Two key advantages of the proposed algorithm are that it doesn't rely on the intuition of the developers for the choice of which physical processes to exploit for efficient B-tagging and that no Physics expertise are needed for the determination of the appropriate input features.

Demonstration on numerous kinematic scenarios. The proposed model has been tested on generic phasespace decays, including complex n-body decays, decays with missing particles, and decays with random noise. The majority of the results are very promising. Moreover it has been shown that the model achieves similar, or even better performance in some cases, on datasets produced with the Belle II software. The most important results of this work are: the 75% of perfect tree predictions for the mixture of all the phasespace datasets, which indicates that the model can be used efficiently on unbalanced data such as generic B-meson decays, and the 95% of perfect predictions on the benchmark decay tree used by Belle II for B-tagging, which suggests that the model has the potential to compete with and even supersede the current algorithm.

This work is the first study on this particular graph approach as part of an effort to replace the current decay tree reconstruction algorithm FEI used by the Belle II collaboration. Therefore there are numerous further steps to be made in the development. Firstly, the model needs to be trained and evaluated on generic B-meson MC decays, after the fist demonstration on a handful of them in this work. Secondly, the model needs to be tested extensively on events where the final state particles have been reconstructed after the simulation of the Belle II detector. The initial findings of this work indicate that the model will be able to achieve similar performance such as those on the MC truth samples. Thirdly, it is required to investigate how efficiently the Belle II software can identify the intermediate particles that are implicit in the decay tree structure predicted by the LCA. However since the 4 momenta of the final state particles are known, if the tree structure is correctly predicted it is likely that the identification of the intermediate particles and B-mesons will be trivial. Another important test is to explore to what level the Belle II software can correct wrong predictions that are made by the model. This would allow its predictive capacity to be widely extended if for example the Belle II software could alter predictions with few mistakes that could potentially be valid graph trees, but invalid physical decay processes. All these steps provide a potential pathway to making a fully fledged alternative for decay reconstruction, that could in the long term replace the FEI.

Appendices

A. Encoder Architecture

In the following the encoder architecture is described in detail. The way the encoder works is presented with a simple example in a pedagogical manner. For this example we consider a dataset containing 6 final state particles and a network where each layer has 64 hidden nodes. Furthermore in this example a shallow version of the network is presented. This means that for the model presented in fig. 3.4 the MLP list contains a single MLP and the model has only one block of layers.

We begin with 4 tensors that are passed to the model. The first one is the feature matrix, a 6×4 tensor that contains the 4 momenta for all the particles. zdiag is a tensor used to fill the diagonal with zeros at the end of the training. The diagonal is filled with zeros since the diagonal of the LCA matrix should always be zero. The other two tensor are used to move from node features to edge labels and vica-versa.

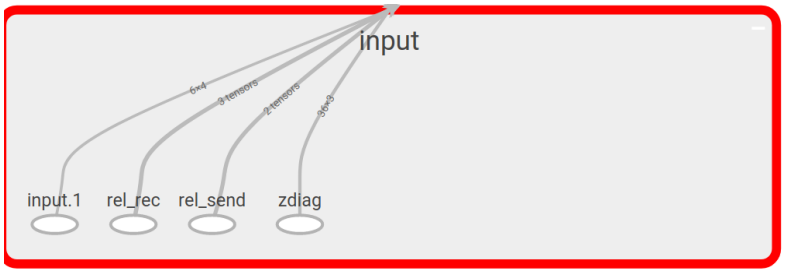


Figure A.1: **inputs** to the forward method of the network

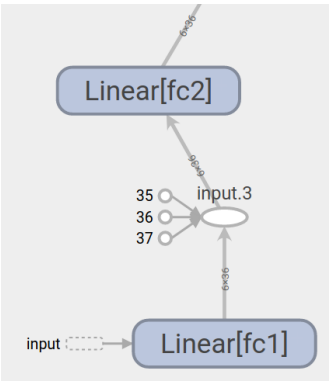


Figure A.2: The first **mlp1:**

The first MLP creates different representations for the node features and for every node.The number of these representations is set by the user when defining the model by the parameter *nhid*. In this particular training $nhid = 64$ as mentioned above, however the number of representations is a hyperparameter of the model. A hyperparameter optimization will give us the optimal number of hidden representations. The example below shows exactly what we mean by representations of the node features (4 momenta):

$$\begin{bmatrix} p_{x1} & p_{y1} & p_{z1} & E_1 \\ p_{x2} & p_{y2} & p_{z2} & E_2 \\ \vdots & \vdots & \vdots & \vdots \\ p_{x6} & p_{y6} & p_{z6} & E_6 \end{bmatrix} \times \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,64} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,64} \\ w_{3,1} & w_{3,2} & \cdots & w_{3,64} \\ w_{4,1} & w_{4,2} & \cdots & w_{4,64} \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{2,1} & \cdots & r_{64,1} \\ r_{1,2} & \cdots & \cdots & r_{64,2} \\ \vdots & \vdots & \vdots & \vdots \\ r_{1,6} & \cdots & \cdots & r_{64,6} \end{bmatrix}$$

Where $r_{i,j}$ is of course the ith representation of the jth node as follows:

$$r_{i,j} = p_{xj}w_{1,i} + p_{yj}w_{2,i} + p_{zj}w_{3,i} + E_j w_{4,i}$$

After the first MLP we have moved to a space of higher dimensions, where each node is described not by 4 numbers anymore, but by 64 different representations of the 4 components of the 4-momentum of each particle.

At this point we need to move from node representations to edge representations. To do this we use the two tensors *rel_rec* and *rel_send* that we gave to the network as input. *rel_rec* and *rel_send* are simply two 36×6 tensors that keep track of the connections inside the fully connected graph. The zeroth dimension is 36 since in a complete graph of 6 nodes has 36 edges.

$$rel\_rec = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, rel\_send = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Now we multiply those matrices with the output of the mlp1 and we essentially copy the representations of each node six times. This is very clear if someones considers the dimensions of the matrices:

$$rel\_rec \times x = receivers$$
$$(36 \times 6) \times (6 \times 64) = (36 \times 64)$$

$$receivers = \begin{bmatrix} r_{1,1} & r_{2,1} & \cdots & r_{64,1} \\ r_{1,1} & r_{2,1} & \cdots & r_{64,1} \\ r_{1,1} & r_{2,1} & \cdots & r_{64,1} \\ \vdots & \vdots & \vdots & \vdots \\ r_{1,6} & \cdots & \cdots & r_{64,6} \\ r_{1,6} & \cdots & \cdots & r_{64,6} \end{bmatrix} senders = \begin{bmatrix} r_{1,1} & r_{2,1} & \cdots & r_{64,1} \\ r_{1,2} & r_{2,2} & \cdots & r_{64,2} \\ r_{1,3} & r_{2,3} & \cdots & r_{64,3} \\ \vdots & \vdots & \vdots & \vdots \\ r_{1,5} & \cdots & \cdots & r_{64,5} \\ r_{1,6} & \cdots & \cdots & r_{64,6} \end{bmatrix}$$

Next we concatenate the receivers and the senders and we end up with a 36×128 tensor. This tensor is the input5 in figure A.3 and is the basis of our label prediction. At this moment we can think of it as a matrix containing 128 different representations for all the 36 edges. Of course at this point these representations are nothing more but some copies of the different representations of the 4 momenta of the nodes that were created by the first MLP. However the edge features is something that will be built on top of this .

The 2nd mlp that follows the node2edge layer is extremely important as it starts to mix representations that originate from different nodes and thus it gives meaning to the edge representation that we want to achieve. This happens in a pairwise manner and is very easy to

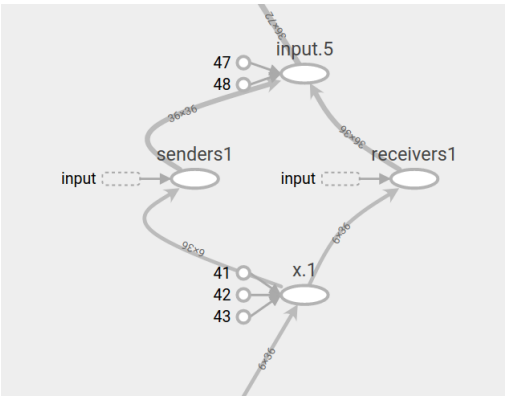


Figure A.3: **node2edge1:** The first passage from nodes to edges happens here.

understand if we consider what kind of representations this 36×64 matrix contains

$$\begin{bmatrix} 1(\times 64) & 1(\times 64) \\ 1(\times 64) & 2(\times 64) \\ 1(\times 64) & 3(\times 64) \\ 1(\times 64) & 4(\times 64) \\ 1(\times 64) & 5(\times 64) \\ 1(\times 64) & 6(\times 64) \\ 2(\times 64) & 1(\times 64) \\ 2(\times 64) & 2(\times 64) \\ 2(\times 64) & 3(\times 64) \\ 2(\times 64) & 4(\times 64) \\ 2(\times 64) & 5(\times 64) \\ 2(\times 64) & 6(\times 64) \\ \vdots & \vdots \\ 6(\times 64) & 5(\times 64) \\ 6(\times 64) & 6(\times 64) \end{bmatrix}$$

The entries of the matrix above are just indices that represent the node from which the representations originate from. For example the entry $[2, 0]$ contains 64 node feature representations of particle's 1 four-momentum while entry $[2, 1]$ contains 64 representations of particle's 3 four-momentum. This matrix is passed to an MLP and the resulting tensor after the linear transformation is a mixture of representations from different node features. Now all the information that could lead us back to the original node features is lost. It is probably fair to say that the output of the mlp2 is a tensor that contains 64 representations for the 36 edges of the graph. We say that it contains edge features since these entries are a mixture of entries of different node attributes.

At this point we create a residual connection as showed in Figure A.5 to keep those first edge representations and use them later. It has been reported that networks start to have worse performance when they get too big. This problem is known as the vanishing gradient problem. Residual connections solve this problem. The tensor that enters the *'incoming'* node at Figure A.5 has the following shape, where these entries are again some indices to help us understand

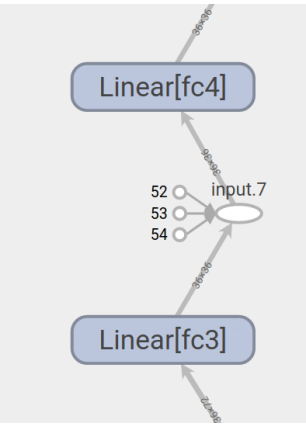


Figure A.4: With the **mlp2** layer we move from node representations to edge representations

better the situation.

$$\begin{bmatrix} (1,1)(\times 64) \\ (1,2)(\times 64) \\ (1,3)(\times 64) \\ (1,4)(\times 64) \\ (1,5)(\times 64) \\ (1,6)(\times 64) \\ (2,1)(\times 64) \\ (2,2)(\times 64) \\ (2,3)(\times 64) \\ (2,4)(\times 64) \\ (2,5)(\times 64) \\ (2,6)(\times 64) \\ \vdots \\ (6,5)(\times 64) \\ (6,6)(\times 64) \end{bmatrix}$$

For example the entry $[6, 0]$ contains 64 representations for the edge between particles 2 and 1.

The edge2node layer gathers all the nodes representations of edges that end up in a specific node. This becomes possible by multiplying $incoming = rel\_rec.t() \times x$ There is also a normalization step where we have : $incoming = incoming/incoming.size(1)$. After the edge2node layer we have ended up with something that looks like a node representation matrix again which is a (6×64) tensor. It seems like we picked some information from all the neighbors with the above procedure and then we created a new node representation. This step is the reason that we say that our approach is a graphical one, since we aggregate information from neighboring nodes.

We pass this output to a third mlp and it seems that it only creates deeper representations of the same information and then from a second node2edge layer to move back to the edge feature space. Everything we described above for those layers stand for here as well.

Like before after the node2edge layer we have a 36×128 tensor but this time we concatenate it with the skip connection we established earlier. Notice that the residual connection was established after the 2nd mlp that followed the first node2edge layer. Finally we use a last mlp layer to reduce the dimensions and a last MLP that makes the predictions for the n classes (3 in our first example) These predictions are turned into probabilities with the help of a Softmax or LogSoftmax function.

After this last layer we use the *zdiag* matrix to fill the diagonal manually with zeros and we

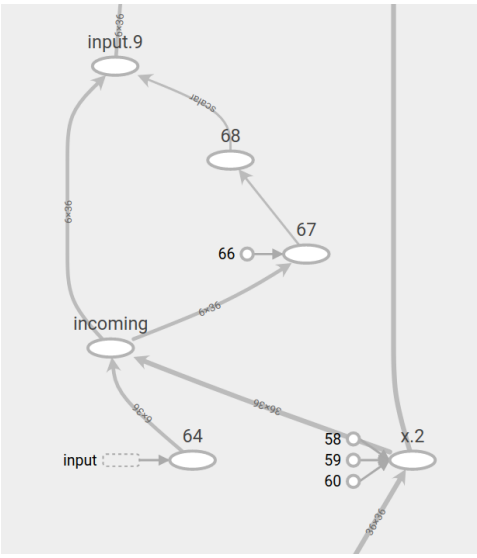


Figure A.5: The **edge2node** layer moves us back to node representations

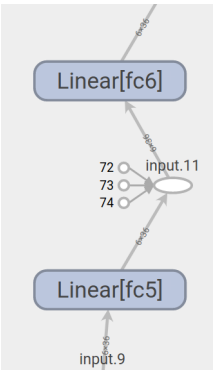


Figure A.6: The **mlp3** creates deeper representations of the reconstructed node features

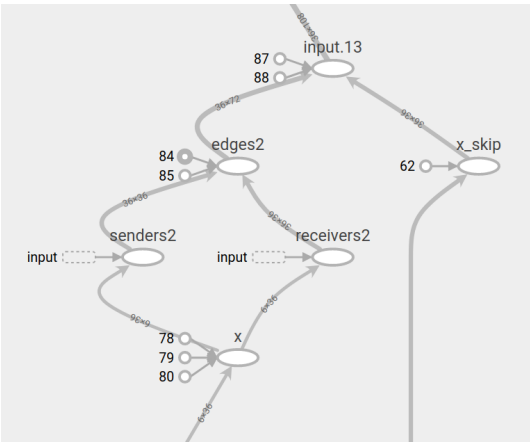


Figure A.7: The second **node2edge** layer moves us back to edge representations

also impose symmetry to our predictions, since the LCA matrix is symmetric by default. These two last steps improve the quality of the predictions.

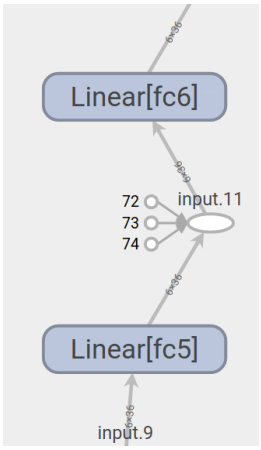


Figure A.8: **mlp4** is the last mlp layer

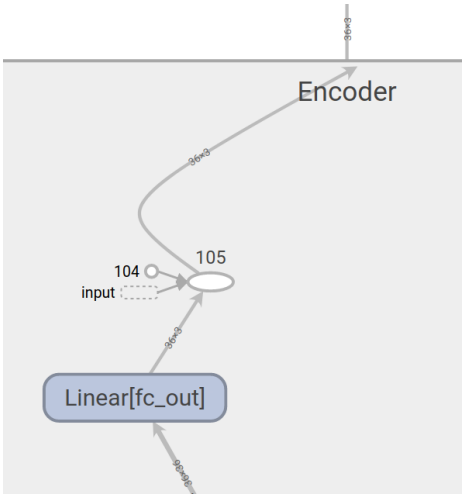


Figure A.9: The last linear layer named **fc_out** makes the predictions for the classes.

B. Training Hyperparameters

Best tuning for mixed datasets						
Set	bsize	lr	dropout	nhid	nBlocks	nMLPs
6par	16	0.0011	0.000744	128	8	14
7par	16	0.000072	0.308	128	8	4
8par	16	0.000185	0.133	80	4	14
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\pi^+$	32	0.001	0.008520	512	4	1
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0))e^+\nu_e$	32	0.001	0.008520	512	4	1
$B^+ \to D^-(\to \pi^-\pi^+\pi^+)\pi^+\pi^+$	64	0.00062	0.1883	128	4	12
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\rho(\to \pi^-\pi^0)$	16	0.00036	0.0624	128	4	12
$B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\omega(\to \pi^+\pi^-\pi^0)\pi^+$	16	0.000485	0.0304	128	4	12
$B^+ \to D^-(\to \pi^-\pi^-\pi^+\pi^0)\pi^+\pi^+\pi^0$	64	0.00117	0.00551	256	4	12
all Phasespace	128	0.001	0.25	1024	2	4
all Belle	128	0.001	0.25	1024	2	4

C. Extra Results

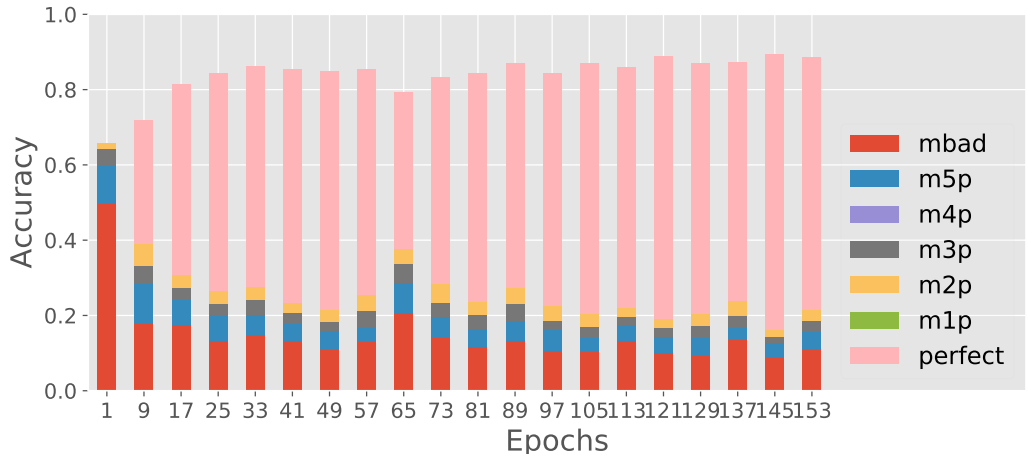


Figure C.1: Performance on the validation set for 7 FSPs

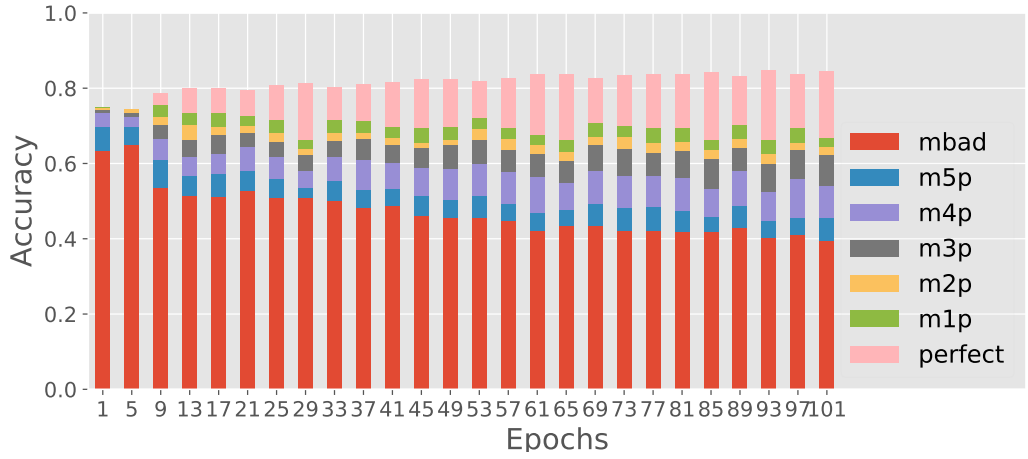


Figure C.2: Performance on the validation set for 8 FSPs

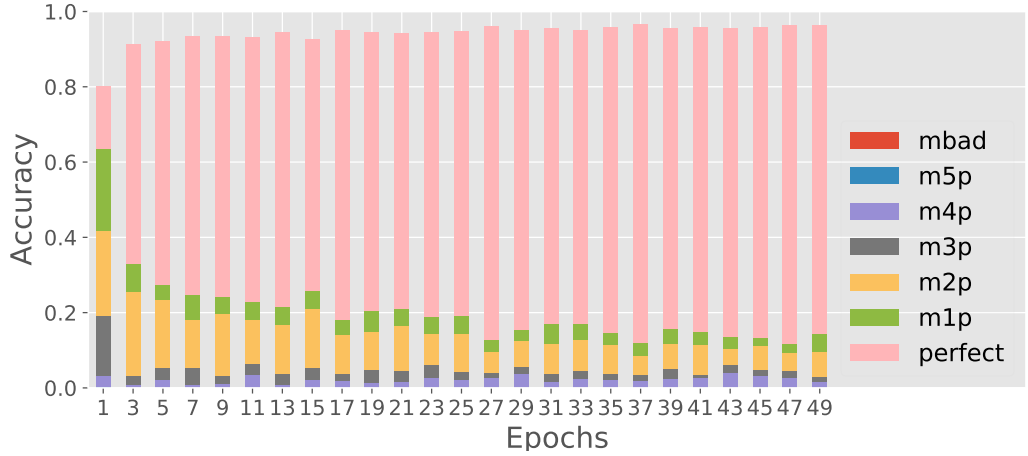


Figure C.3: Performance on the validation set for the 3o3 dataset with 2 missing particles on one side

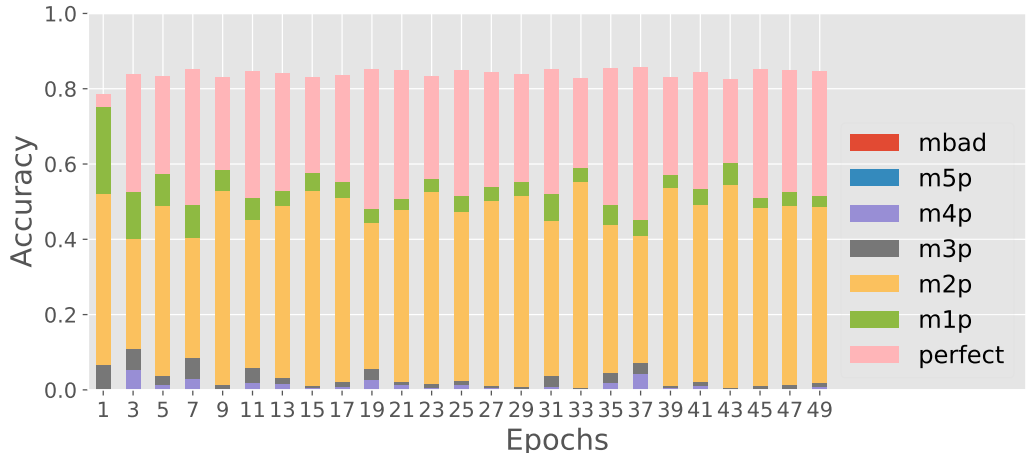


Figure C.4: Performance on the validation set for the 3o3 dataset with 2 missing particles from two intermediate particles

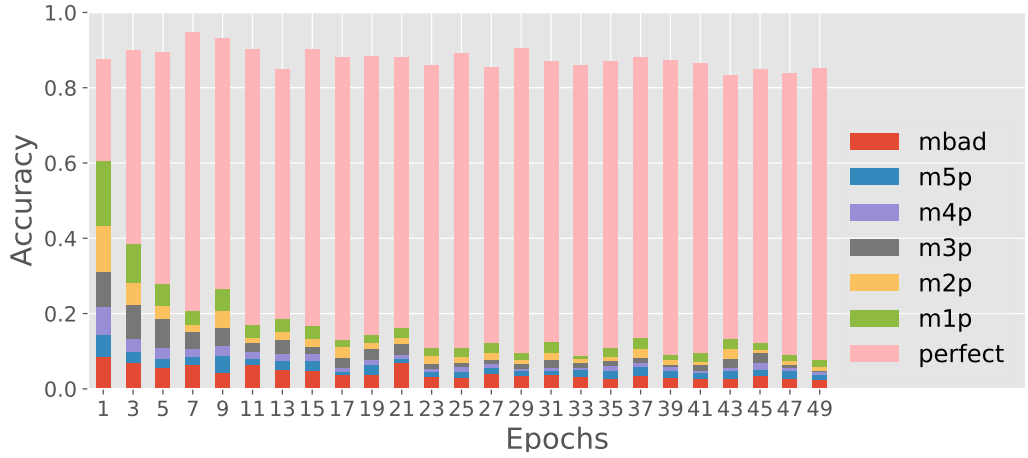


Figure C.5: Performance on the validation set for $B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\rho(\to \pi^-\pi^0)$

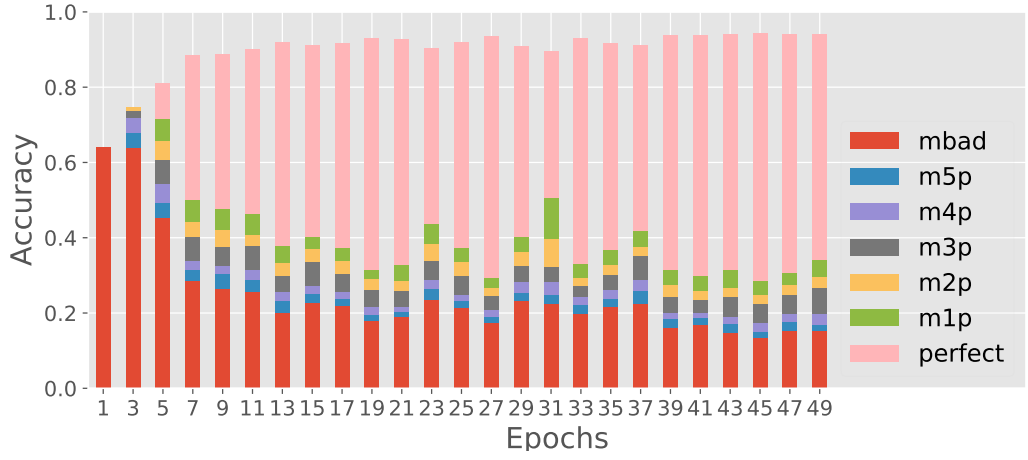


Figure C.6: Performance on the validation set for $B^+ \to \overline{D^0}(\to K^+\pi^-\pi^0)\omega(\to \pi^+\pi^-\pi^0)\pi^+$

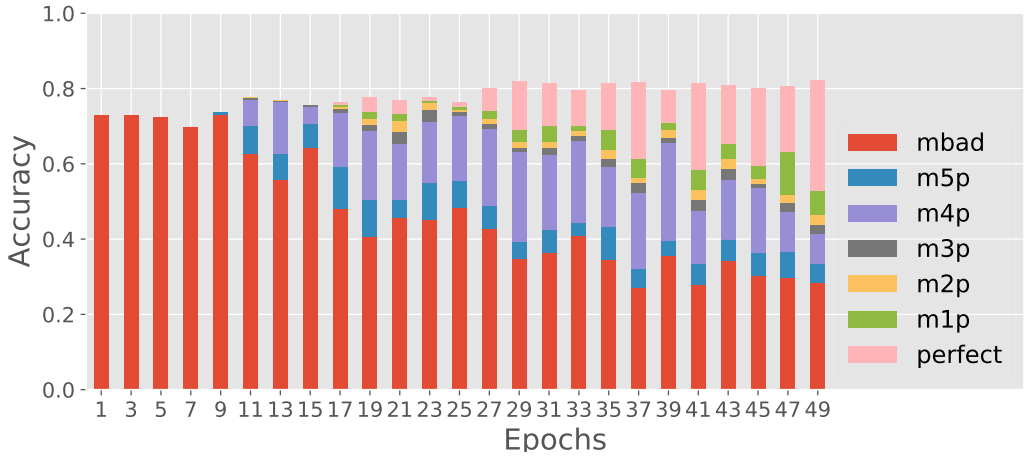


Figure C.7: Performance on the validation set for $B^+ \to D^-(\to \pi^-\pi^-\pi^+\pi^0)\pi^+\pi^+\pi^0$

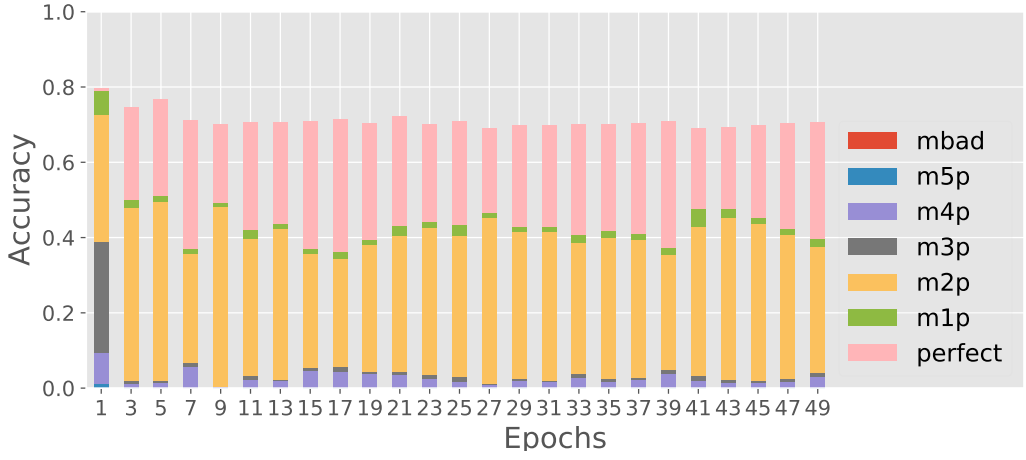


Figure C.8: Performance on the validation set for $B^+ \to D^-(\to \pi^-\pi^+\pi^+)\pi^+\pi^+$

Bibliography

[1] J. L. Hewett. The standard model and why we believe it, 1998.

[2] Makoto Kobayashi and Toshihide Maskawa. CP-Violation in the Renormalizable Theory of Weak Interaction. *Progress of Theoretical Physics*, 49(2):652–657, 02 1973.

[3] Mark Thomson. *Modern particle physics*. Cambridge University Press, New York, 2013.

[4] Charlie Wood. Why do matter particles come in threes? a physics titan weighs in. *Quanta Magazine*.

[5] John Ellis. Physics beyond the standard model. *Nuclear Physics A*, 827(1-4):187c–198c, Aug 2009.

[6] Kazunori Akai, Kazuro Furukawa, and Haruyo Koiso. Superkekb collider. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 907:188–199, Nov 2018.

[7] T. Keck, F. Abudinén, Florian U. Bernlochner, R. Cheaib, S. Cunliffe, M. Feindt, T. Ferber, M. Gelb, J. Gemmler, P. Goldenzweig, and et al. The full event interpretation. *Computing and Software for Big Science*, 3(1), Feb 2019.

[8] E Kou, P Urquijo, W Altmannshofer, F Beaujean, G Bell, M Beneke, I I Bigi, F Bishara, M Blanke, C Bobeth, and et al. The belle ii physics book. *Progress of Theoretical and Experimental Physics*, 2019(12), Dec 2019.

[9] The Belle II collaboration. Belle ii technical design report, 2010.

[10] Thomas Keck. Fastbdt: A speed-optimized multivariate classification algorithm for the belle ii experiment. *Computing and Software for Big Science*, 1, 12 2017.

[11] Thomas Keck. *Machine learning algorithms for the Belle II experiment and their validation on Belle data*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2017.

[12] Thomas Keck. *Machine Learning at the Belle II Experiment*. Springer Theses. Springer International Publishing, 1 edition, 2018.

[13] Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68(1):161–181, Oct 2018.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[15] Diana Militaru. New trends in machine learning for speech recognition. 06 2015.

[16] Felix Stahlberg. Neural machine translation: A review, 2019.

[17] Swakkhar Shatabda, Shadma Shadab, Md Khan, Nazia Neezi, and Sheikh Adilina. Deepdbp: Deep neural networks for identification of dna-binding proteins. 11 2019.

[18] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[19] Char C. Aggarwal. *Neural Networks and Deep Learning.*

[20] Tobias Glasmachers. Limits of end-to-end learning, 2017.

[21] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2012.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[23] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2020.

[24] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2018.

[25] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification, 2019.

[26] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal Self-Attention Network for Graph Classification. *arXiv preprint arXiv:1909.11855*, 2019.

[27] Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. Knowledge graph embedding via graph attenuated attention networks. *IEEE Access*, 8:5212–5224, 2020.

[28] Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 101(5), Mar 2020.

[29] Isaac Henrion, Johann Brehmer, Joan Bruna, Kyunghyun Cho, Kyle Cranmer, Gilles Louppe, and Gaspar Rochette. Neural message passing for jet physics. 2017.

[30] Michael A. Bender, Martín Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. Lowest common ancestors in trees and directed acyclic graphs.

[31] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[32] Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations, 2019.

[33] Frederik Diehl. Edge contraction pooling for graph neural networks, 2019.

[34] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning, 2019.

[35] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

[36] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning, 2018.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[39] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification, 2017.

[40] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[41] Albert Puig and Jonas Eschle. phasespace: n-body phase space generation in python. *Journal of Open Source Software*, oct 2019.

[42] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[43] Glenn F Knoll. *Radiation detection and measurement; 4th ed.* Wiley, New York, NY, 2010.

[44] J. R. Caldwell, R. A. Watson, C. Thies, and J. D. Knowles. Deep optimisation: Solving combinatorial optimisation problems using deep neural networks, 2018.

[45] *Physics Letters B*, page 257–267, Jul 2017.