Faculté de physique et ingénierie
Université de Strasbourg

master PSA Physique Subatomique Astroparticules

Master Science, Mention Physique
Spécialité Physique Subatomique et Astroparticules

Années **2022-2023**

Corentin SANTOS

OPTIMIZATION OF A DEEP GRAPH NEURAL NETWORK TO RECONSTRUCT GENERIC B DECAYS AT BELLE II

Rapport de stage de Master
sous la direction de Giulio DUJANY et Jacopo CERASOLI

Du 27 Février 2023 au 21 Juillet 2023

IPHC Institut Pluridisciplinaire Hubert CURIEN STRASBOURG

cnrs

QMat QUANTUM SCIENCE AND NANOMATERIALS

Abstract

This work presents the results of a fourteen weeks internship on the optimization of a deep graph neural network in the context of the search for New Physics (NP) at the Belle II experiment. The neural network is the core of an algorithm named GRAFEI to reconstruct B decays. A powerful probe for the search of NP is the $B \to K\nu\bar{\nu}$ decay channel because of its rarity and the precision of its branching ratio prediction in the Standard Model (SM). The Belle II experiment is the only experiment capable of detecting such a decay because of its cleanness and hermeticity. Moreover, the production of a partner B during e^+e^- collisions at the SuperKEKB collider allows to recover the missing energy from invisible particles in the final state, a technique name Full Event Reconstruction (FER), specific to Belle II. During this internship, starting from a 15% B reconstruction efficiency, the multiple optimizations performed on the GRAFEI get the performances to 21%. This is about 6 times the reconstruction efficiency of the main reconstruction algorithm currently used at Belle II, the Full Event Interpretation (FEI).

Abstract

Ce travail présente les résultats d'un stage de quatorze semaines sur l'optimisation d'un réseau de neurones à graphes profonds dans le cadre de la recherche de nouvelle physique à l'expérience Belle II. Le réseau de neurones est au cœur d'un algorithme nommé GRAFEI pour reconstruire les désintégrations de B. Une sonde puissante pour la recherche de nouvelle physique est le canal de désintégration $B \to K\nu\bar{\nu}$ en raison de sa rareté et de la précision de sa prédiction du rapport de branchement dans le modèle standard. L'expérience Belle II est la seule expérience capable de détecter une telle désintégration en raison de sa propreté et de son herméticité. De plus, la production d'un partenaire B lors des collisions e^+e^- au collisionneur SuperKEKB permet de récupérer l'énergie manquante des particules invisibles dans l'état final, une technique nommée FER, propre à Belle II. Au cours de ce stage, partant d'une efficacité de reconstruction du B de 15%, les multiples optimisations effectuées sur le GRAFEI portent les performances à 21%. C'est environ 6 fois l'efficacité de reconstruction du principal algorithme de reconstruction actuellement utilisé à Belle II, le FEI.

Acknowledgements

I would like to thank my supervisors, Dr.Giulio Dujany and Dr.Jacopo Cerasoli, who constantly listened to me, helped me, and answered even the dumbest questions I had. I also thank the whole Belle II group for their welcome, the ambiance which creates a very nice working environment, and the time they spend to help me with rehearsals or to correct the reports, as well as their implication during the doctoral school application and auditions.

Contents

Chapter 1

Introduction

This internship took place in the context of the search for physics beyond the Standard Model (SM) at collider experiments. The SM is currently the most accomplished theory to describe particle physics. During its history, prediction by the SM and experimental results were compared and the prediction proved to be accurate [1]. While this theory is very successful, some points remain to be described such as the presence of dark matter and dark energy [2, 3], or the matter/antimatter asymmetry [4]. To answer those questions, physicists have two methods: either using more energetic colliders, or increasing the precision of the measurements to compare the predictions with the experimental results. Those are described as the energy and intensity frontiers.

The indirect search for New Physics (NP) is based on the possible appearance of NP particles in quantum loops during particle physics processes, such as decays or many-body interactions. These contributions from NP, in addition to the ones from the SM, can modify the probability of those events happening with respect to the SM predictions. The NP particles appearing in quantum loops can be heavier than the available energy of the collider, since they are produced off-shell, making the intensity frontier sensitive to higher energy scales than the energy frontier. However since the NP contributions can be small it is important to consider rare decays, where also the SM contribution is small, and to make precise measurements.

In this respect the $B \to K\nu\bar{\nu}$ decay is particularly promising to look for NP: in the SM it is a rare process dominated by quantum loops with a branching fraction $\mathcal{O}(10^{-5})$ [5]. The contributions from NP can thus have a sizeable impact. However, it is almost invisible, with the two neutrinos being undetected and it has not been observed yet because of the inherent difficulty of the task [6].

The Belle II experiment [7] located at the SuperKEKB collider in Tsukuba, Japan, looks for NP by providing data with the highest instantaneous luminosity ever achieved [8]. There, e^+e^- pairs are collided at the Center of Mass (CoM) energy of 10.58 Gev, corresponding to the mass of $\Upsilon(4S)$, a meson that almost always decays into a pair $B\bar{B}$[9], making it an ideal place to study B-physics. Moreover, the characteristics of Belle II, such as its hermeticity and the clean environment it offers, allow for the study of B decays with invisible particles, in particular $B \to K\nu\bar{\nu}$.

While it is rather common to study three-body decays with one invisible particle, having two particles undetected in the problem makes it impossible to solve. Or at least, not while considering the B as isolated. For such problems, it is usual to use the knowledge of the initial state conditions in e^+e^- collisions and the conservation of the four-momentum to recover information on the missing energy, a technique known as Full Event Reconstruction (FER). At Belle II, considering that the B of interest, the signal B_{sig}, is produced via $\Upsilon(4S) \to B\bar{B}$ with a companion, the tag B_{tag}, it is possible to constraint information about the signal via full reconstruction of the tag. This is exactly what the algorithm called Full Event Interpretation (FEI) does at Belle II [10]. However, this algorithm has an efficiency of just a few percent, since the decay channels need to be hard-coded. While about 10000 decays are considered these correspond to only 15% of the total B branching fraction.

This report explains the benefits of a novel deep learning algorithm for Belle II, named GRAFEI (graph-based Full Event Interpretation), in this context of precision measurements and

new physics studies, and how we optimized this algorithm for the reconstruction of generic B meson decays.

This thesis is divided in four chapters. Chapter 2 defines the theoretical background needed to understand the results, with a deeper description of Belle II and the FER in Section 2.1, an overview of the $B \to K\nu\bar{\nu}$ channel in Section 2.2, followed by a quick recap on the FEI in Section 2.3. Then, a brief introduction to the elements of machine and deep learning in Section 2.4, to better understand the algorithm of GRAFEI in Section 2.5. This will be completed by the exposition of the results in Chapter 3. First, on the study of training time in Section 3.1. Then on the optimization of the input variables to the system in Section 3.2, like the mass of the particle or its momentum. The section 3.3 will be about the particles to reconstruct, for example to study if one should use the π^0 as Final State Particles (FSPs) or to reconstruct them. Finally, the optimization of the hyperparameters, such as the deepness of the neural network, in Section 3.4. Lastly, we will conclude on the work done during this internship on Chapter 4 and open on the future of this problem.

Chapter 2

Method

2.1 Belle II

Belle II is a particle physics experiment taking data at the SuperKEKB collider. This is an e^+e^- collider where the positrons are accelerated at 4 GeV and the electrons at 7 GeV. Since it is a lepton-lepton collider, it does not suffer from pile-up and event-induced background as hadron colliders. The collider effectively reaches and energy in the CoM of 11 GeV, corresponding to the $\Upsilon(4S)$ meson, a $b\bar{b}$ bound state. This resonance will almost always decay in a $B\bar{B}$ pair, with about half of the cases being B^+B^- and the other half being a $B^0\bar{B}^0$ [9]. Belle II thus offers a clean environment which is ideal to study partially invisible B decays. Because of the asymmetry in the acceleration of the two leptons, the B mesons produced are boosted "forward", *i.e.* in the direction of the higher energy beam.

The first point of interest in using Belle II is its hermeticity. In this case, it is important to cover the low angles near the beam because of the boost and the (non-)interaction of invisible particles such as neutrinos. This task is carried out by a series of sub-detectors. From the nearest to the collision point to the furthest: the Vertex Detector (VXD) which is composed of the Silicon Vertex Detector (SVD) and silicon PiXel Detector (PXD); the Central Drift Chamber (CDC), the Time Of Propagation counter (TOP) and Aerogel Ring Imaging CHerenkov (ARICH) for particle identification; an Electromagnetic CaLorimeter (ECL) and finally the K_L^0 and Muon detector (KLM). The disposition of the detectors can be seen in figure 2.1.

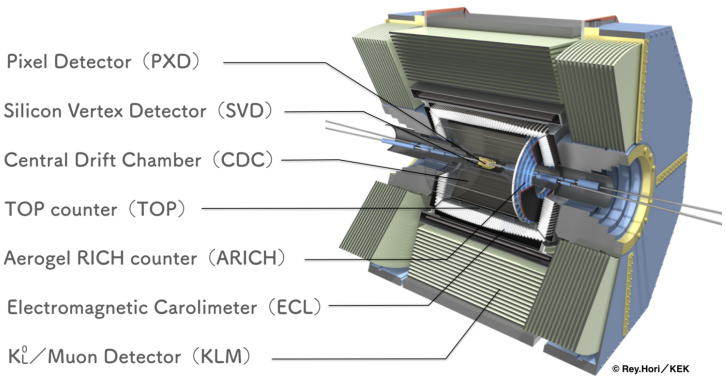


Figure 2.1: Overview of the Belle II detector [11].

The FER technique is built on the fact that, at Belle II, the B are always produced in pairs from $\Upsilon(4S)$ decays, as shown in figure 2.2. Thus, to study the decay $B_{\text{sig}} \to K\nu\nu$ one needs to detect a kaon and fully reconstruct the B_{tag}, which can only be done thanks to the hermeticity of Belle II. The reconstruction of the B_{tag} is done via the FEI software, described in section 2.3.

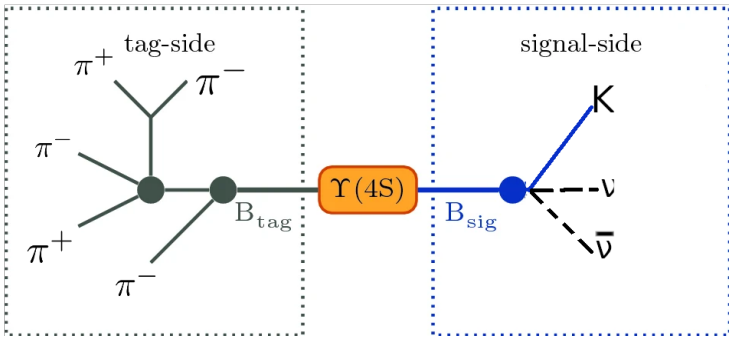
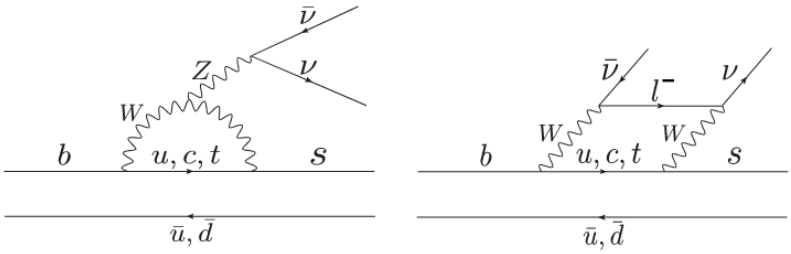


Figure 2.2: Schematic view of the FER. In order to study signal sides with missing energy,, *e.g.* the decay $B_{\text{sig}} \to K\nu\bar{\nu}$, one needs to reconstruct the tag side, *e.g.* $B^-_{\text{tag}} \to D^0\left(\to K^0_s\left(\to \pi^-\pi^+\right)\pi^-\pi^+\right)\pi^-$, to constrain the kinematic [10].

2.2 $B \to K\nu\bar{\nu}$

As specified previously, this channel is hard to detect because of the presence of two invisible particles. Moreover, $B \to K\nu\bar{\nu}$ is a powerful probe for NP because it is a Flavor Changing Neutral Current (FCNC), as shown in figure 2.3, which can only proceed through loops and is highly suppressed in the SM by the so-called Glashow-Illiopoulos-Maiani (GIM) mechanism. The predicted branching ratios are about 10^{-5}, $\text{BR}\left(B^+ \to K^+\nu\bar{\nu}\right) = (5.67 \pm 0.38) \times 10^{-6}$ and $\text{BR}\left(B^0 \to K^0\nu\bar{\nu}\right) = (4.67 \pm 0.35) \times 10^{-6}$ [5]. Differences in the rate of FCNC processes may be explained by the presence of new physics inside the loops. Many new physics models describe how deviations from the SM could influence this channel [12]. Another point of interest for this channel is its cleanness. The fact that neutrinos are chargeless and cannot couple to the photon makes it a cleaner decay compared to similar ones involving charged leptons like $B \to Kl^+l^-$. Indeed, the presence of photons' coupling increases the theoretical uncertainties [12].



(a) penguin (b) box

Figure 2.3: Leading orders of the $B \to K\nu\bar{\nu}$ decay in the SM.

2.3 Full Event Interpretation

Currently, the main tool used to perform the FER is the FEI algorithm. This machine learning-based algorithm uses gradient-Boosted Decision Trees (BDTs) to reconstruct the tag-side via a six stages approach. Those stages are defined such that the output of the previous stage is used as input for the following one, as illustrated in figure 2.4. The algorithm starts from information on the FSPs, *i.e.* tracks from charged particles, clusters, and displaced vertices from neutral ones. Then, FSPs are combined to form intermediate particles with an associated likelihood of being correctly identified. This likelihood is then used as input for the next stage. This process is repeated up to the B candidate, which also has an identification likelihood associated with it. A complete description of the algorithm and its construction can be found in the reference [10].

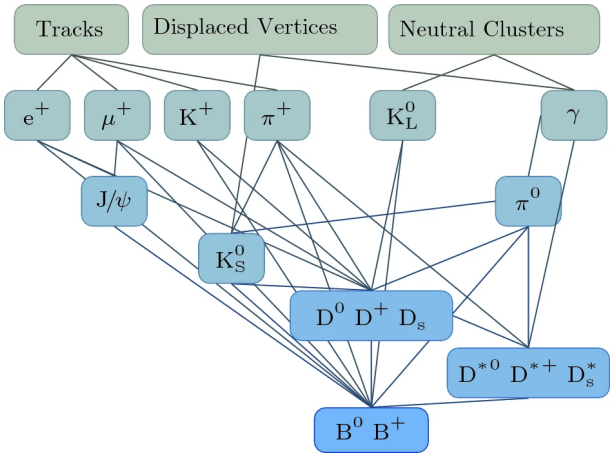


Figure 2.4: Sketch of the six stages based reconstruction performed by the FEI [10].

However, the main drawback of this tool is the fact that the decays need to be explicitly hard-coded at each stage. In the case of the FEI, about 10000 B decays are considered, which represent only 15% of the total branching fraction of B mesons. For this reason, the algorithm only has a reconstruction efficiency of a few percent. Regarding the question of $B \to K\nu\nu$, or even on a larger scope the question of NP, it is useful to overcome this limitation if one wants to increase the sensitivity. That is where deep learning comes into action.

2.4 Some elements of Deep Learning

2.4.1 About Neural Networks

Artificial Neural Networks (ANNs) provide the foundation for a particular machine learning technique known as deep learning. Inspired by biological neural networks, it can essentially be reduced as a set of nodes and edges, the nodes being neurons and the edges representing connections between neurons. The simplest ANNs is called perceptron. Perceptrons are composed of $n \in \mathbb{N}^*$ nodes corresponding to the component of an input vector $\mathbf{x} \in \mathbb{R}^n$, and an output node. The inputs are weighted and summed. The set of weights is represented via the matrix $\mathbf{W} \in M_{1,n}(\mathbb{R})$. The perceptron then acts as a *non-linear activation function* $f : \mathbb{R} \to \mathbb{R}$ which takes as input $\mathbf{W}, \mathbf{x}$ and a bias $b \in \mathbb{R}$. Thus, the output of the perceptron is simply $f(\mathbf{W}\mathbf{x} + b)$. The set of weights and biases are called the *parameters* of the model. A sketch of the perceptron principle is in figure 2.5(a).

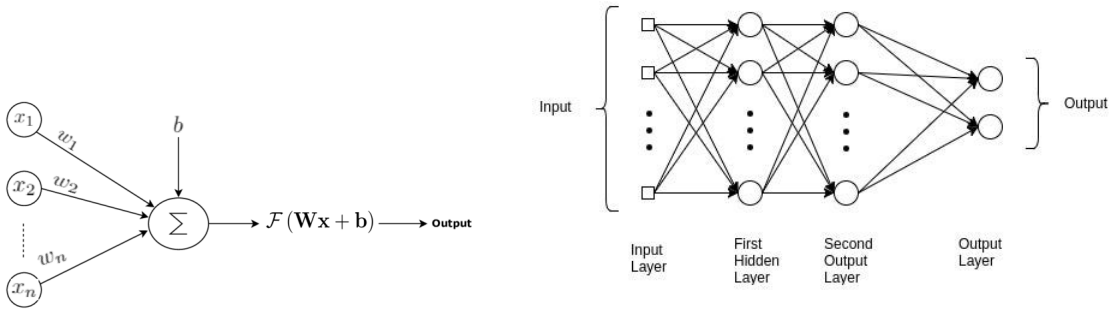
One can generalize the perceptron to multiple output nodes. The function f then generalises to $\mathcal{F} : \mathbb{R}^m \to \mathbb{R}^m$, a non-linear activation function which takes as input $\mathbf{W}$, a $m \times n$ matrix with $m \in \mathbb{N}^*$ the number of output nodes, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$.

Finally one can stack several layers of perceptrons with multiple output nodes obtaining a MultiLayer Perceptron (MLP), represented in figure 2.5(b). This network will have an input layer an output layer and several hidden layers in between. Each neuron in a layer will take as input the output of the previous layer.

The deepness of the algorithm comes from the hidden layers, which may be present in the MLP. Parameters related to the geometry of the model, like the number of hidden layers or their dimension, are known as *hyperparameters*.

2.4.2 About training

Training refers to the optimization of the model's parameters (weights and biases) with the use of *labeled* data. The main idea is to give a training input to the model, then compare the output to what was expected. The difference between the model output and the data's label is quantified via a cost, or *loss* function $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$, which depends on the parameters of the model. The goal of the training is to minimize the loss function via the so-called *gradient-descent*



(a) (b)

Figure 2.5: (a) Sketch of a simple perceptron with a bias and its activation function [13]. (b) Sketch of a MLP with hidden layers [14].

technique [15]. As its name implies, the first step is to compute the gradient of the loss function, $\nabla\mathcal{L} : \mathbb{R}^m \to \mathbb{R}^p$, where $p \in \mathbb{N}^*$ is the number of parameters. The opposite of the gradient will give the direction with the *steepest descent* to move in the phase space to minimize locally the cost function. This gradient is computed for the $N \in \mathbb{N}^*$ events and the descent is performed using the average gradient over all the events. The distance traveled in the phase space between each gradient computation is known as the *learning rate*, $\varepsilon \in \mathbb{R}_{>0}$. To summarize, the parameters' vector $\mathbf{w} \in \mathbb{R}^p$ will be *updated* into a vector $\mathbf{w}' \in \mathbb{R}^p$ via

$$\mathbf{w}' = \mathbf{w} - \frac{\varepsilon}{N} \cdot \sum_{i=1}^{N} \nabla\mathcal{L}_i \tag{2.1}$$

This whole process is illustrated in figure 2.6.

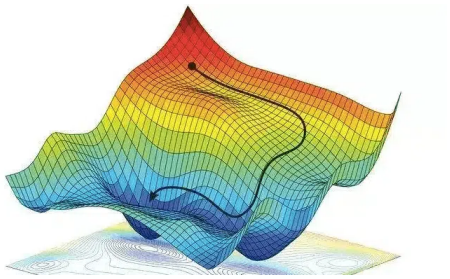


Figure 2.6: Visualisation of the gradient-descent technique in the case of two parameters. The surface represented is an arbitrary loss function [15].

One must note that this technique can only find a local minimum. There is nothing to guarantee that the found minimum is a global one.

One can also split the training set into $k \in \mathbb{N}^*$ randomly generated subsets called *batches* and average the gradient of the loss $\mathcal{L}_i, i \in [\![1, k]\!]$, for each subset over all batches. This splitting will reduce the computation complexity and the training time [16]. This is possible because the approximation

$$\frac{1}{N}\sum_{i=1}^{N} \nabla\mathcal{L}_i \approx \frac{1}{k}\sum_{j=1}^{k} \nabla\mathcal{L}_j \tag{2.2}$$

is verified when k is large enough. Thus, the equation 2.1 will be redefine as

$$\mathbf{w}' = \mathbf{w} - \frac{\varepsilon}{k} \sum_{i=1}^{k} \nabla\mathcal{L}_i \tag{2.3}$$

This technique is known as the *stochastic gradient descent* and will be used for the GRAFEI.

Training on the complete dataset is an *epoch*. The more the epochs, the more the performances of the model increase, because of the parameters' optimization, but the longer the training time takes.

Moreover, each of the hidden layers increases the number of parameters to better fit the problem. While it will increase the performances of the algorithm on the training dataset, it also increases the risk of *overfitting*. Overfitting corresponds to the case where a model has too many parameters for what it needs to predict. Equivalently, the model is overfitted when it learns the noise in the training dataset, becoming very sensitive to noise addition or removal. An illustration of overfitting can also be found in figure 2.7.

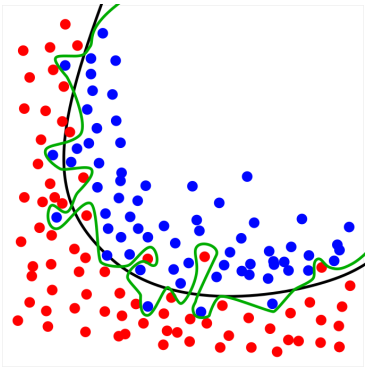


Figure 2.7: The green line represents an overfitted model, with a 100% success rate with the training dataset but very sensitive to the addition of noise, and the black line represents a "simpler" model with more errors with respect to the dataset but fewer sensitive to the noise [17].

To prevent overfitting, also called *overtraining*, one can increase the size of the dataset or randomly set to 0 some rows of the parameters' matrix $\mathbf{W}$ with a certain probability at each epoch. This method is known as *dropout* [18].

Another well-known issue with deep learning could have repercussions on the training of a too-deep network: the *vanishing-gradient problem* [19]. This problem is related to the derivative of the activation function. If the many small derivatives are multiplied the total gradient becomes too small and the parameters cannot be updated. To avoid this, many solutions exist. Among them is the addition of *skip connections*, transforming our neural network into a *Residual Neural Network* [20]. Those connections save the input $\mathbf{x}$ before applying the map $\mathcal{F}$ and simply add this to the output $\mathcal{F}(\mathbf{x})$. A sketch showing a skip connection can be seen in figure 2.8.

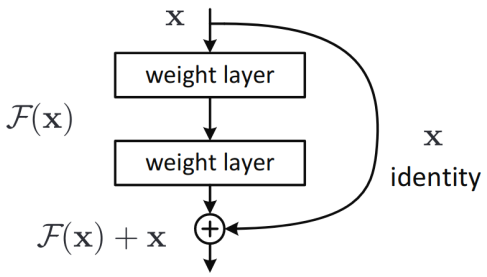


Figure 2.8: Schematic representation of a skip connection. Modified from [20].

2.5 Graph-based Full Event Interpretation (GRAFEI)

The fact that the decays have to be hard-coded is the main drawback of using the FEI for our problem. More specifically, one wants an algorithm capable of reconstructing the decay tree from the FSPs without any assumptions on the intermediate particles. The task of creating such an

algorithm can be summarized in a question: is it possible to reconstruct a tree from its leaves only? The answer to this question is: yes, via Graph Neural Networks (GNNs) [21]. GNNs are a type of ANNs which act on graphs.

Graphs $\mathcal{G}$ are composed of two sets of elements which are nodes $\mathcal{V} = \{(\nu_i)_{i\in I}\}$ connected by edges $\mathcal{E} = \{e_{ij} = (\nu_i, \nu_j), (i,j) \in I^2\}$. As shown in figure 2.9, decay trees can be described by rooted directed acyclic tree graphs. One way to represent a graph algebraically is via its adjacency matrix $\mathbf{A}$, a matrix whose elements correspond to a pair of nodes $(i,j) \in I^2$ in the graph with value 1 if the two nodes are linked and 0 otherwise. While this matrix perfectly describes the topology of the graph, this implies knowledge about the number of intermediate particles which is not the case in particle physics experiments, where only the FSPs are detected. A different representation that solves this problem is called the Lowest Common Ancestor (LCA) matrix. The FSPs correspond to the rows and columns of the matrix while each element is the lowest ancestor common to both particles. To avoid having a unique identifier for each ancestor, it was decided to use a system of classes. As for the FEI with the stages, each intermediate particle belongs to a certain class (5 for B mesons, 4 for D^*, 3 for D, 2 for K_s^0, 1 for π^0 or J/ψ and 0 for particles not in the decay tree). This new representation of the LCA is called the Lowest Common Ancestor Stage (LCAS) matrix.

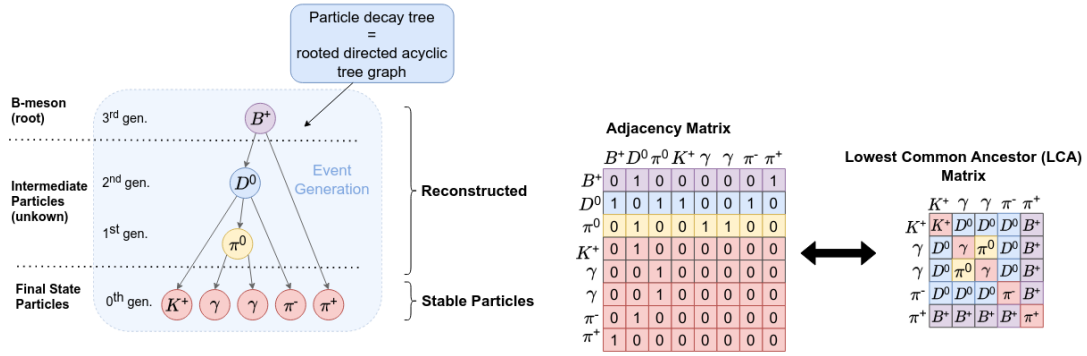


Figure 2.9: Example of a B-decay represented by an adjacency matrix and a LCA matrix [22].

Inside graphs, information is stored in the nodes, the edges, or the graph as a whole. Thus $\mathcal{G}$ is characterized by three sets of features: node, edge, and global. In order to reconstruct the LCAS matrix, the input of the network should be a *fully connected* graph representing the FSPs.

The node features are information about the particle, such as its energy or mass, the edge features are quantities depending on pairs of particles, for example, the angle between the momenta of the particles, and the global features are information about the overall system, *e.g.* the number of particles in the graph. The output of the model is a graph with the same structure as the input but updated values of its features.

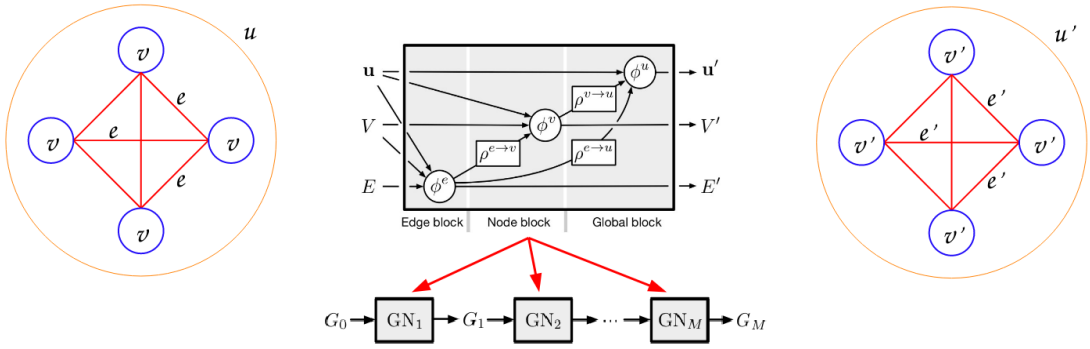


Figure 2.10: Schematic view of the GRAFEI with its input, the model and the output [23].

The graph-based FEI or GRAFEI proceeds as follows: starting from a fully-connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as an input for the GNNs, a series of graph network blocks $(GN_l)_{l\in I}$ (at least two, an

input and an output) will apply series of functions ϕ^a and $\rho^{a\to b}$ on the features of the graphs. Here, a and b indicate edge, node, or global features, e, ν, u. Each GN_l is composed of three (sub-)blocks: the edge, the node, and the global block in this exact order. The output edge features are used to predict the element of the LCAS matrix.

The functions $\phi^i : \mathbb{R}^n \to \mathbb{R}^n, n \in \mathbb{N}^*, i \in \{\text{e},\nu,\text{u}\}$, are called the *update functions*. For the GRAFEI, those update functions will be fully connected networks. The functions $\rho^{i\to j} : \mathbb{R}^n \to \mathbb{R}, (i,j) \in \{\text{e},\nu,\text{u}\}^2$ are called the *aggregation functions*. In the case of the GRAFEI, they compute the mean of the output of an update function and pass it to the input of the next update function. Inside each GN_i, $i \in I$, the first function applied is the update function ϕ^e which takes as entry the features of the k-th edge e_k, the features of the nodes it links (ν_{l_k}, ν_{r_k}), the global features u (broadcasted to every edge) and it returns the updated edge features e'_k. This is where the *edge block* ends. Afterward, the edge features are averaged over the edges via the aggregation functions, $\rho^{e\to\nu}$ and $\rho^{e\to u}$, respectively for edges connected to the same node ν and for all the edges in the graph. Then, they are passed to the update functions ϕ^ν and ϕ^u. The *node block* is composed of the update function ϕ^ν, which takes as input the output of the previous block via the aggregation function $\rho^{e\to\nu}$, the node features to update, and the broadcasted global features. Similarly, the node features are then averaged over all the nodes in the graph via the aggregation function $\rho^{\nu\to u}$ and passed to the last update function, ϕ^u in the *global block* where the output of the two previous blocks and the global features are used to compute new values of the latter.

This description is illustrated in figure 2.10. One can then go from LCAS to the adjacency matrix and recover the decay tree.

The tasks at hand are now to optimize both the features given as input and the hyperparameters of the network to improve the performances of the GRAFEI. The performances of the model will be evaluated using two metrics: the **average perfect LCA**, which is the percentage of correctly reconstructed LCAS in the dataset, and the loss function $\mathcal{L}$. The loss function used here is the *cross-entropy* loss, also known as log loss. The cross-entropy measures the effectiveness of a classification model which gives a probability as an output. In the GRAFEI case, the outputs of the model are a real number for each class, which are then mapped to $[0,1]$, therefore defining probabilities [24]. The loss for one LCA element i is then computed as

$$\mathcal{L}_i = -\sum_{k=1}^{C} q_{i,k} \log\left(p_{i,k}\right) \tag{2.4}$$

where C is the total number of classes (6 in this case), $(q_{i,k})_{k\in[\![1,C]\!]}$ equals 1 if k is the right class, 0 otherwise, and $(p_{i,k})_{k\in[\![1,C]\!]}$ is the probability given by the model. The loss is then averaged over all the elements of all the decays in a training batch. *Nota Bene*: the dependency of the loss function on the parameters' vector $\mathbf{w}$ is hidden inside the dependencies of $(p_{i,k})_{k\in[\![1,N]\!]}$.

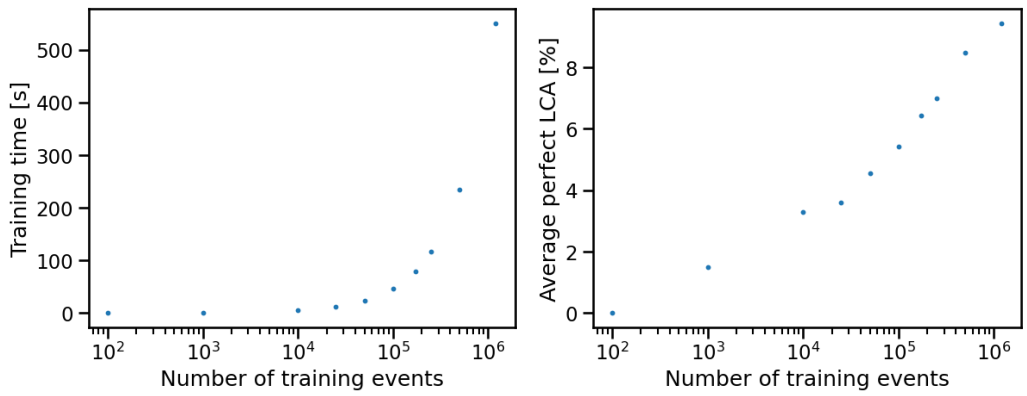
The training dataset was generated via Belle II Analysis Software Framework, or BASF2 [25], and each dataset is split into two sets: the training and the validation set. As its name implies, the goal of the validation set is to guarantee that the obtained values are non-biased. It can also be used to assess overtraining.

Chapter 3

Results

3.1 Training time versus sample size

The first task consists in finding an optimal number of training (and validation) events to avoid long training during the features and hyperparameters optimization. Indeed, training over a sample of around twelve million events and one hundred epochs takes approximately one week. This task is performed by training the model over one epoch but for a different number of training samples, starting from one hundred up to one million. One can find a graph summarizing this study in figure 3.1.



(a) (b)

Figure 3.1: (a) Graph of the training time for various training sample sizes. (b) Graph of the average perfect LCAS in validation for various training sample size.

Considering the training time for 1 epoch, one can extrapolate by multiplying the training time by 100 to get the training time over 100 epochs. The conclusion was that 500 000 events could serve right for a quick training while 1 000 000 would be of use for longer training, which means *a priori* a higher average perfect LCAS. Testing the extrapolation, it appears that the training is faster and more accurate than expected by the extrapolation as shown in table 3.1.

Table 3.1: Table summarizing the execution time over 100 epochs for different sample sizes

Training sample size	Time	Avg perfect LCAS [%]
12M	$\approx$ 1 week	19.13
1M	$\approx$ 17h	16.74
500k	$\approx$ 8h	15.94

While the reason for this speed was not investigated, a consensus for the training was found: 1 000 000 samples over 50 epochs is optimal for the tasks to come.

3.2 Input optimization

The first part of the model optimization has to be done on the physical quantities given to the model as input, the *features*. As explained previously, GRAFEI uses three types of features: node, edge, and global.

For the rest of the description, the coordinate system will be: z is the direction of the beam, and (x, y) are the coordinates of the transverse plane.

3.2.1 Node features

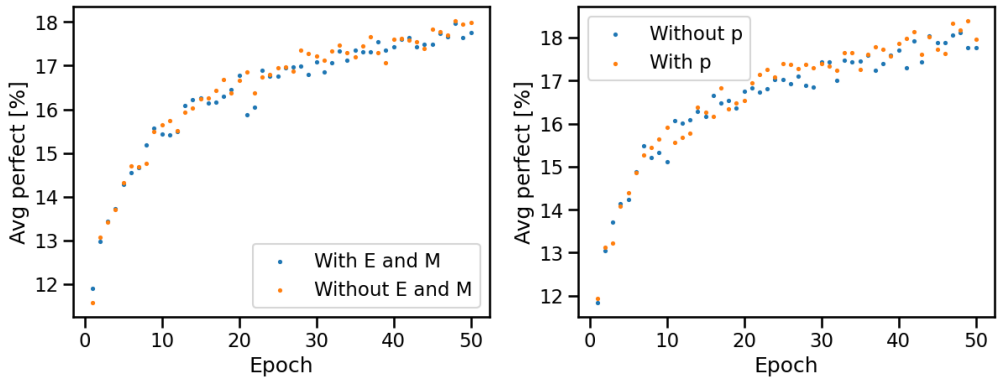
The model's input for the charged particles, which leave tracks in detector, are: the Particles ID (PIDs); the magnitude of the momentum p and of the transverse momentum p_T; the three components of the momentum (p_x, p_y, p_z); the radial and z component of distance to the impact point $(\mathrm{d}r, \mathrm{d}z)$; the energy and the mass (E, M); the charge q. Neutral particles are characterized by clusters in the calorimeter meaning the input to the model is different:CLUSTERNHITS is the number of cells hit in the ECL; CLUSTERTIMING returns the time between the collision and the moment the cluster is registered in the ECL; CLUSTERE9E21 indicates whether the activation is due to photons or hadrons. A detailed description is in reference [26]. This work will mainly focus on the charged particle features optimization by adding or removing them to study how the average perfect LCAS will evolve.

The first step involves the PIDs, which quantifies the likelihood for a given charged track to have a certain mass hypothesis. This probability is computed via the likelihood that a particle is either an electron, a muon, a kaon, a pion, a proton, or a deuteron, divided by the sum of all these likelihoods. For example, the ELECTRONID, the probability that the targeted particle is an electron, is

$$\frac{\mathcal{L}_e}{\mathcal{L}_e + \mathcal{L}_\mu + \mathcal{L}_K + \mathcal{L}_\pi + \mathcal{L}_p + \mathcal{L}_d} \tag{3.1}$$

The likelihood is computed using information from all sub-detectors. Due to an error in the computation of the PIDs in the simulation, one had to remove some sub-detectors, namely the SVD for all particles to identify, and the TOP for the electrons.

The results of the study correspond to the second point in figure 3.3 and show a clear enhancement of the model's performances. After this study was done, the time came for the feature optimization. Starting with the question of energy E and mass M: can the model perform as well without giving it the mass hypothesis? One wants to remove the mass hypothesis because it may be wrong and it may affect the reconstruction efficiency. Indeed, the PIDs are not well reconstructed by the Monte-Carlo simulation. When comparing the training with and without (E, M) the answer is clearly yes as one can observe in figure 3.2(a).



(a) (b)

Figure 3.2: (a) Graph of the average perfect LCAS in validation with and without E and M. (b) Graph of the average perfect LCAS in validation with and without p.

The coordinates (p_x, p_y, p_z) as well as p_T are all related, notably via p. Removing p_x and p_y proved that the redundancies have close to no impact on the performances of the model. The remaining question was about p, the magnitude of the 3-momentum. While one can see in the study shown in figure 3.2(b) that keeping or removing it seems to not have a big impact, there was still a doubt whether one should use it when coupled with its error p_{err}.

The last study of this section rather indicates that the error, p_{Err}, does not have an impact on the LCAS average perfect, allowing us to discard p from the node features, as presented in figure 3.3.

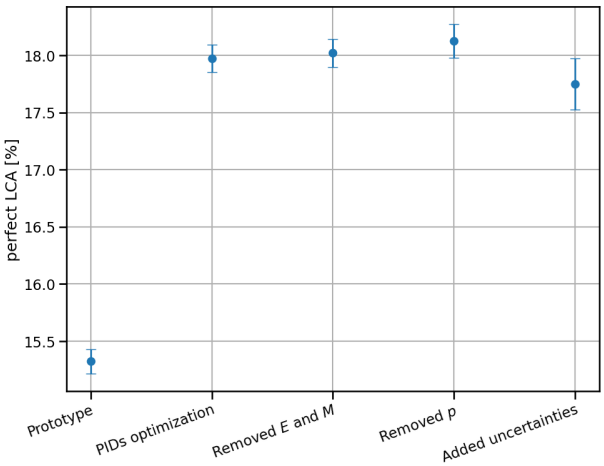


Figure 3.3: Recapitulating graph of the node features optimization showing the changes versus the average perfect LCAS in % on the validation set. The recapitulating graphs are done using the max between 45 and 50 epochs for the value and the standard deviation on the same epochs for the uncertainties.

After this optimization, only the PIDs, p_T, p_z, dr, dz, the charge q and the clusters information are kept as node features.

3.2.2 Edge features

Regarding the edge features, only one was used at the beginning: the cosine of the polar angle θ between two tracks. Two features to add were proposed: the Distance Of Closest Approach (doca) and the cosine of the azimuthal angle ϕ between the two tracks. The polar angle θ between the momentum $\mathbf{p_1}$ and $\mathbf{p_2}$ of two particles is defined as

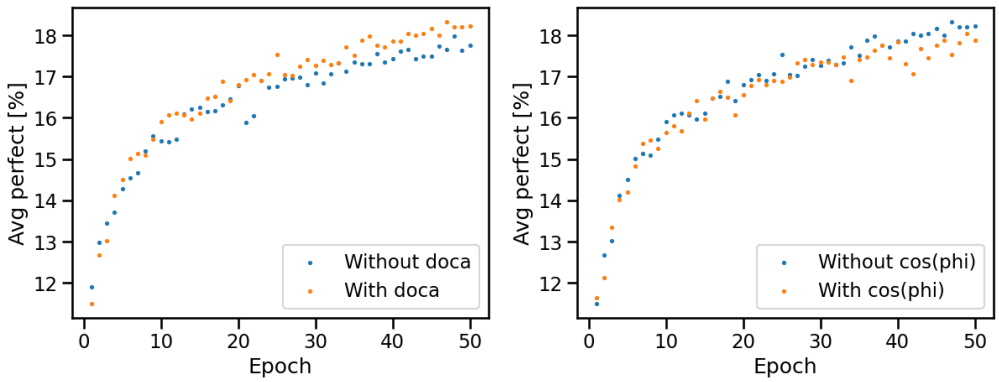
$$\cos(\theta) = \frac{\mathbf{p_1} \cdot \mathbf{p_2}}{||\mathbf{p_1}|| \times ||\mathbf{p_2}||} \tag{3.2}$$

Seemingly, the azimuthal angle ϕ is defined as

$$\cos(\phi) = \frac{p_{x1}p_{x2} + p_{y1}p_{y2}}{||\mathbf{p_{T1}}|| \times ||\mathbf{p_{T2}}||} \tag{3.3}$$

The two studies are shown in figure 3.4. While adding the doca improves the performances, figure 3.4(b), the addition doesn't increase the reconstruction efficiency. Furthermore, since those edge features were at first computed during the training, the training time increased.

To keep the pros of those new edge features without the longer training, the idea was to compute them directly when creating the training dataset, as it is done for the node features. This allowed us to use BASF2 and its functions, notably to add the uncertainties on $\cos(\theta)$, $\cos(\phi)$ and doca. Aside from a shorter training time, the addition of uncertainties on the edge features to the model does not improve the performance. The studies are recapitulated in figure 3.5.



(a) (b)

Figure 3.4: (a) Graph of the average perfect LCAS in validation with and without doca. (b) Graph of the average perfect LCAS in validation with and without $\cos(\phi)$.

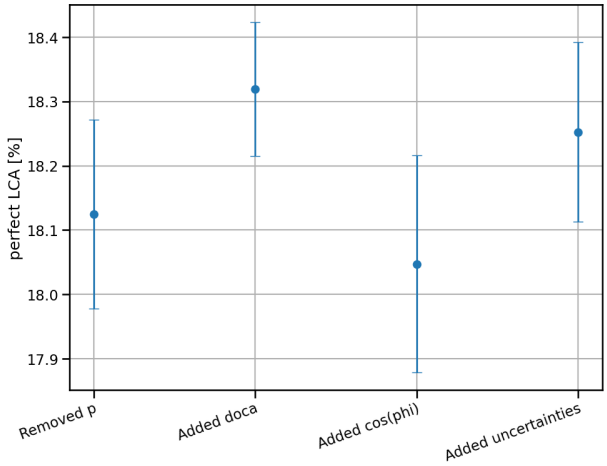


Figure 3.5: Recapitulating graph of the edge features optimization showing the changes versus the average perfect LCAS in % on the validation set.

3.2.3 Global features

The only global feature used until now was the total number of particles in the event. However, heuristically speaking, giving the number of particles should not have an impact on the performance of the model. Indeed, the number of FSPs could be deduced by the model from all the input variables. This way of thinking was verified when comparing the training with and without this feature, as shown in figure 3.6.

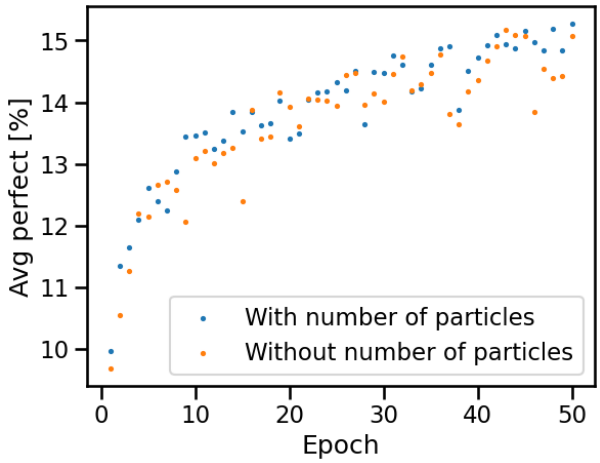
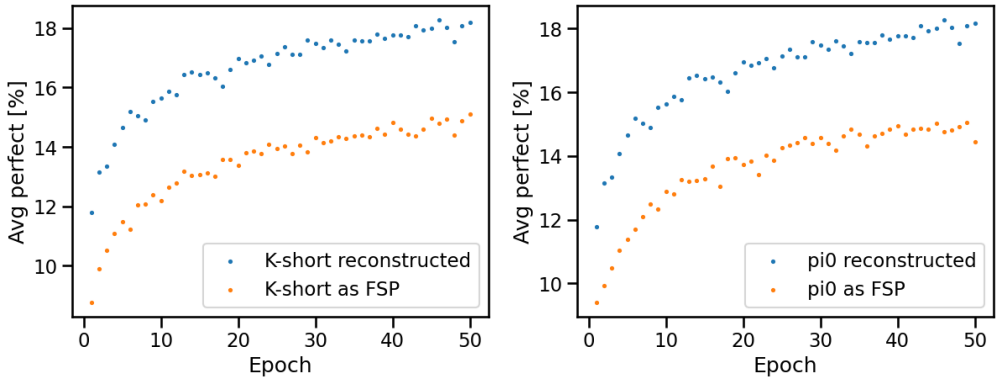


Figure 3.6: Graph of the average perfect LCAS in validation with and without the number of particles.

3.3 Class optimization

The classes used by the GRAFEI as of now for the LCAS definition are: 5 for B mesons, 4 for D^*, 3 for D, 2 for K_S^0, 1 for π^0 or J/ψ and 0 for particles not in the decay tree. This list of classes is inspired by the FEI stages as shown in figure 2.4 of section 2.3. While being based on something existing, the defined classes are still arbitrary and do not reflect a deeper physical meaning, since FEI and GRAFEI are fundamentally different. Another way of optimizing the algorithm was then to study and test different classes and arrangements of classes.

The first test done was about the K_S^0 and π_0. At this point, those two particles are reconstructed from their decay products during the training. To ease the task for the GRAFEI by making it reconstruct fewer particles, one can also do this reconstruction task when creating the training dataset. Additionally, this choice seemed to be harmless to the model from a physicist's point of view. The results, in figure 3.7, indicate worse performances for the model when reconstructing K_S^0 and π^0 in the training dataset rather than during the training.



(a) (b)

Figure 3.7: (a) Graph of the average perfect LCAS in validation with and without K_S^0 as Final State Particle (FSP). (b) Graph of the average perfect LCAS in validation with and without π^0 as FSP.

Another test was to move the J/ψ from its class to the same class as the D. The reason for this move is the same as for K_S^0 and π^0: to reduce the number of classes and ease the reconstruction for the model. The results of this test are shown in figure 3.8.

Since these results did not improve the performances, it was decided to discard them.

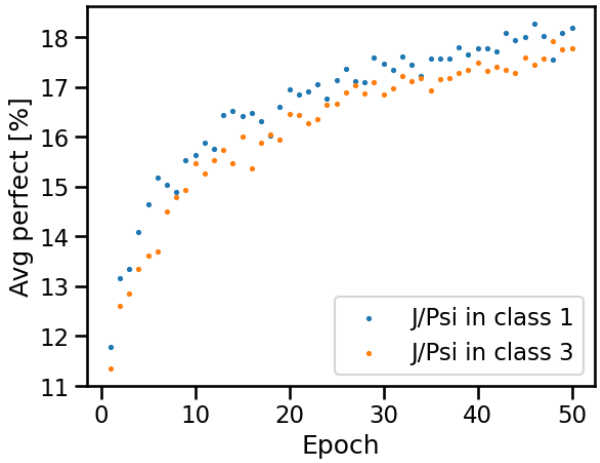


Figure 3.8: Graph of the average perfect LCAS in validation with J/ψ in the class 1 versus class 3.

3.4 Hyperparameters optimization

The list of hyperparameters is fixed, but one can choose which ones are important to be modified or not. Among this list, one may find the *learning rate*; *the batch size*, which is the number of events considered by the model for one step of the training; the *number of hidden layers* and their *dimension*; the number of GNNs between the input and output, named "*num_ML*" (*cf.* section 2.5); and the *dropout*.

To perform this task, the OPTUNA library was used [27]. This package automatically explores the predefined possibilities for the hyperparameters and gives at the end the best trial it ran relatively to a certain metric. In this case, the metric is the perfect LCAS.

Using the skip connections described in subsection 2.4.2, one could launch the hyperparameters optimization and make it explore solutions for the number of hidden layers and the number of GNNs, inaccessible otherwise because of the deep learning mentioned issues.

Hyperparameters	Minimum value	Maximum value	Mode
Learning rate	1×10^{-5}	1	LOGUNIFORM
Batch size	32	512	CATEGORICAL
Hidden layer's dimension	64	1024	CATEGORICAL
Number of hidden layers	1	3	INT
Number of GNNs	1	3	INT
Dropout	0	0.5	UNIFORM

Table 3.2: Table summarizing the hyperparameters explored by OPTUNA. "Mode" refers to how the set will be explored: UNIFORM and LOGUNIFORM are self-explanatory since they refer to the distributions of the same name; INT corresponds to a uniform distribution with integer step; CATEGORICAL means that the hyperparameters will be selected in a specified list of values. For "Batch size" and "Hidden layer's dimension", the values can only be powers of 2, meaning that the sets are $[32, 64, 128, 256, 512]$ and $[64, 128, 256, 512, 1024]$. More information about OPTUNA in reference [27].

Finally, one obtains a set of hyperparameters with the highest "OBJECTIVE VALUE". Unfortunately, because of an issue with OPTUNA we are currently trying to resolve, we are unable to give the set of parameters this optimization is associated to. However, the optimization lead to an improvement of $\approx 11\%$ of the average perfect LCAS, as shown in figure 3.9

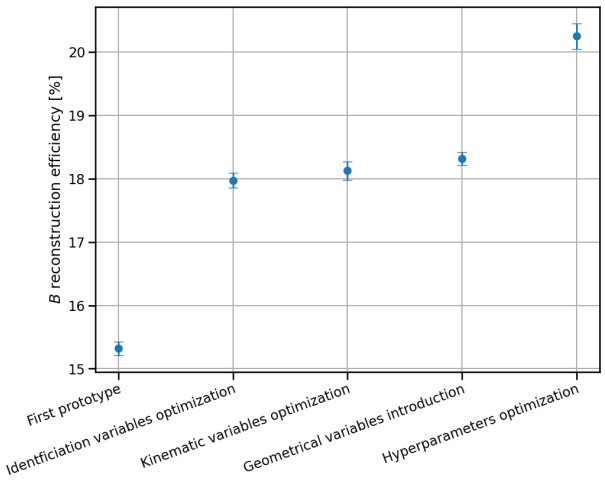


Figure 3.9: Recapitulating graph of the various optimizations showing the changes versus the average perfect LCAS in % on the validation set.

Chapter 4

Conclusion

The $B \to K\nu\bar{\nu}$ is a powerful probe for NP because of its rarity and the precision of its prediction in the SM. We presented a tool named graFEI conceived to reconstruct the partner B produced in the event, an essential ingredient for the study of the aforementioned decay. The graFEI proved to be much more efficient than the current algorithm used for reconstruction in Belle II, the FEI. The initial graFEI prototype already was two times more efficient for the reconstruction of B decays with respect to the FEI. The optimizations increased the reconstruction efficiency of the graFEI by 33%.

Various additional tasks are currently being investigated. We are currently trying to give more freedom to the model by using different geometries for the edge and node blocks (*cf.* section 2.5) and letting Optuna optimize those geometries. Additionally, new predictions by the model are being investigated, such as the prediction of the momentum of the reconstructed B. Lastly, one could for example study the impact of overtraining on the performance of the system. Indeed, the new hyperparameters use a dropout probability much smaller than the previous one, which may increase the overtraining of the model.

The work is being documented in an internal note for the Belle II collaboration. The ultimate goal is to use this algorithm on real data, which is expected to be five times what is currently used for the search for $B \to K\nu\bar{\nu}$.

Bibliography

[1] J. L. Hewett, *The Standard Model and Why We Believe It*, 1998, arXiv:hep-ph/9810316.

[2] G. Bertone and D. Hooper, *History of dark matter*, Reviews of Modern Physics **90**, 10.1103/revmodphys.90.045002 (2018), arXiv:1605.04909v2.

[3] A. I. Lonappan et al., *Bayesian evidences for dark energy models in light of current observational data*, Physical Review D **97**, 10.1103/physrevd.97.043524 (2018), arXiv:1707.00603.

[4] L. Canetti, M. Drewes, and M. Shaposhnikov, *Matter and antimatter in the universe*, New Journal of Physics **14**, 095012 (2012), arXiv:1204.4186.

[5] W. Parrott, C. Bouchard, and C. D. and, *Standard Model predictions for $B \to Kl^+l^-$, $B \to Kl_1^-l_2^+$ and $B \to K\nu\bar{\nu}$ using form factors from N_f=2+1+1 lattice QCD*, Physical Review D **107**, 10.1103/physrevd.107.014511 (2023).

[6] J. Grygier et al., *Search for $\boldsymbol{B \to h\nu\bar{\nu}}$ decays with semileptonic tagging at Belle*, Physical Review D **96** (2017), arXiv:1702.03224.

[7] T. Abe et al., *Belle II Technical Design Report*, 2010, arXiv:1011.0352.

[8] *Instantaneous luminosity record of SuperKEKB*, https://www.kek.jp/en/newsroom/2020/06/26/1400/.

[9] *Upsilon(4s) properties and decays*, https://pdg.lbl.gov/2014/listings/rpp2014-list-upsilon-4S.pdf.

[10] T. Keck et al., *The Full Event Interpretation*, Computing and Software for Big Science **3**, 6 (2019), arXiv:1807.08680.

[11] *Overview of the Belle II Detector*, https://www.kmi.nagoya-u.ac.jp/eng/blog/2019/03/11/belle2-phase3/.

[12] A. J. Buras, J. Girrbach-Noe, C. Niehoff, and D. M. Straub, *$B \to K^{(*)}\nu\bar{\nu}$ decays in the Standard Model and beyond*, 2014, arXiv:1409.4557.

[13] N. Jebreel et al., *Efficient Detection of Byzantine Attacks in Federated Learning Using Last Layer Biases*, in (Aug. 2020), pp. 154–165.

[14] R. Dixit et al., *Handwritten Digit Recognition using Machine and Deep Learning Algorithms*, International Journal of Computer Applications **176**, 27–33 (2020).

[15] B. Polyak, *Introduction to Optimization* (July 2020).

[16] L. Bottou and O. Bousquet, *The Tradeoffs of Large Scale Learning*, in Advances in neural information processing systems, Vol. 20, edited by J. Platt, D. Koller, Y. Singer, and S. Roweis (2007).

[17] *Overfitting*, https://en.wikipedia.org/wiki/Overfitting.

[18] N. Srivastava et al., *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research **15**, 1929–1958 (2014).

[19] S. Hochreiter et al., *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*, A Field Guide to Dynamical Recurrent Neural Networks (2003).

[20] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, 2015, arXiv:1512.03385.

[21] J. Kahn et al., *Learning Tree Structures from Leaves For Particle Decay Reconstruction*, en, Machine Learning: Science and Technology **3**, arXiv:2208.14924 [physics], 035012 (2022), arXiv:2208.14924.

[22] *Full Event Interpretation using Graph Neural Networks*, https://publish.etp.kit.edu/record/22115.

[23] P. W. Battaglia et al., *Relational inductive biases, deep learning, and graph networks*, 2018, arXiv:1806.01261.

[24] CROSSENTROPYLOSS *function in* PYTORCH, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

[25] T. Kuhr et al., *The Belle II Core Software*, Computing and Software for Big Science **3**, 10.1007/s41781-018-0017-9 (2018), arXiv:1809.04299.

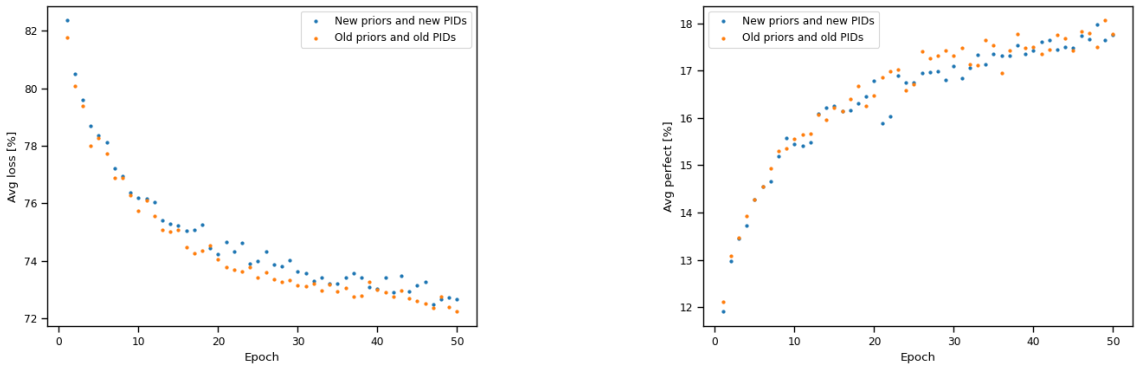
[26] *Description of the variables used in* BASF2, https://software.belle2.org/development/sphinx/analysis/doc/Variables.html.

[27] *The optuna package, used for hyperparameters optimization*, https://optuna.org/.

Appendices

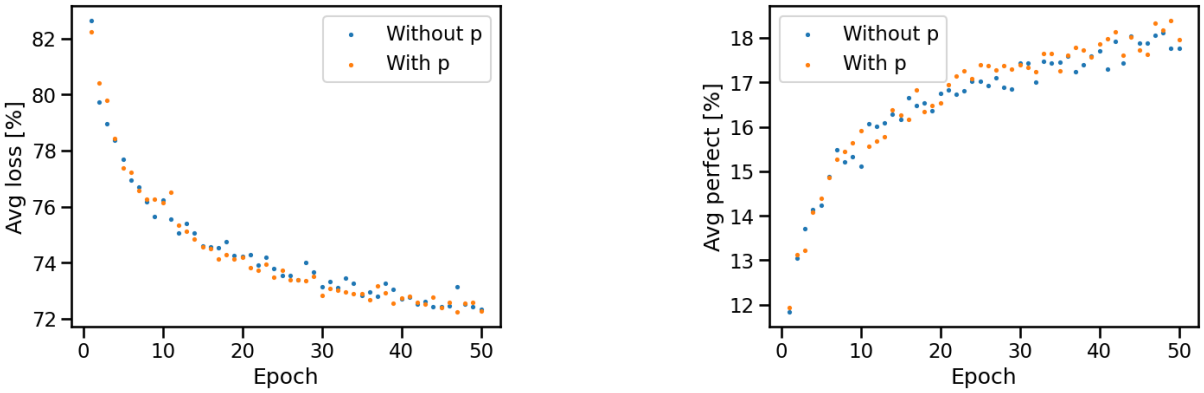
Appendix A

Additional results' graphs



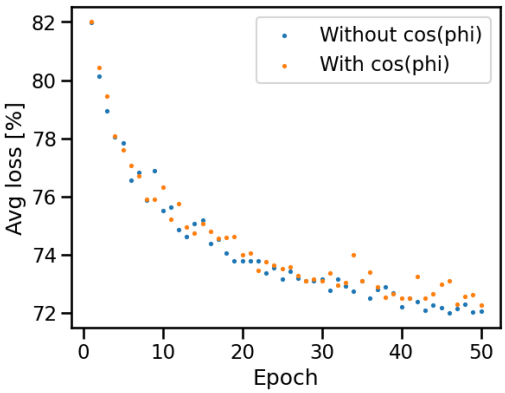
(a) (b)

Figure A.1: (a) Graph of the average loss in validation for old and new PIDs and priors. (b) Graph of the average perfect LCA in validation for old and new PIDs and priors.

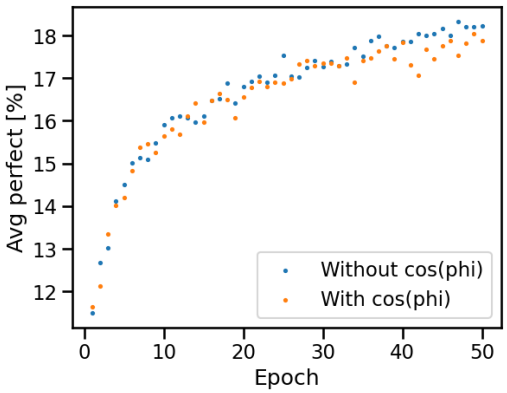


(a) (b)

Figure A.2: (a) Graph of the average loss in validation with and without p. (b) Graph of the average perfect LCA in validation with and without p.



(a)



(b)

Figure A.3: (a) Graph of the average loss in validation with and without $\cos(\phi)$. (b) Graph of the average perfect LCA in validation with and without $\cos(\phi)$.