TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF PHYSICS

Bachelor's Thesis

Implementing the TabNet Deep Learning Algorithm for Selecting $\tau^- \to \pi^- \pi^- \pi^+ \nu_\tau$ Events from Belle II Data

Author:	Yannik Fausch
1st Examiner:	Prof. Dr. Stephan Paul
2nd Examiner:	Prof. Dr. Sherry Suyu
Advisor:	Dr. Stefan Wallner
Submission Date:	23.08.2024

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF PHYSICS

Bachelor's Thesis

Implementierung des TabNet Deep Learning Algorythmus für die Selektion von $\tau^- \longrightarrow \pi^-\pi^-\pi^+\nu\tau$ Events von Belle II Daten

Author:	Yannik Fausch
1st Examiner:	Prof. Dr. Stephan Paul
2nd Examiner:	Prof. Dr. Sherry Suyu
Advisor:	Dr. Stefan Wallner
Submission Date:	23.08.2024

I confirm that this bachelor's thesis in physics is my own work and I have documented all sources and material used.

Munich, August 23, 2024 Yannik Fausch

Abstract

This thesis aims to compare TabNet, a Deep Learning (DL) algorithm for classification of tabular data, to the established Boosted Decision Tree (BDT) used for event selection of the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay. The goal is to see whether TabNet can outperform the established BDT. First the impact of hyperparameter values of TabNet were studied manually before conducting a high-dimensional automated hyperparameter optimization. The established BDT still yields a better performance over all purities than TabNet. Additionally, the feature importance of TabNet was studied and compared to the established BDT, to see the influence of features on to the prediction of these two models. For both models the most important category was the event-shape and the third most important were the vertex reconstruction features. The second and fourth features differ for TabNet and the established BDT. The second most important feature for TabNet were the γ and π^0 rejection features and the forth most important features were the kinematic features. For the established BDT it was the other way around.

Contents

Chapter 1

Introduction

The Standard Model (SM) contains three generations of leptons. The heaviest lepton is called τ lepton and has a mass of $1.777\,\mathrm{GeV/c^2}$. The decay time of the τ lepton decaying via the weak force is around 290 fs [17]. As it is the heaviest lepton it can decay into lighter leptons, i.e. electron (e) and muon (μ). It can also decay into light hadrons like Pion (π) and Kaon (K). The τ lepton allows to study Quantum Chromodynamics (QCD), the theory describing the strong interaction, under very clean conditions due to its semileptonic decays, especially in the low-energy regime. Additionally, τ leptons provide a chance for detecting New Phyiscs (NP). This includes the detection of the violation of the CP-symmetry, the question whether neutrinos are massless, if the lepton number is violated and testing lepton universality [24].

τ decays to multi-body hadronic final states can be used for spectroscopy of hadronic resonances, shedding light to the strong interaction and the properties of mesons. These meson resonances are short-lived states decaying via the strong interaction. Their characterization is essential to understand the strong interactions influence on quarks. For meson spectroscopy a Partial Wave Analysis (PWA) is used, allowing to extract properties such as the spin (J), parity (P), mass and width from the distribution of the decay products.

The PWA needs a clean sample of the signal decay. In this thesis, the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$[1] decay is studied. The pions in the signal decay are dominantly produced via an $a_1(1260)$ resonance. The shape, mass and width of this resonance are not well determined. Additionally, two more resonances could occure, namely the $a_1(1420)$ and $a_1(1640)$ resonance [26]. In order to study these resonances the signal events need to be selected from the Belle II data set including a variety of other decays, such as τ decays that do not have a 3×1 prong topology, where the number of prongs is equal to the number of charged particles the τ leptons decay into. The two most challenging backgrounds are $q\bar{q}$

[1] The charge conjugate is taken into account implicitly.

events originating from e^+e^- events as continuum background [12], followed by $\tau^+ \to \pi^-\pi^-\pi^+\pi^0\nu$ decays, which also have a 3×1 topology.

Before the PWA of the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay was conducted a BDT was used to select the signal event from the sample to get an as pure as possible sample. The sample consist out of tabular data. Therefore a BDT was prefered over a Neural Network (Neural Network), as they struggle with tabular data due to e.g. the presence of uninformative features [16]. Despite these struggles there are some Deep Learning (DL) algorithm approaches that seem to yield promising results. TabNet [4] is one of these approaches that was compared with other Machine Learning (ML) models such as BDTs and even performed better on e.g. the Higgs-Boson data set [4] by achieving higher accuracy scores.

The goal of this thesis is to apply TabNet on simulated Monte Carlo (MC) data from the Belle II experiment and see whether it can outperform the currently used established BDT for event selection of the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay. Additionally the importance of the input features of TabNet, which are the simulated variables, is studied and compared to the input feature importance of the established BDT, as both used the same input features for training.

The performance of TabNet is optimized by changing hyperparameter values manually to see their impact on the performance of TabNet. After this a high-dimensional automated hyperparameter optimization was conducted using the Optuna [2] framework. The obtained performance is compared to the established BDTs performance.
The feature importance is studied by using the Shapley Additive Explanations (SHAP) [27] framework, which calculates Shapley values to determine the importance of the used features. The Shapley values were calculated for the established BDT and TabNet making the feature importance comparable among these two models and gives insights into the decision-making of the models.

In chapter 2 the Belle II experiment alongside its seven subdetectors is presented. In chapter 3 BDT and TabNet are introduced alongside the preprocessing steps and the data set used in this analysis. In chapter 4 the evaluation methods are introduced in order compare the models studied in this analysis. In chapter 5 the performance optimization process of TabNet is discussed and the performance is compared to the established BDT performance. In chapter 6 the feature importance of TabNet is discussed and compared to the feature importance of the established BDT. In chapter 7 the results are concluded and an outlook for future analyses is given.

Chapter 2

The Belle II Experiment

The Belle II Experiment is located at the SuperKEKB accelerator in Tsukuba, Japan. It is an electron-positron collider experiment with the goal to make precise measurements of weak interaction parameters, study exotic hadrons and search for New Phyiscs (NP) [9]. The collision energy is around the $\Upsilon(4S)$ resonance peak. The collider has asymmetric beam energies of 7.0 GeV for electrons and 4.0 GeV for positrons [12]. Since June 2022 SuperKEKB has held the current world record for luminosity with a value of $4.7 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [9]. Despite this world record the target luminosity of $8.0 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ is yet to be reached [10]. During the run from 2019 to 2022 a data sample corresponding to an integrated luminosity of $400\,\mathrm{fb}^{-1}$ [9] was acquired and set to achieve a value of $50\,\mathrm{ab}^{-1}$ to get a higher statistical precision [10].
Since $\Upsilon(4S)$ practically only decays to $B\overline{B}$-meson pairs with a cross-section of 1.05 nb [18] the SuperKEKB accelerator is also referred to as B-factory. As the production of τ-lepton pairs has a similar cross-section of 0.92 nb [18], Belle II can also be called a τ-factory, enabling the analysis of τ-physics in decays such as $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$. Besides these two physic processes there are also others e.g. $q\overline{q}$, $\mu\overline{\mu}$ and Bhabar events.

In section 2.1 the Belle II detector and its sub-detectors will be explained.

2.1 The Belle II Detector

The Belle II detector has a cylindrical shape consisting of two endcaps and one main barrel region parallel to the beam axis. The "forwards"-direction is along the electron beam and the "backwards"-direction is along the positron beam. For the analysis of the received data from the detector a suitable coordinate system needs to be defined. The origin of the coordinate system is chosen at the Interaction Point (IP) of the two beams. The positive z-axis lies in the moving direction of the electron beam and the y-axis vertically upwards. Together with the x-axis this forms a right handed Cartesian coordinate system [20].

The Interaction Point (IP) is surrounded by an ultra-light, 2 cm diameter beryllium beam pipe [12]. Around the pipe there is the Belle II detector consisting of seven subdetectors in total, each one fulfilling a different task e.g. determination of the momenta of particles. All detectors and their arrangement are depicted in figure 2.3.

Closest to the IP are two layers of Pixelated Silicon Sensors (PXD) and four layers of Double-Sided Silicon Strip Sensors (SVD). The PXD layers are of the high resolution DEPFET type [3]. Their task is to measure decay vertex positions of short living particles like B mesons. Together, PXD and SVD are referred to as Vertex Detector (VXD).

The trajectories and energy loss (dE/dx) of charged particles are determined via a Central Drift Chamber (CDC) which is filled with helium and ethane in a ratio of 50:50. The three dimensional track of the particle passing through is reconstructable.

In the next layer the Time-of-Propagation (TOP) and an aerogel-based proximity focusing ring imaging Cherenkov system (ARICH) are located. Both use the Cherenkov effect to determine the particles passing through [25].

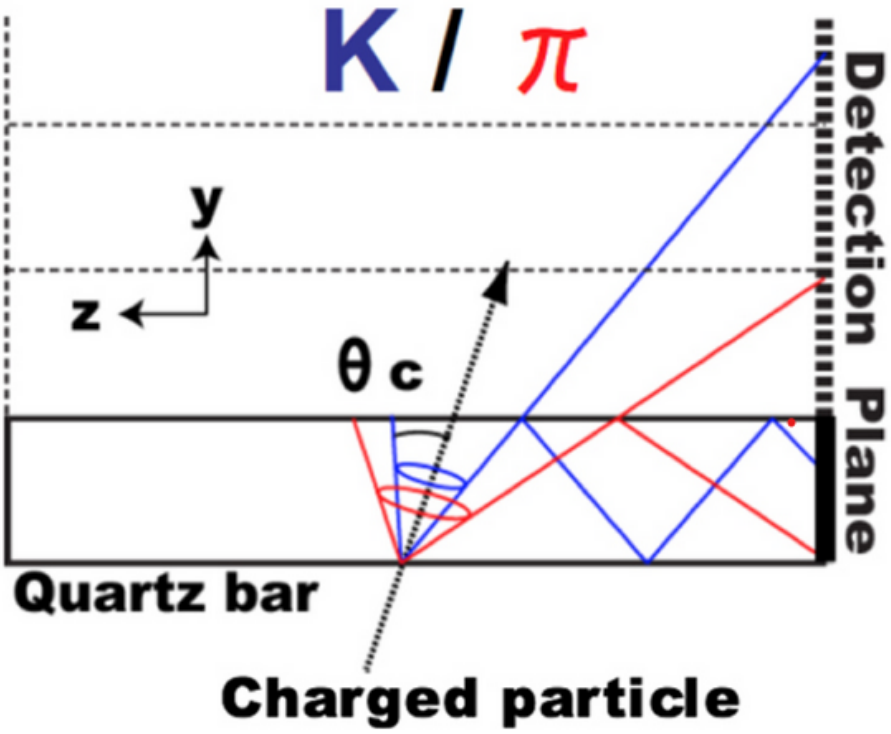


Figure 2.1: Scheme of a Kaon (K) and Pion (π) passing through a Time-of-Propagation (TOP) producing Cherenkov radiation. Taken from [14]

The TOP detector is located in the barrel region. Figure 2.1 shows a scheme of the TOP. It consists out of a 2 cm-thick quartz bars, in which the passing particle emits its characteristic Cherenkov radiation. Based on the angle in which the photons are emitted the arrival time on the end of the bar differs slightly for every particle due to total internal reflection. This allows to distinguish between different particles e.g.K and π, where π has a larger angle as it is lighter than K. At the end of the quartz bar Photomultiplier Tubes (PMTs) are placed to detect the photons.

Figure 2.2 shows a scheme of the ARICH located at the front endcap of the barrel. In this detector the traversing particle passes a total of 4 cm-thick aerogel. This aerogel consists of two different layers to have two different refraction indices $n_1 = 1.045$ and $n_2 = 1.055$. Each layer is 2 cm-thick. After the aerogel-layers the light propagates through an expansion volume, before hitting the photon detector and creating characteristic Cherenkov rings, where the radius depends on the particle [15].

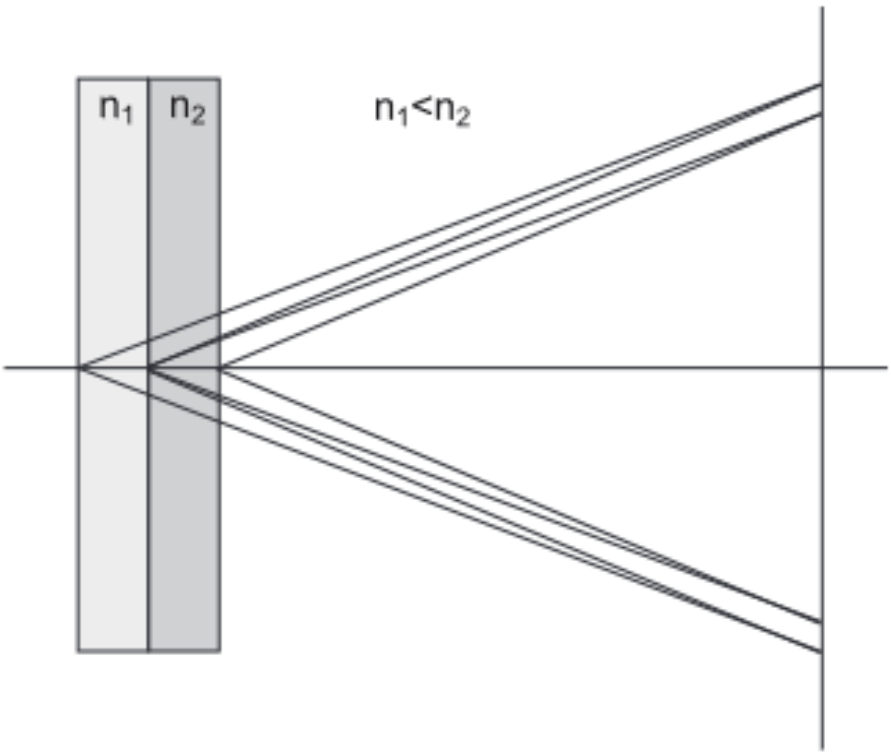


Figure 2.2: Scheme of the aerogel-based proximity focusing ring imaging Cherenkov system (ARICH). The two layers of aerogel are visible with their respective refraction indices ($n_1 = 1.045$ and $n_2 = 1.055$). The vertical bar displays the photon detectors. Taken from [12].

Electromagnetic Calorimeters (ECLs) are used to detect electrons and photons, which deposit almost all their energy in the ECL, via electromagnetic showers. Thereby the energy of the particle is measured. The ECL is located at the barrel and endcaps covering almost 90 % of the solid angle in the center-of-mass system.

Around the barrel ECL is a large-bore solenoid coil providing a 1.5 T magnetic field. It bends the trajectories of the charged particles. Thereby the momenta of charged particles are measured.

The last layer consists of the K_L^0 and Muon Detectors (KLM)s. As muons and kaons travel through the other calorimeters in the Belle II detector only these two types of particles are detected in this layer [12].

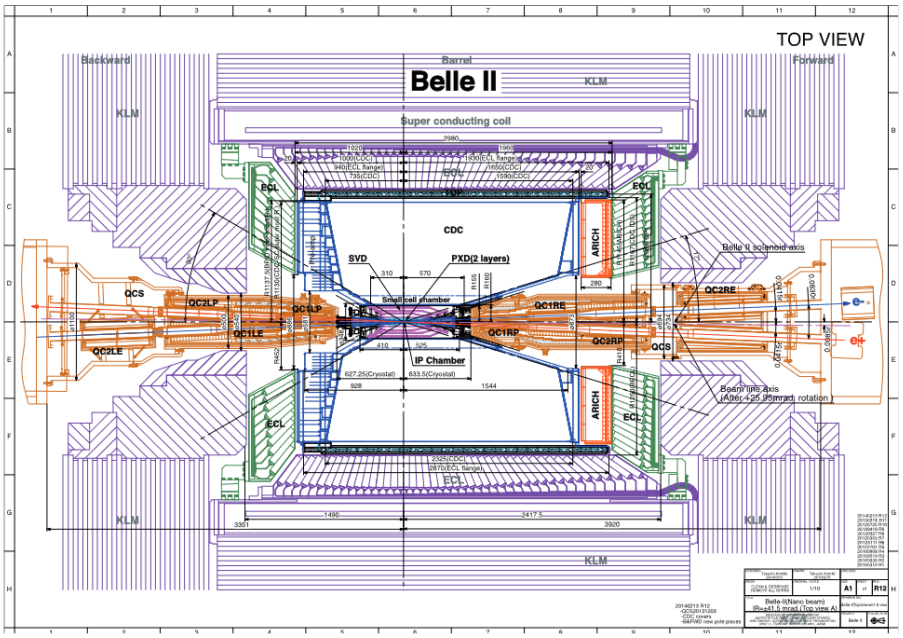


Figure 2.3: Top view of the Belle II detector. The purple and green detectors are the K_L^0 and Muon Detector (KLM) and the Electromagnetic Calorimeter (ECL). The orange detector is the aerogel-based proximity focusing ring imaging Cherenkov system (ARICH). The turquoise detector is the Time-of-Propagation (TOP). The blue detector is the Central Drift Chamber (CDC). The light purple detector is the Double-Sided Silicon Strip Sensors (SVD). The light blue detector is the Pixelated Silicon Sensors (PXD). Taken from [12]

Chapter 3

MVA Methods for Fineselection

Multivariate Analysis (MVA) uses two or more features at once in order to find patterns and correlations between them. This can be done by using different ML models. Model refers to the algorithm that learned by training on a specific data set to find patterns and make predictions by the patterns it has learned. This can include tasks like classification, where the model needs to distinguish between e.g. signal and background. In order to make precision physics measurements it is crucial to be able to distinguish between signal and background [5]. The process of separating signal from background is called event selection. The event selection scheme is divided into three steps. First the reconstruction is used to reconstruct the original particle from the measured data of the decay products. Second the preselection is applied, which uses the 3×1 topology of the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ event in order to distinguish it from background. Third the fineselection is applied to get a maximally pure signal event sample. In this step a MVA is used in the form of a BDT to get the maximum precision of the measured data as an event is characterized by multiple features of the data [5]. This thesis compares TabNet to the established BDT in this event selection step.

As DL algorithms are biased towards smooth functions the irregular patterns, in tabular data, are challenging [16]. Further more they are not robust against uninformative features often appearing in tabular data [16]. Apart from data structure the computation time needed for DL algorithms tends to be longer than for BDTs [16]. Also the vast amount of hyperparameters makes it challenging to find optimal solutions.
Still there are many attempts to make DL work on tabular data. One promising and in this thesis used attempt is called TabNet [4]. The authors of [4] compare several tree-based algorithms on several data sets, against the proposed TabNet. The most outstanding result comes from the Poker Hand data set [6], where TabNet outperforms tree-based models by almost 30% [4]. Making it

interesting for physics data applications is the comparison on the Higgs boson data set. Here the performance is not as outstanding as compared to the Poker Hand data set but still higher with an accuracy of 78.84% against 75.97% for the BDT [4]. Therefore TabNet is used in this thesis.

Section 3.1 presents the data set used in this thesis and the used features in the analysis. Section 3.2 introduces the BDT model. Section 3.3 presents the TabNet model, its architecture and the scaling of the data alongside with the training.

3.1 The Data Set

The data used for training, validating and testing the models analysed in this thesis is simulated Monte Carlo (MC) data from the Belle II MC. Also the established BDT was trained on this data set. A total of $4\times362\,\mathrm{fb}^{-1}$ generic MC data was produced, containing the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ signal and every background. The data set is divided into 32 equal chunks. In this thesis only chunk 31 was used with $15,481,952$ events. As signal the truth $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay was chosen according to the known MC truth and the rest was considered to be background. The background consisted of *taubBkg* which represent other τ decays not belonging to the signal excluding the $\pi^-\pi^-\pi^+\pi^0\nu$ decay as this also has a 3×1 topology and decays of $q\bar{q}$ events as the continuum background [12] the other channel are decays not originating from τ lepton decays. The total numbers of decays per channel are shown in table 3.1.

Table 3.1: Number of events alongside their fraction in the preselected data. Only *tau3pi* is considered as signal the rest is referred to as background.

Decay	**Events**	**Fraction**[%]
total	$15,481,952$	100%
$q\bar{q}$	$7,085,954$	46%
tau3pi	$3,841,829$	25%
$\pi^-\pi^-\pi^+\pi^0\nu$	$1,841,148$	12%
other	$1,392,989$	9%
tauBkg	$1,320,032$	8%

3.1.1 Utilized Features

A hand-made set of features including 30 features are used as input to TabNet in this thesis. The same features were used for the established BDT. The utilized features can be divided into four categories:

- **Event-shape:** features that characterise the events by their shape. The thrust is the only feature of this category used in this thesis.

- **Kinematics:** these features correspond to the energy and momenta of the particles in the events. This also includes missing momenta.
- **Vertex reconstruction:** feature of this category give information about the origin of the tracks.
- γ **and** π^0 **rejection:** features that suppresses background events with photons and π^0.

A list of all features divided into their subcategories is shown in section A.2.

Some features need more explanation as it is not clear from their label what they describe. The first one is *thrust*. It describes how co-linear the tracks of an event are aligned. It is defined as

$$T = \max_{\vec{n}} \frac{\sum_i |\vec{p_i}\vec{n}|}{\sum_i |\vec{p_i}||\vec{n}|}, \tag{3.1}$$

where $\vec{n}$ is called thrust axis. The value of T is in the interval of $[0, 1]$. When the value gets close to 1 the event is more jet-like and closer to 0 corresponds to a more spherical event.
Next are the features belonging to the list of reconstructed γ and π^0 rejection features. The π^0 decay into two photons that are measured and reconstructed to a π^0. Every photon that is not assigned to a π^0 is used under the following conditions. The first condition only includes photons with an energy larger than 200 MeV. The next cut is called *loosePhoton*. Here only photons with an energy larger than 100 MeV are included. The last cut is an MVA cut. This cut separates photons coming from physical processes, which we want to suppress from photons from background processes like photons coming from bumping into the beam pipe or are due to the interaction of the particles in the beam.
Most of the kinematic features are measured in the Center of Mass System (CMS) and are denoted with CMS.

3.2 Boosted Decision Trees

A Decision Tree (DT) is like a cut-based analysis, where thresholds in the d-dimensional feature space are chosen to separate signal from background. The input data gets split recursively based according to the cuts made on the features. Each split is made to maximize the information gain. This continues until some stopping condition is met. A schematic representation is shown in figure 3.1.

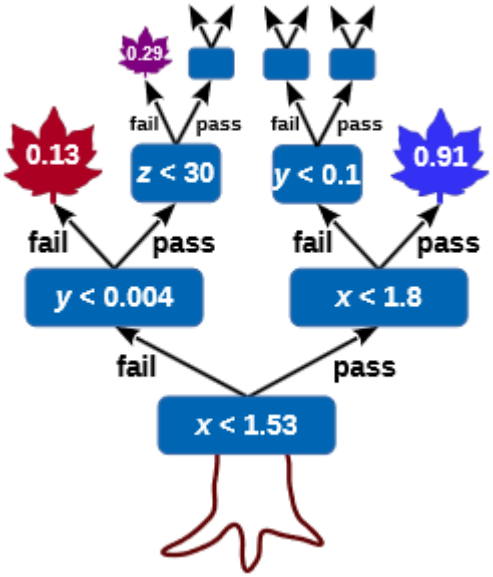


Figure 3.1: Schematic representation of a Boosted Decision Tree (BDT). The blue boxes depict the nodes and leaves represent the end nodes or decision. x, y, z represent the thresholds for the split of the data. Taken from [7].

The BDT is a combination of decision trees and a boosting algorithm like gradient boosting. A single DT is a weak learner, but if combined with many DTs becomes a strong classifier. After data is processed by a single DT, a new DT is created using gradient descent which designed to improve the previous trees. Again this procedure continues until a stopping condition is met [7].
In this thesis a BDT that was already trained on the full simulated MC data from the Belle II experiment (see section 3.1) and had optimized hyperparameter values is used as comparison to TabNet. This BDT is referred to as the "established BDT"[1].

3.3 TabNet

The architecture of TabNet is shown in figure 3.2 with multiple decision steps. Each decision step has the same structure. TabNet uses sequential attention [13] to select the input features used to make predictions. By this approach the network filters uninformative features and even makes it interpretable. A single step consists of an attentive transformer providing a mask to mask the input features followed by a feature transformer. Both will be explained in more detail in sections 3.3.1 and 3.3.2.

[1]The established BDT was developed by Dr. Stefan Wallner and Yingming Yang.

First the input goes through a Batch Normalization Layer (BN) before passing the first feature transformer, explained in section 3.3.1 and then entering the first step. In a single step the input passes the attentive transformer, explained in section 3.3.2, which provides the mask for the input features. Every mask is aggregated to provide interpretability at each step. Next is a feature transformer, after which the output is split into two. One part undergoes a ReLu[2] function and is subsequently added to the overall output. The other part gets feed to the next steps attentive transformer. This process is repeated n_{steps}[3] times. Finally the summed output of all decision steps is processed through a Fully Connected Layer (FC), which provides the final output of TabNet. In the analysis of this thesis of the simulated MC data from the Belle II experiment the final FC will be a two dimensional layer, as we are using binary classification in terms of signal and background events.
Additional information, in addition to the decision output is provided by the masks, which gives insights on information the computer used for its decisions in a single step. This is also referred to as local interpretability.

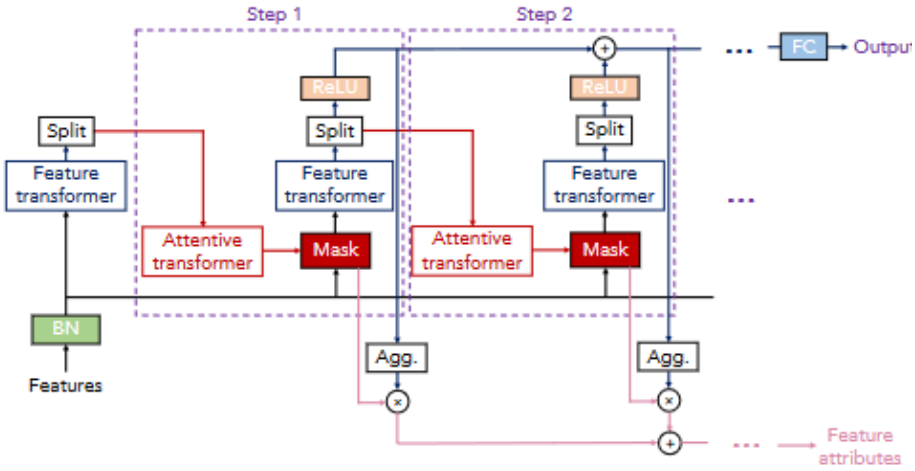


Figure 3.2: Architecture of TabNet. Two steps are depicted, but more can be added. Each step has the same structure. The most important parts are the feature and attentive transformers. Before the first feature transformer a Batch Normalization Layer (BN) is placed. The last layer providing the output is a Fully Connected Layer (FC). Taken from [4].

[2]Rectified Linear Unit (ReLU) is an activation function setting negative values to zero.
[3]a parameter representing the number of decision steps.

3.3.1 Feature Transformer

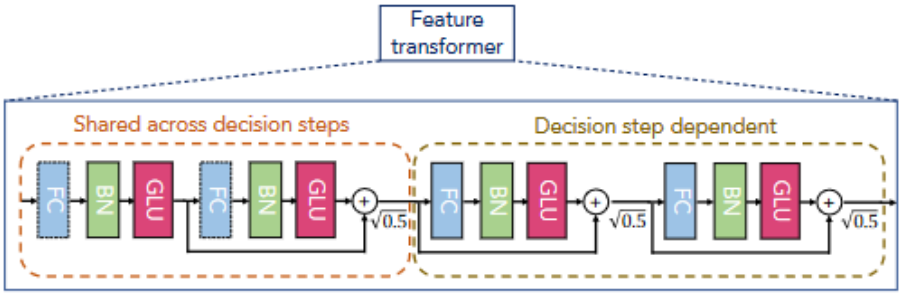


Figure 3.3: A feature transformer used in the TabNet architecture. One transformer block consists of a Fully Connected Layer (FC), Batch Normalization Layer (BN) and a Gated Linear Unit (GLU). Here two decision step dependent and two shared blocks are represented. After each block the result is added to the previous result, with a scaling factor of $\sqrt{0.5}$. Taken from [4].

In figure 3.3 the structure of a feature transformer is depicted, which consists of 4 transformer blocks divided into two types indicated by the dashed lines. One type is shared across the decision steps and the other one is dependent on the decision steps. Sharing across decision steps means, that the same weights achieved in earlier steps are taken into account, while the independent transformer block starts from scratch. A single transformer block is composed of a Fully Connected Layer, a Batch Normalization Layer and a Gated Linear Unit. The GLU is defined as

$$GLU(x) = \sigma(x) \cdot x, \tag{3.2}$$

where $\sigma(x)$ is a Sigmoid function[4].
Every block's result is added to the previous result, with a scaling factor of $\sqrt{0.5}$. The scaling factor is introduced to help stabilize the learning by ensuring that the variance does not change drastically throughout the network [4]. The feature transformer takes input of dimension $n_{features}$ and outputs information of dimension $n_d + n_a$, which are both hyperparameters, that will be explained in section 3.3.3. $n_{features}$ is the dimension of the features i.e. how many features are passed to the network.

[4]Here $\sigma(x) = \frac{1}{1+e^{-x}}$ is used.

3.3.2 Attentive Transformer

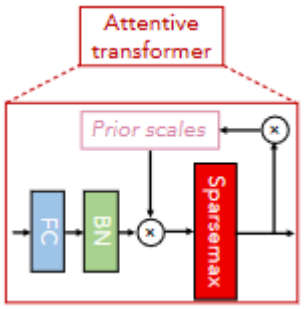


Figure 3.4: An attentive transformer used in the TabNet architecture. It starts with an FC followed by a BN. Before inputting the features to the Sparsemax function they are scaled (Prior scales). The output of the Sparsemax function is again used to scale the next Sparsemax input. Taken from [4].

Figure 3.4 depicts the structure of an attentive transformer. Again the first two layers are FC and BN layers. The reusage of features is controlled by scaling them with 'Prior Scales'. It rescales the features before entering the Sparsemax function according to

$$P_i = \Pi_{j=1}^{i}(\gamma - M_j), \quad with \quad P_0 = 1 \tag{3.3}$$

where M_j is the previous mask and γ is the relaxation factor[5]. Prior scaling is therefore performed based on the knowledge of what is known about a feature and how often it was used in the previous steps.
In order to obtain the masks the scaled outputs are fed into a Sparsemax function. A Sparsemax function is a sparser version of the Softmax function [21]. It sets features with negative values to zero and is responsible for the instance-wise feature selection. The attentive transformers input dimension is n_a and its output dimension is $n_{features}$.

3.3.3 Hyperparameters of TabNet

Here a list of the used hyperparameters is presented [11], their default values are presented in table A.2 alongside their recommended range according to the authors of [4] and [11]. The default values will be referred to as model default hyperparameter values. There are a few parameters left out as they are not relevant in this application, e.g. three parameters for categorical features, which are not of interest as there are no categorical features in the Belle II data. Left out hyperparameters are listed in table A.1. During this thesis the hyperparameter values are constantly adjusted. During the manual hyperparameter optimization (see section 5.1) the model default hyperparameter values are used for the

[5] a parameter controlling the reusage of features

first model. Afterwards the hyperparameter values are changed one by one.

The hyperparameters can be split into two categories: model parameters and fit parameters.
First we will take a look at the model parameters, which impact the structure of the model.

- n_d : parameter for the dimension of the decision prediction layer. The documentation states that this parameter is a risk for potential overfitting[11].
- n_a : dimension of the attention masks embedding. It is recommended to set this to the value of n_d. Therefore both features are referred to as $n\_da \equiv n_a = n_d$.
- n_{steps} : number of decision steps.
- *gamma* : coefficient for feature reusage in the masks. If set to 1 each feature will be used in only a single decision step. Larger values allow features to be used in more than one decision step.

Next are the fit parameters, which impact the training of the model, but are not directly part of the model.

- *lambda_sparse* : sparsity coefficient for feature selection. The bigger the coefficient the sparser the model will be.
- *learning_rate* : parameter used for the Adam[6] optimizer, used in the analysis. The learning rate corresponds to the magnitude of change performed on the parameters, based on the computed error.
- *gamma_scheduler* : this hyperparameter is connected to the scheduler used for adjusting the *learning_rate* of the network. Here an exponential scheduler[7] is used, which decays the learning rate by *gamma_scheduler* per epoch.
- *max_epochs* : maximum number of epochs TabNet is trained.
- *patience* : early stopping interrupts the training when for *patience* epochs a given metric does not change. Here the validation Area Under the Curve (AUC) is used for early stopping.
- *batch_size*: size of the batches. A large *batch_size* of around $1 - 10\%$ of the sample size is recommended.
- *virtual_batch_size* : size of batches used for the ghost batch normalization, which normalizes batches of the size of *virtual_batch_size* to mimics training with small *batch_size* [19]. *batch_size* should be a multiple of this value.

[6] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[7] $learning\_rate(epoch) = gamma\_scheduler \times learning\_rate(epoch - 1)$

3.4 Scaling the Data and Training of TabNet

Before TabNet is trained and used to make predictions the input data needs to be preprocessed. This includes scaling the data and handling missing values. In the data set used in this analysis there are no missing values, reducing the prepocessing steps to scaling the data. Without scaled data the problem gets more complex, as each feature may be on a different scale i.e. kilograms or hours. The error gradient may have large values and therefore the weight values may change a lot which makes the learning process unstable [23]. As TabNet was trained with a specific scaler the scaler needed to be saved as it is part of TabNet. In this thesis the initial training sample was used to learn the scalar parameters. After scaling the data the hyperparameter values are selected and subsequently the model is initiated and fit with the scaled data for *max_epoch* epochs. This only holds if $patience = 0$.
TabNet uses BN to normalize the data before entering the first feature transformer (see section 3.3). Therefore the authors of [4] state that TabNet can handle raw data making the scaling unnecessary. Whether this holds true is studied in section 5.1.1.
The goal of training a neural network is to adjust the weights of the activation function of the network. In order to do so a loss function is used to determine the difference between the predicted label and the actual label (see section 4.1). During the training the learning rate tells the network how much the weights need to be changed and the scheduler adjusts the learning rate according to the scheduler after each epoch. The network then tries to find a minimum for the loss function to maximize its accuracy.
Section 3.4.1 explains the StandardScaler and section 3.4.2 explains the MinMaxScaler. Both scalers are part of the sklearn.preprocessing library [22].

3.4.1 The StandardScaler

The StandardScaler standardizes the data according to

$$z = \frac{x - u}{s}, \tag{3.4}$$

where x is the sample, u is the mean of the sample and s is the standard deviation of the sample. This transformation sets the mean of every standardized feature z to 0 and the standard deviation to 1.

3.4.2 The MinMaxScaler

The second scaler considered in this thesis is the MinMaxScaler. It scales each feature to be in the range [0,1]. This happens according to

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{3.5}$$

where x_{min} is the minimum value in the data and x_{min} is the maximum value in the data. Each individual value is then scaled by

$$x_{scaled} = x_{std}(max - min) + min, \tag{3.6}$$

where max is the maximum and min is the minimum of the given range.

Chapter 4

Evaluation Methods

The training of TabNet models is evaluated using the log-loss and the intrinsic sparse-loss of TabNet. The log-loss and sparse-loss of the validation sample is used to check for overfitting during training and consequently if the training was successful. The performance of a model is the measurement of how accurate the prediction of the model is on new data, the better a model predicts new data the better its performance. In order to compare the performance of two models the Receiver Operating Characteristic (ROC)-curve and the AUC are used. Besides the performance the feature importance can also be measured. For this the Shapley values originating from cooperative game theory are used.

In section 4.1 the logloss is introduced and the intrinsic sparse-loss of TabNet. In section 4.2 the ROC-curve is introduced used for performance analysis. In section 4.3 Shapley values are introduced use for feature importance analysis.

4.1 The Log-loss

TabNet combines two loss metrics the log-loss and the sparsity-loss according to

$$L = L_{pred} + \lambda_{sparse} L_{sparse}. \tag{4.1}$$

The first loss metric is the log-loss. It reflects the discrepancy between the prediction probability of a model and the actual label. The log-loss is defined as

$$L_{pred} = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i) \log(1 - p_i), \tag{4.2}$$

where N is the sample size, y_i is the true label $(0, 1)$ and p_i is the predicted probability. This means that a lower log-loss corresponds to a better prediction and consequently performance. [8].

The second loss metric to punish its model during training is referred to as

sparsity-loss as it is used to regulate and favour the use of sparse feature masks [4]. It is adjustable by tuning *lambda_sparse*, introduced in section 3.3.3. The sparsity-loss is defined as

$$L_{sprase} = \sum_{i=1}^{N_{steps}} \sum_{b=1}^{B} \sum_{j=1}^{D} -\frac{M_{bj}[i] log(M_{bj}[i] + \epsilon)}{N_{steps} B}, \tag{4.3}$$

where N_{steps} corresponds to n_{steps}, B is the *batch_size*, D the amount of features used, M_{bj} the mask of feature j in batch b and ϵ a constant for numerical stability, also a hyperparameter, which should not be changed.
From here on the final loss will be referred to as logloss, as TabNet also utilizes this labeling. An example of a desired logloss as function over the epochs is shown in figure 4.1.

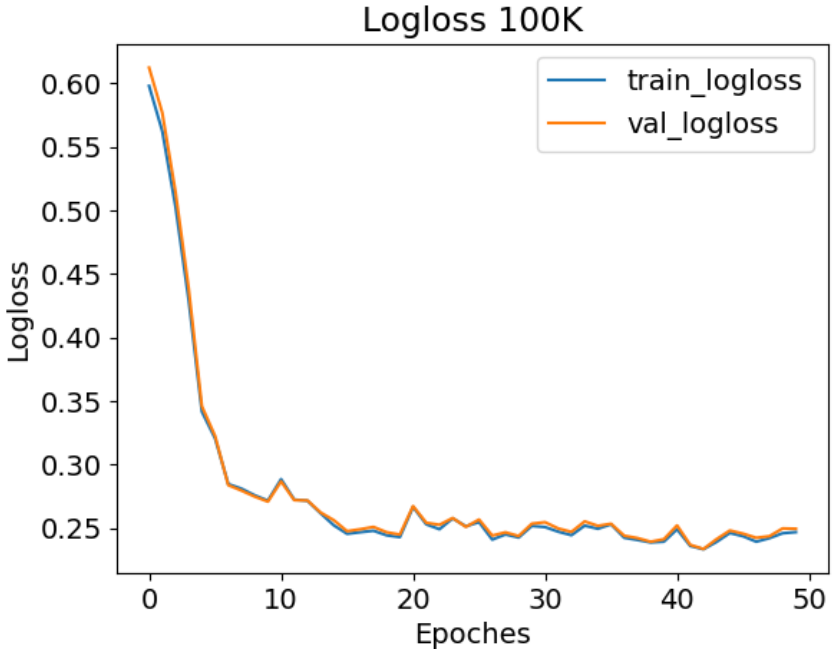


Figure 4.1: Logloss of a TabNet model trained on 100K events for 50 epochs and model default hyperparameter values. The blue curve represents the logloss evaluated on the training sample of each epoch (*train_logloss*). The orange curve represents the logloss evaluated on the validation sample of each epoch(*val_logloss*).

The logloss decreases exponentially. This means that within the first epochs the logloss decreases rapidly and afterwards converges slowly towards a static plateau. If the validation logloss increases while the trainings logloss decreases

the model is overfitting. When the model is overfitting it memorizes the training data and therefore generalizes poorly, which is leading to a bad performance and is therefore not desirable. Also, if both loglosses are increasing overfitting is implicated. Spikes within the logloss are normal as long as it decreases afterwards, since the model exits a local minimum in the prediction function in order to find a more global one. The models training is therefore considered successful when the logloss decreases exponentially and reaches a static plateau without overfitting.

4.2 The ROC-Curve

In order to compare the performance of the established BDT with the TabNet model a Receiver Operating Characteristic (ROC)-curve is used. It depicts the True-Positive-Rate (TPR) plotted against the False-Positive-Rate (FPR), for every threshold. TPR is defined as

$$TPR = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.4}$$

and describes the rate of signal events being identified as signal. FPR is defined as

$$FPR = \frac{FalsePositive}{TrueNegative + FalsePositive}. \tag{4.5}$$

and describes the rate of background events being identified as signal.

A model predicts probabilities, returning values in the range of [0,1]. Therefore a certain threshold is set to determine which values are accepted as signal and which as background. Each threshold is used to calculate TPR and FPR and therefore the ROC-curve. An example is shown in figure 4.2. The closer the curve gets to the upper left corner the better the selection performance of the model [29]. If the model is just guessing the ROC-curve would be described by $f(x) = x$ i.e. a diagonal line.

The ROC-curve can also be given in terms of Purity and Efficiency. Efficiency is the fraction of signal events correctly classified, it is exactly the same as the TPR. Purity, on the other hand differs from the FPR as it describes the correctly classified signal events in the total selected sample.

$$Purity = \frac{selected\_signal}{selected\_signal + selected\_background}. \tag{4.6}$$

In that case the performance is better the closer the curve gets to the upper right corner. An example is shown in figure 4.3. When looking at the ROC-curve in terms of efficiency and purity a good performing model has a high efficiency over a long range of purities.

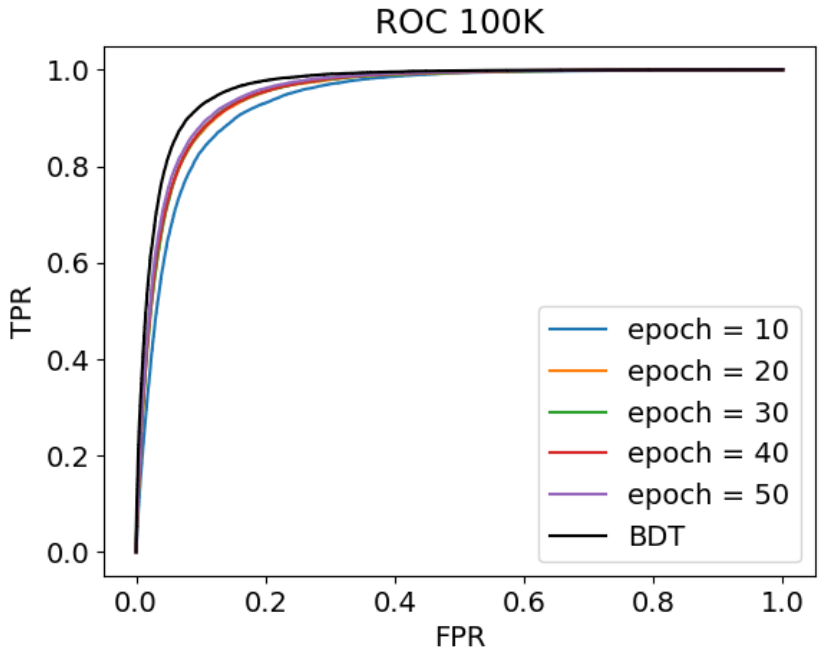


Figure 4.2: Receiver Operating Characteristic (ROC)-curve of TabNet model trained on 100K events for a total of 50 epochs and the model default hyperparameter values in terms of True-Positive-Rate (TPR) and False-Positive-Rate (FPR). After every 10 epochs the TabNet model was saved to see the performance improvements over the epochs. The ROC-curve of the established BDT is shown as reference (black curve).

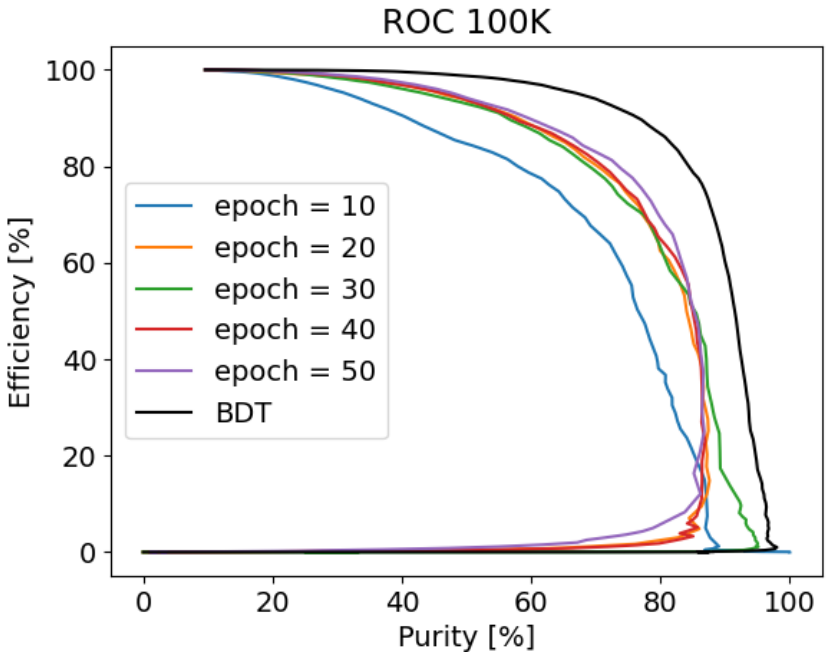


Figure 4.3: Receiver Operating Characteristic (ROC)-curve of TabNet model trained on 100K events for a total of 50 epochs and the model default hyper-parameter values in terms of Efficiency and Purity. After every 10 epochs the TabNet model was saved to see the performance improvements over the epochs. The ROC-curve of the established BDT is shown as reference (black curve).

A metric that can be calculated from the ROC-curve is the AUC, which is the area under the ROC-curve. It is between 0 and 1 and the model is performing better the closer its AUC is to 1.

4.3 The Shapley Value

Understanding how a neural network makes its decisions is still a major challenge in the field of AI. As explainability is imperative there are many approaches to make them more interpretable. One approach is to calculate the Shapley values, which are derived from cooperative game theory and try to explain the decision making of machine learning models by explaining the prediction as a sum of the contribution of each feature. This happens on an event by-event basis enabling the interpretation of single events.
In theory the model is presented a subset of the test sample, where the feature of interest is excluded and makes a prediction. Then the model is presented the full test sample including the feature of interest and makes a prediction. The difference between these two prediction values is the Shapley value. This

is mathematically describe by

$$\Phi_i = \sum_{S \subseteq N \setminus i} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup i) - f_x(S)], \tag{4.7}$$

where S is a possible subset, $|S|!$ is the number of permutations of all feature values before the i-th feature value, M the number of features and f_x the prediction function i.e. TabNet or the BDT output in this case [28].
As this is computationally expensive approximations are considered in order to calculate Shapley values. Here the SHAP framework was used [27] more specifically the KernelExplainer. In the case of the KernelExplainer a linear surrogate model is used implying that the model of interest is at least locally linear. Additionally the features are assumed to be independent of each other.
As models often can not handle missing data, the algorithm takes a background data set[1] and replaces the feature value of interest with a random feature value of the given background data set. In order to be able to compare two models the background data set needs to be the same for both and hence needs to be saved. Then linear regression is used to approximate the relation between the input features and the model prediction of each perturbation. The coefficients of the linear regression are the Shapley values.[1]
In the case of binary classification a positive Shapley value means that for the given event the feature is pushing the prediction towards positive labels (1) and a negative Shapley value means that the feature is pushing the prediction towards negative labels (0).
The feature importance of a feature is given by the mean of the absolute Shapley values of the test sample.

[1] this is a subset of the training sample

Chapter 5

Performance Analysis

The thesis evaluates whether TabNet can outperform the established BDT on event selection for the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay. Therefore TabNet is first tuned by manually adjusting the hyperparameter values to get an idea of how certain hyperparameter affect TabNet, before an automated hyperparameter optimization framework, namely Optuna [2], is used to find the optimal settings for the hyperparameter.[1]

In this chapter the performance of TabNet, on the Belle II MC $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ simulated data (see section 3.1) is discussed. Section 5.1 discusses the process of manually optimizing selected hyperparameter of TabNet. It starts by discussing the selection of a scaler. Next the *batch_size* and sample size are investigated. Last the risk of overfitting with the *n_da* hyperparameter is discussed. Section 5.2 discusses the automated hyperparameter optimization and the correlation between the hyperparameters. Last the performance of TabNet, with the optimal choice of hyperparameter values, is compared to the performance of the established BDT, in order to determine whether TabNet performs better in event selection than the established BDT.

5.1 Manual TabNet Hyperparameter Optimization

Before showing the results of the manual hyperparameter optimization it is important to establish some ground rules about the labeling of the plots. The sample size is given in terms of the total sample size i.e. 1M means $800,000$ training, $100,000\%$ validation and $100,000\%$ testing sample size as we use a 80/20 split for the data. Each plot consists of a title, where the type of plot, used sample size, compared hyperparameter and epoch, if there is no epoch split, are declared. In the legend the hyperparameter value of interest is displayed.

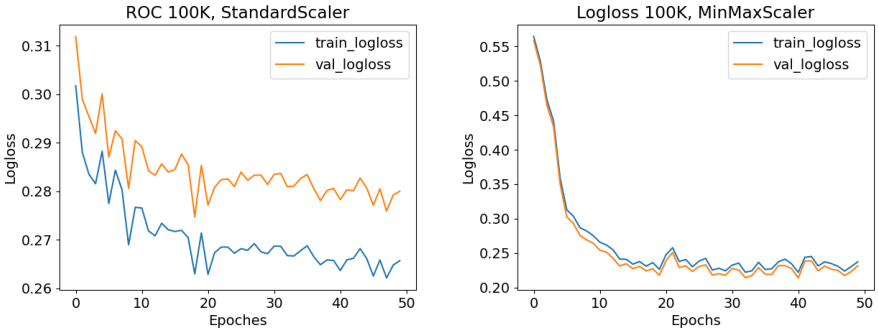
[1] During the initial approach the scaler, which is an essential part of the neural network was not saved. This was fixed in the presented results

After each analysis the best performing model from the previous analysis is used to compare with the new results.

5.1.1 Scaler Optimization

TabNet requires to define a suiting preprocessing scaler,explained in section 3.4, for the data. Additionally a NoScaler model, which trained on not preprocessed data, is used, as the first layer of the TabNet model is a BN and the authors of [4] state that this makes the application of a preprocessing step redundant.

In figure 5.1 the logloss of the StandardScaler and the MinMaxScaler model is shown. Both models trained on 100K events, a total of 50 epochs and the model default hyperparameter values. In both models the logloss decreases exponentially. While the validation logloss and the training logloss of the MinMaxScaler model follow almost the same line. This validates that the training of the MinMaxScaler model was successfull. In comparison the validation logloss and the training logloss of the StandardScaler model do not follow the same line. While both loglosses decrease, the validation logloss stays above the training logloss, which is already a sign that the model is overfitting. When the model overfitts it memorizes the training data and therefor generalizes poorly. In figure A.1 the logloss of the NoScaler model is shown, it shows the same behaviour as the logloss of the MinMaxScaler and therefore also indicates the success of the training.

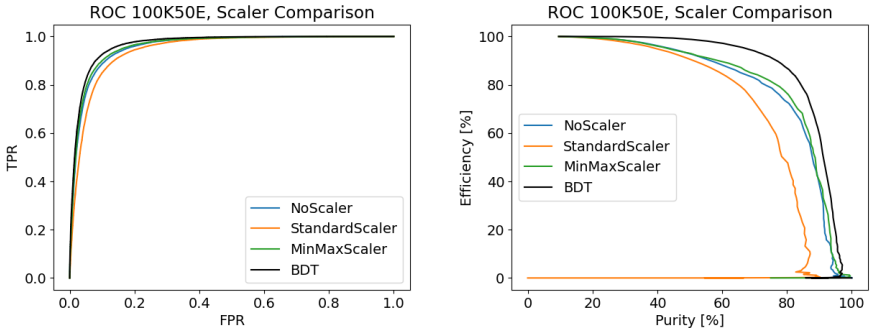


(a) (b)

Figure 5.1: Logloss as in figure 4.1. The data was scaled using the (a) StandardScaler, (b) MinMaxScaler

In figure 5.2 the performance of all three models is shown alongside the established BDT performance as reference. The TabNet models were trained using 100K events and 50 epochs. Figure 5.2a shows the ROC-curve in terms of TPR and FPR, while figure 5.2b shows the ROC-curve in terms of Efficiency and Purity. In both cases the StandardScaler model has the worst performance, while the MinMaxScaler and NoScaler model show almost the same performance,

with the MinMaxScaler showing a small improvement. Therefore the data is preprocessed using the MinMaxScaler for every following analysis.

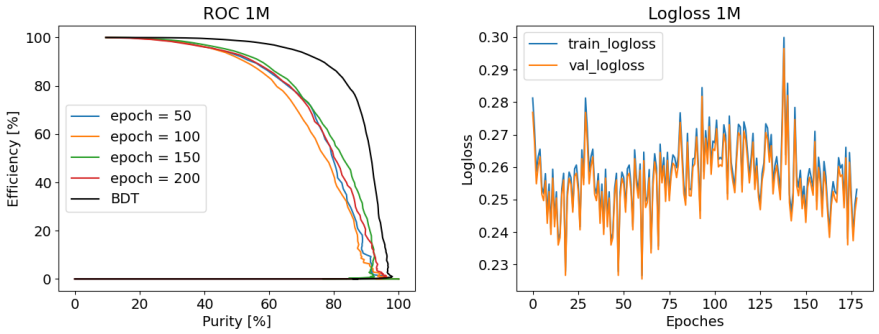


(a) (b)

Figure 5.2: ROC-curve as in figure (a) 4.2, (b) 4.3. The models are scaled using no scaler (blue), StandardScaler (orange) and MinMaxScaler (green).

5.1.2 Batch and Sample Size Optimization

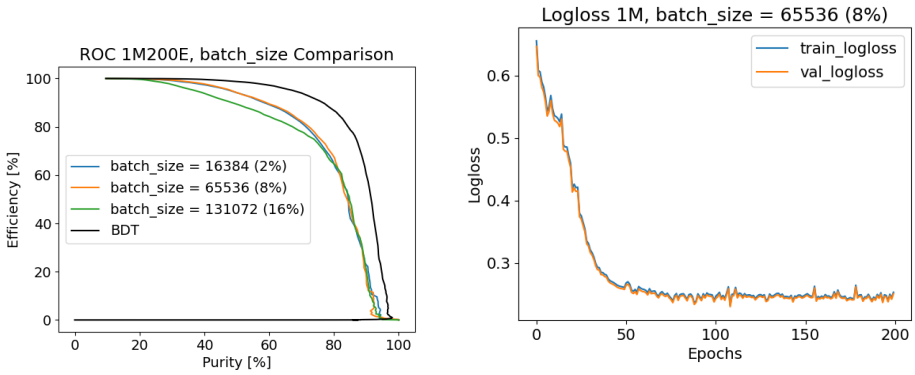
From experience a lot of data often helps to increase performance of deep neural networks. This is why here two different sample sizes are discussed. In figure 5.3b the logloss as a function of the epoch of a TabNet model trained on 1M events for 200 epochs is shown. The logloss does not converge to lower values and the training was unsuccessful. Figure 5.3, which shows the performance after every 50 epochs of training of the model trained on 1M data, supports this hypothesis, as the performance fluctuates around the curve of the model after 50 epochs. A steady increase or decrease of performance is expected.



(a) (b)

Figure 5.3: (a)ROC-curve as in figure 4.3 trained on 1M events, a total of 200 epochs and saved after every 50 epochs. (b) Logloss as in figure 4.1. The model trained on 1M events and 200 epochs.

Up until this point the model default hyperparameter values were used. The authors of [4] state that a high *batch_size* in the range of $1-10\%$ of the sample size, is recommended in order to achieve desired results. Compared to 1M events the model default *batch_size* is low. Therefore three models were trained using 2.0%(16348), 8.2%(65538) and 16.4%(131072). The *virtual_batch_size* was set to *batch_size* divided by 32. In figure 5.4b and A.2 the loglosses of these models are shown. Each logloos looks better than the one shown in figure 5.3b, as they converge to lower values.



(a) (b)

Figure 5.4: (a)ROC-curve as in figure 4.3 trained on 1M events, a total of 200 epochs and *batch_size* = 16384 (blue), 65536 (orange), 131072 (green). (b) Logloss as in figure 4.1. The model trained on 1M events, 200 epochs and *batch_size* = 65536.

From these plots it is not safe to say which model is performing the best. There-

fore the ROC-curves are shown in figure 5.4a. All trained TabNet models show a similar result. The model trained with a *batch_size* of 65536 consistently outperforms the other models. The model trained with the largest *batch_size* underperforms for purities below 80%. Afterwards it yields almost the same performance as the other trainings.
In conclusion a large *batch_size* between $2 - 8\%$ is favourable for an optimal performance, but it should not exceed this range, as the model trained with a *batch_size* of 131072 shows.

Now the question remains whether a larger sample size leads to an increase in performance. Therefore models trained with 100K and 1M events are compared, both are trained for 200 epochs. Figure 5.5 shows the performance of the model trained with the model default hyperparameter values and 100K events compared to the model trained with 1M events and a *batch_size* of 65536, while every other hyperparameter remains unchanged.

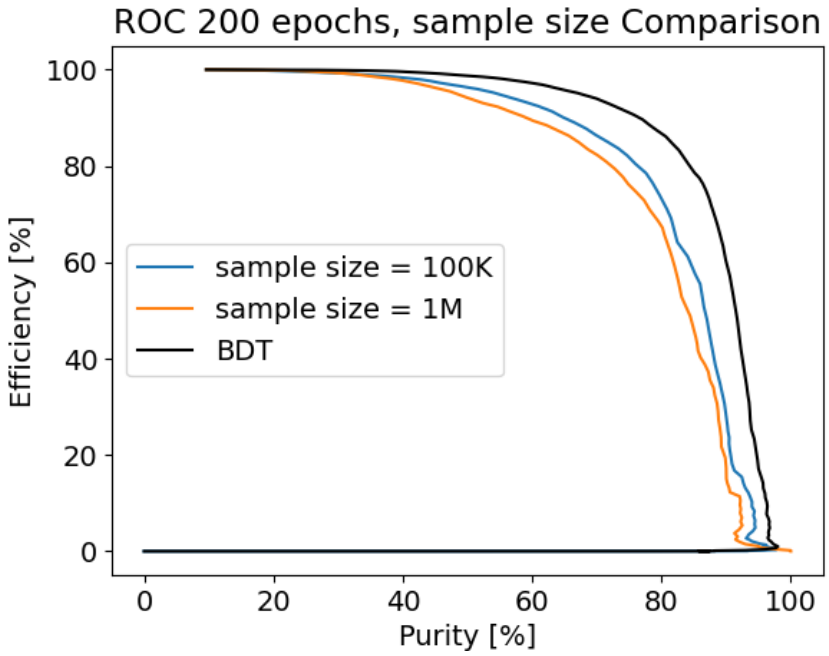


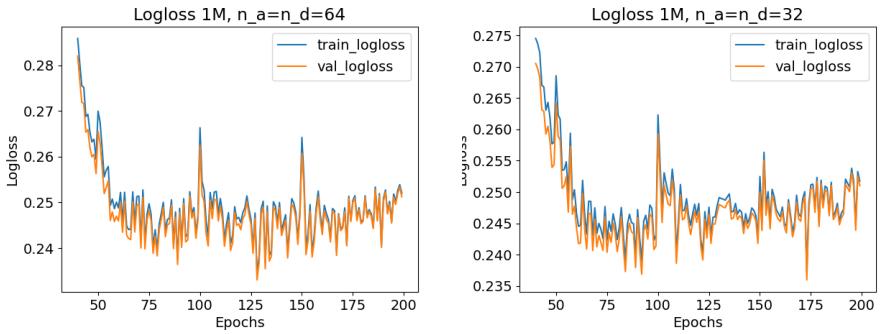
Figure 5.5: ROC-curve as in figure 4.3 trained on 100K events (blue), 1M events (orange), a total of 200 epochs and *batch_size* = 65536.

The model trained on 100K events performs better than the model trained on 1M events. This might be due to the choice of the other hyperparameter values, as the values of e.g. *n_d* and *n_a*, explained in section 3.3.3, is low. Therefore each following model is trained on 1M events and a *batch_size* of 66536.

5.1.3 Overfitting Check with n_da

The authors of [11] state that the choice of the width of the decision prediction layer n_d, defined in section 3.3.3 might cause overfitting. Therefore the impact of n_da, defined in section 3.3.3, is investigated in this section.

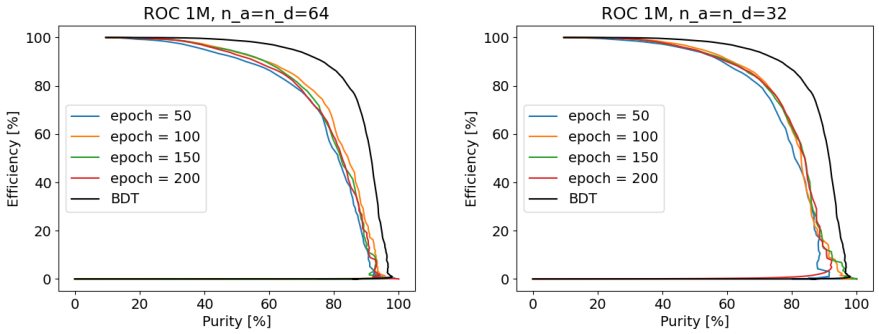
The model default hyperparameter value $n\_da = 8$, which is the minimum recommended value, did not yield any signs of overfitting as shown in the previous sections figure 5.5. In this section $n\_da = 32$ and $n\_da = 64$ are analysed, as 64 is the maximum recommended value and 32 is right in between the maximum and minimum recommended value.
Figure 5.6b and 5.6a show the loglosses of each model from epoch 40 to 200. Before epoch 40 the loglosses decrease exponentially (see figure A.3). Therefore the shown loglosses are zoomed in to tell whether the models overfit. Both models were trained on 1M events for 200 epochs. The logloss of the model trained with $n\_da = 64$ increases after reaching a minimum at epoch 128, implicating overfitting, as the logloss increases steadily afterwards. The logloss of the model trained with $n\_da = 32$ shows a successful training until epoch 100 where the logloss spikes. Until epoch 120 the logloss decreases. Afterwards it increases steadily and therefore also implicates overfitting, despite having its minimum after 173 epochs of training. Judging from the loglosses the training was successful until epoch 127 for $n\_da = 64$ and 100 for $n\_da = 32$.



(a) (b)

Figure 5.6: Logloss as in figure 4.1 showing epoch 50-200. The model trained on 1M events, 200 epochs, $batch\_size = 65536$ and (a) $n\_da = 64$, (b) $n\_da = 32$

In figure 5.7 the ROC-curves of the two models trained with $n\_da = 32$ and $n\_da = 64$ are shown, respectively. After every 50 epochs of training the models were saved with a maximum of 200 epochs.



(a) (b)

Figure 5.7: ROC-curve as in figure 4.3 trained on 1M events a total of 200 epochs saved after every 50 epochs, $batch\_size = 65536$ and (a) $n\_da = 64$ and (b) $n\_da = 32$.

Figure 5.7a shows the performance of the models trained with $n\_da = 64$. The model trained for 100 epochs has the best performance. Afterwards the models performances decrease, which supports the beginning of overfitting visible in figure 5.6a. Despite having the minimum after epoch 127 the performance of the model trained for 150 epochs is worse than the model trained for 100 epochs. In figure 5.7b the performance of the models trained with $n\_da = 32$ is shown. The performance after every 50 epoch almost follow the same curve. For purities below 75% the model trained for 100 epochs has the best performance, but for purities above 75% the model trained for 200 epochs has the best performance in this hyperparameter analysis. This might be due to the minimum in the logloss at epoch 173.
In both cases it is not certain if the models do have the best performance as shown in figure 5.7, since both show signs of overfitting in their loglosses. For the model trained with $n\_da = 32$ it seems plausible that the best performance is after 200 epochs as the model uses the weights of the best training epoch (here 173), which corresponds to the lowest logloss (see section 4.1). The model trained with $n\_da = 64$ does not show this behaviour, as its minimum is at 127 epoch and the models after 150 epochs of training are not outperforming the models with fewer epochs of training.

To conclude the analysis of the n_da hyperparameter, the performance of the best performing model trained with $n\_da = 64$ and $n\_da = 32$ is compared. Figure 5.8 shows the performance of the model default hyperparameter value compared to the two chosen values and the established BDT. The established BDT is still outperforming TabNet. The change of hyperparameter barely change the performance, therefore they are not the most optimal values to outperform the established BDT and a automated hyperparameter optimization was conducted.

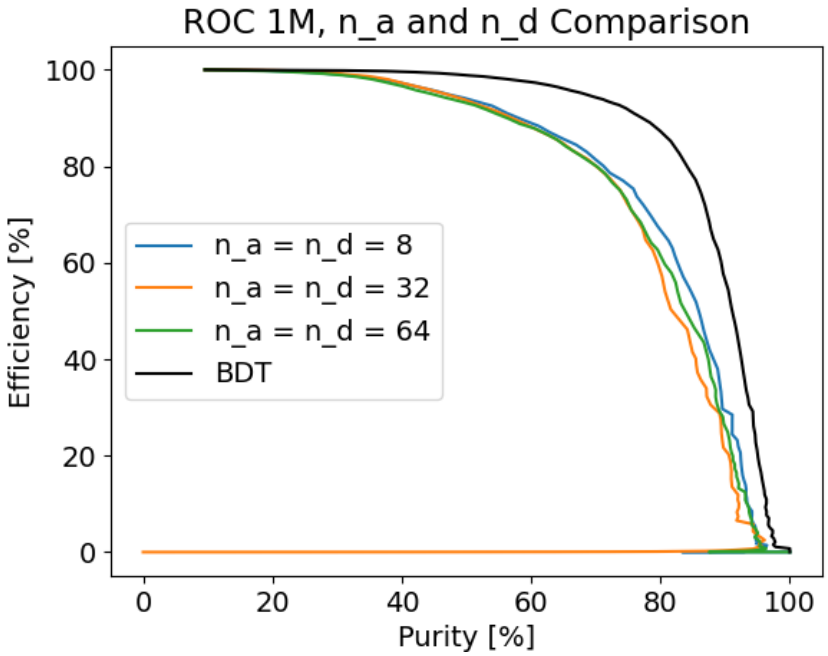


Figure 5.8: ROC-curve as in figure 4.3 trained on 1M, a total of 200 epochs , $batch\_size = 65536$ and $n\_da = 8$ (blue), 32 (orange), 64 (green).

5.2 Automated Hyperparameter Optimization

After conducting the manual hyperparameter optimization (see section 5.1) the performance of TabNet was still lower than the established BDT performance. Therefore changing a single variable seemed suboptimal in order to increase the performance, as the model trained with the model default hyperparameter value yielded the best performance in section 5.1.3. For that reason a high-dimensional automated hyperparameter optimization is conducted using the Opunta [2] framework. The goal is to increase the performance of TabNet by adjusting multiple hyperparameter values simultaneously and compare it to the performance of the established BDT.

Each model discussed in this section was trained using early stopping to get a maximum of trials in the given time as training the models can be very time consuming and early stopping tries to prevent overfitting. For the automated hyperparameter optimization the *patience* hyperparameter (see section3.3.3), controlling the early stopping, was set to 50 and used the validation AUC metric for early stopping. This metric was chosen as we are using the ROC-curve for performance analysis and the AUC describes the area under the ROC-curve (see section 4.2). The validation AUC metric will be referred to as *objective value* in the following, as the goal during the automated hyperparameter optimization was to maximize it.

Subsection 5.2.1 discusses the trials of the automated hyperparameter optimization. Subsection 5.2.2 discusses the performance of the model trained with the optimal hyperparameter values and compares it to the established BDT performance. Section 5.2.3 discusses the combination of hyperparameters.

5.2.1 Tunning of the Hyperparameter

The framework used for the automated hyperparameter optimization was Optuna [2]. Optuna requires a range of values for each hyperparameter that are optimized. Table 5.1 lists the hyperparameters and the step sizes, which controlled the change of the hyperparameters. Every hyperparameter that impacts the training was given with the recommended range of [11]. The step sizes were chosen to be as small as possible, while the number of possible hyperparameter values stayed low to optimize the time and therefore the number of trials. For some hyperparameter ranges no step size was given, as they either were not a range but a set of values or no recommended range was given. A single trial is the training of one model with a specific set of hyperparameter values. This means that different trials used different combinations of hyerparameter values. All trials together are called study.

Table 5.1: Hyperparamete ranges optimized in the automated hyperparameter optimization and their and steps sizes. "-" indicates no value was given.

Hyperparameter	**Range**	**Step**
n_da	$8 - 64$	2
n_steps	$4 - 10$	1
$gamma$	$1.2 - 2.0$	0.1
$n\_independent$	$2 - 5$	1
n_shared	$2 - 5$	1
$momentum$	$0.01 - 0.1$	0.01
$lambda\_sparse$	$0 - 10^{-3}$	–
$learning\_rate$	$2 \times 10^{-2} - 5 \times 10^{-2}$	–
$batch\_size$	$16384, 32768, 65536, 131072$	–
$virtual\_batch\_size$	$512, 1024, 2048, 4096$	–
$gamma\_scheduler$	$0.5 - 0.99$	0.01

In figure 5.9 the history of the automated hyperparameter optimization is shown. Each dot represents the *objective value* of a single trial ordered by their trial number. The red line represents the highest value of the *objective value* of all previous trials.
A total of 17 trials were conducted. Each trial was trained on 1M events. As early stopping was used in this study the maximum amount of epochs for each trial was different. The index of the trials do not correspond to the number of the trial, as some trials were interrupted, due to testing and technical difficulties. Here the labels of the trials are used and each model will be referred to as trial plus the index number. Trial 18 achieved the highest *objective value* of 0.9609422. Trial 22 achieved the second highest *objective value* of 0.9569875. Compared to bot the trials with the third to fifth highest *objective value* did not come close, as the difference between these two and the third best was in the order of 10^{-3}, while the difference between the third to fifth highest *objective value* trials was of the order of 10^{-5}.

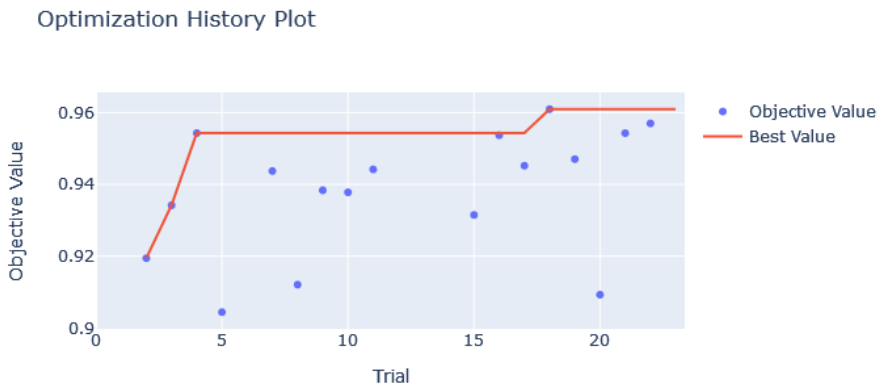
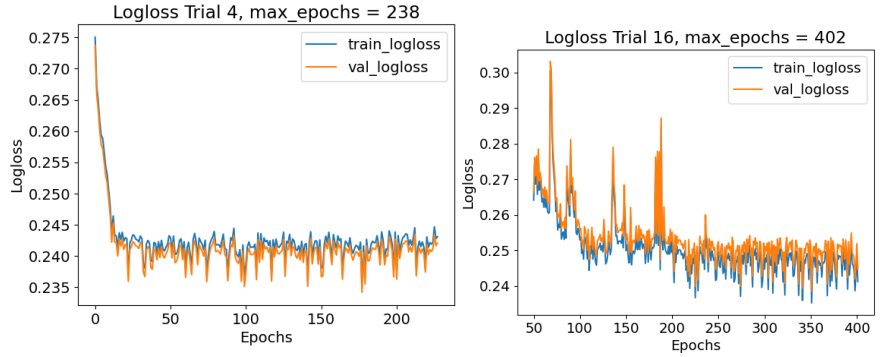


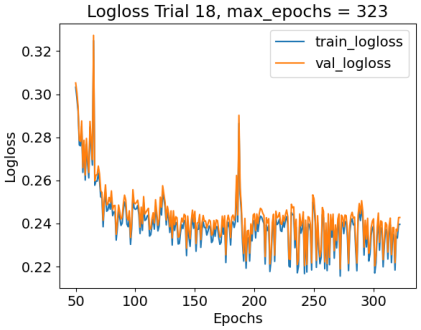
Figure 5.9: History of the automated hyperparameter study with Optuna [2]. A total of 17 trials were conducted, while the labels of the trials were not continual as some trials were interrupted. The red line shows the objective value of earlier trials.

In figure 5.10 the zoomed logloss of the models, ordered by their trial index, achieving the five highest *objective values* are shown. The logloss of trial 4 shown in figure 5.10a decreases the fastest and stays low for the entire training showing no signs of overfitting. In figure 5.10b the logloss of trial 16 is shown. Despite having some spikes the logloss decreases over all and also shows no signs of overfitting. In figure 5.10c the logloss of trial 18 is shown. This trials logloss also has spikes but decreases steadily with no signs of overfitting. In figure 5.10d the logloss of trial 21 is shown. It has the smallest *max_epochs*. The logloss decreases steadily and shows no signs of overfitting. In figure 5.10e the logloss of trial 22 is shown. After 102 epochs the logloss starts to increase indicating that the model is overfitting.
Besides trial 22 no model is overfitting. As the *patience* was set to 50 and trial 22 trained for 129 epochs the overfitting is visible in 5.10e. In that case early stopping prevented the model from overfitting. Trial 18 achieved the lowest logloss, but as the logloss values are very low the difference is only marginal. In order to see which model performed the best the ROC-curve needs to be investigated (see section 5.2.2).
Every models training was succesful and showed no signs of overfitting besides trial 22, which started to overfit after 102 epochs of training. The hyperparameter values of trial 4, 16, 18/22, 21 are given in tables A.3, A.4, 5.2 and A.5, respectively.

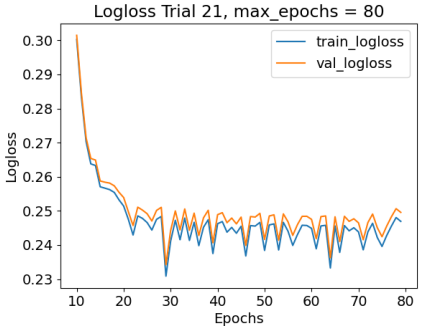


(a)

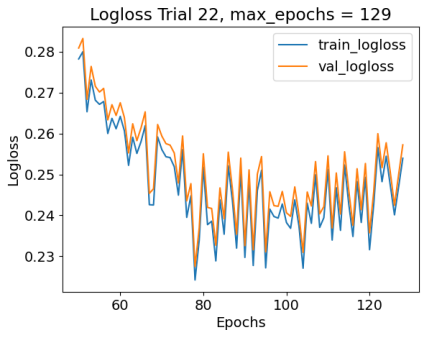
(b)



(c)



(d)



(e)

Figure 5.10: Loglosses as in figure 4.1. The models trained on 1M events with the maximum number of epochs and hyperparameter values of (a) trial 4, (b) trial 16, (c) trial 18, (d) trial 21, (e) trial 22.

In figure 5.11 the hyperparameter importance based on the Optuna study is shown. The sum of all hyperparameter importances is 1. *n_steps* and *n_da* make up 0.64, which is more than all other hyperparameter importances combined. The gap between these two hyperparameters and the third most important (*n_independent*) is 0.17, which is more than the hyperparameter importance of every other hyperparameter. After *n_independent* there is another notable gap. The difference between *n_independent* and *n_shared*, which is the fourth most important hyperparameter, is 0.08, which again is higher than every other following value. After these two gaps the importance is relative evenly spread across the remaining hyperparamters. In the use-case of TabNet in this thesis the hyperparameters therefore have very different contributions to the learning process of TabNet.

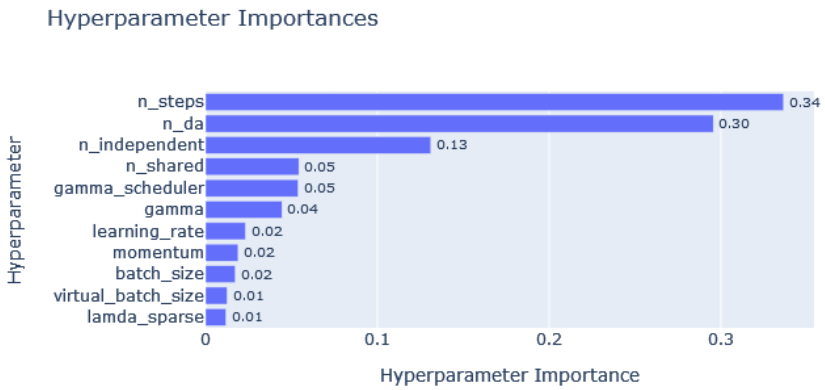


Figure 5.11: Hyperparameter importance for TabNet based on the conducted Optuna study.

5.2.2 Boosted Decision Tree vs TabNet Performance

After checking if the five models with the highest *objective value* are overfitting, (see section5.2.1) the performance is compared in figure 5.12.

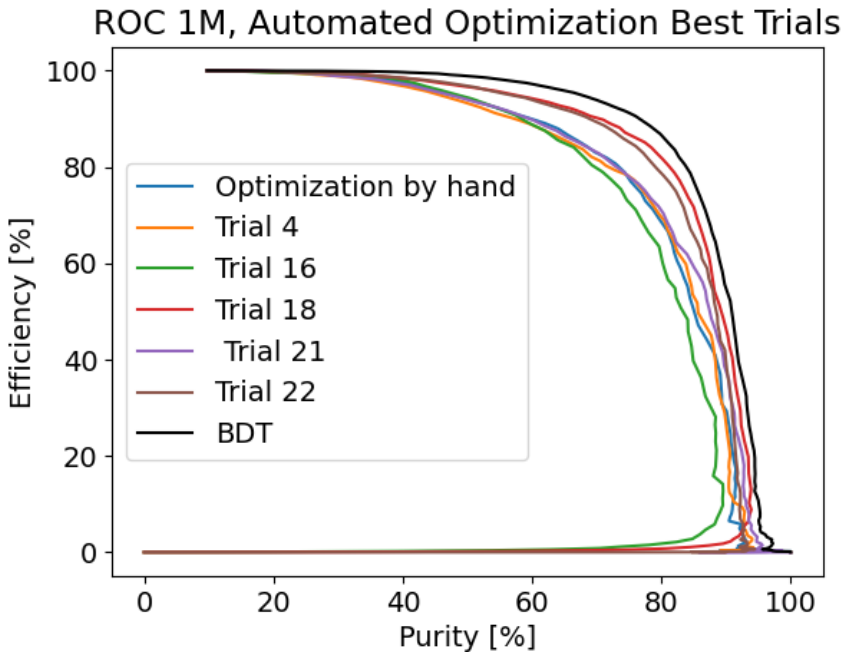


Figure 5.12: ROC-curve as in figure 4.3 trained on 1M events, with the maximum number of epochs and hyperparameter values of best optimization by hand (blue), trial 4 (orange), trial 16 (green), trial 18 (red), trial 21 (purple), trial 22 (brown).

Trial 22 (brown) performs only slightly worse than trial 18. All the other trials perform worse. Trial 18 (red) performs the best. Despite the increase in performance with the automated hyperparameter optimization compared to the model with the model default hyperparameter values (see figure 5.13) the established BDT is still outperforming every TabNet model discussed in this analysis. In order to beat the established BDT performance TabNet would need another increase in performance comparable to the increase from trial 4 (orange) to trial 18 (red). As this increase occurred within the last trials more trials need to be conducted, which was not possible during this analysis due to time, as the training of a single model remains time consuming even when including early stopping. Therefore trial 18 might not yield the most optimal hyperparameter values for event selection of the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay.

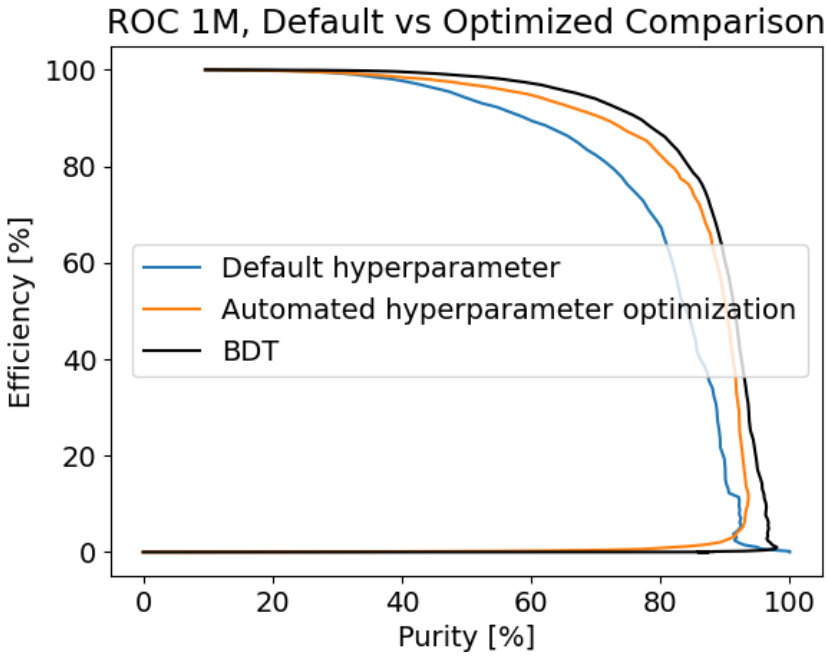


Figure 5.13: ROC-curve as in figure 4.3 trained on 1M events with the model default hyperparameter values for 200 epochs (blue) and maximum number of epochs and hyperparameter values of trial 18.

5.2.3 Combinaton of Hyperparameter

In table 5.2 the best hyperparameter values of the best two trials are listed. Eight out of eleven hyperparameters have different values. Despite the results of section 5.1.3 the selected hyperparameter values for the width of the decision prediction layer and the attention masks embedding (n_da) tends to be larger than the model default hyperparameter value $n\_da = 8$ and overfitting does not occur (see section 5.2.1). This indicates that the combination of hyperparameter values is important and the change of a single hyperparameter value will most likely not improve the performance. Figure 5.14 gives clues about the interplay of hyperparameter values.

Table 5.2: Hyperparameter, maximum epochs and validation AUC of Trial 18 and Trial 22.

	Trial 18:	**Trial 22:**
Hyperparameter	**Value**	**Value**
n_da	52	42
n_steps	8	6
gamma	1.6	1.2
n_independent	5	5
n_shared	5	4
momentum	0.07	0.07
lambda_sparse	$9.831\,368\,331\,738 \times 10^{-6}$	$3.907\,557\,640\,709\,79 \times 10^{-4}$
learning_rate	0.047533385594715795	0.032519974768828466
batch_size	66536	65536
virtual_batch_size	2048	512
gamma_scheduler	0.99	0.91
max_epochs	323	129
val_auc	0.9609422	0.9569875

A low feature reusage (*gamma*) is combined with a large coefficient for the sparsity-loss (*lambda_sparse*) to get a sparse model and vice versa for not sparse models in order to achieve high *objective values*. Trial 22 is an example for a sparse model, as the feature reusage (*gamma*) is low and the sparsity-logloss (see section 4.1) is multiplied with a large coefficient (*lambda_sparse*) compared to trial 18. Trial 18 on the other hand is an example for a not sparse model as the feature reusage (*gamma*) is high and the sparsity-loss is almost multiplied by a coefficient (*lambda_sparse*) close to 0. The combination of a high feature reusage and a high coefficient for the sparsity-loss was also considered but did not achieve large *objective values*, when compared to sparse or not sparse model combination. This shows a tendency towards either sparse models or not sparse models. As trial 22 and trial 18 are example for a sparse and not sparse model, respectively, it is not save to say which sparsity is optimal for the even selection of $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay.

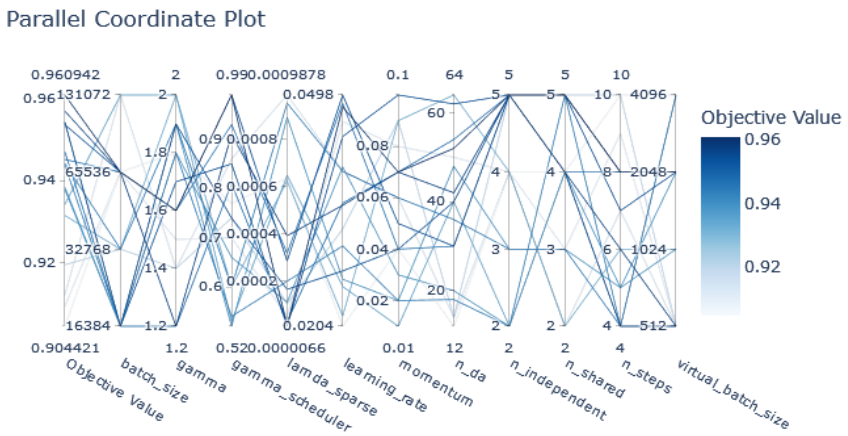


Figure 5.14: Parallel Coordinate plot of the automated hyperparameter optimization. It show the relation between the hyperparameters. The blue tone depicts the objective value. The lighter the blue tone, the further the combination of hyperparameters was from the best achieved objective value.

The second interplay visible in figure 5.14 is the combination of the step size (*learning_rate*) and the adjustment of it (*gamma_scheduler*) after every epoch during training. A large step size corresponds to a small adjustment after every epoch (see section 3.3.3) and a small step size corresponds to a large adjustment of it. The combination of a large step size with small adjustments yielded the better results. This implies that the model tries to learn fast but risks overshooting minima.
Four possible batch sizes were given but only two yielded large *objetive values*. The *batch_size* yielding the best results is the same as in section 5.1.2 (65536). The second *batch_size* considered was 16384. The relation of the *batch_size* with the *virtual_batch_size* was known before figure 5.14, as *batch_size* is a multiple of *virtual_batch_size*. This thesis chose $batch\_size = 32 \times virtual\_batch\_size$, which is supported by figure 5.14 as 2048 and 512 were the only choices yielding large *objective values*.
The last interplay visible in figure 5.14 is the combination of the number of independent (*n_independent*) and shared (*n_shared*) transformer blocks (see section 3.3.1). A hyperparameter value of 5 is selected for both in almost every trial yielding a large *objective value* with the exceptions of trial 4 and 22, which selected $n\_shared = 4$. As the upper limit is selected most of the time a larger upper limit in the range for the hyperparameter values might give a better idea of the hyperparameter value needed for an improvement in performance. The choice of both hyperparameters having the same hyperparameter value indicates $n\_independent = n\_shared$ as a good choice, which is the same behaviour as for the hyperparameter values of the width of the decision prediction layer and the attention mask (*n_da*) (see section 3.3.3). If this holds for a larger upper limit

of the range is uncertain.
The number of decision steps (*n_steps*) and the momentum for BN (*momentum*) indicated no interplay wit other hyperparameters, as the selected hyperparameter values showed no tendencies.

In conclusion the automated hyperparameter optimization yielded a better performance than the manual hyperparameter optimization. Although optimizing the performance of TabNet the established BDT still outperforms every TabNet model in event selection for the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay obtained in this analysis.
In section 5.1.3 the default model value was performing the best, while the larger *n_da* hyperparameter values overfitted. Compared to trial 18, where the hyperparameter value $n\_da = 52$ was selected due to the combination with other hyperparameter values, the performance increased and did not overfit. Therefore when training the model an automated hyperparameter optimization is recommended as tuning a single hyperparameter can give wrong clues for the optimal setting of a hyperparameter value and the combination of hyperparameter values is important for a good TabNet performance.

Chapter 6

Feature Importance Analysis

Interpretability plays a crucial role in understanding how ML and DL algorithms make their predictions and building confidence and trust in these algorithms. Explainable AI is a whole field of study dedicated to make Artificial Intelligence (AI) interpretable. In this thesis the Shapley value (see section 4.3) is used to interpret the decision making of TabNet and the established BDT as additional source of information about the interpretability, next to the intrinsic interpretability of BDTs and TabNet. Shapley values measure the importance of the features for the predictions, this is referred to as feature importance. Using the Shapley value for interpretability of the established BDT and TabNet makes the feature importance comparable, as the intrinsic measure of feature importance differs for BDTs and TabNet.
Each model was trained using the same hand-made set of features, but the use of the information of the features can be different for the models, as some features can obtain redundant information. The goal is to partly understand how TabNet learns from these features compared to the established BDT. Minimizing the feature set is also important to get rid of unwanted biases. The Shapley value for both models was calculated using the same data set as well as the same background set in order to make the results comparable.

Section 6.1 discussed the calculated Shapley values of TabNet. Section 6.2 discusses the comparison of the TabNet and established BDT Shapley values. Section 6.3 discussed the intrinsic interpretability of TabNet and the established BDT.

6.1 Shapley Values of TabNet

Figure 6.1 shows the distribution of the Shapley values for each of the input feature value. On the x-axis the calculated Shapley value is displayed, where a

positive Shapley value pushes the prediction towards signal (1) and a negative Shapley value pushes the prediction towards background (0). The colors of the data points display the feature value. A high feature value corresponds to a more red data point and a low feature value corresponds to a more blue data point.

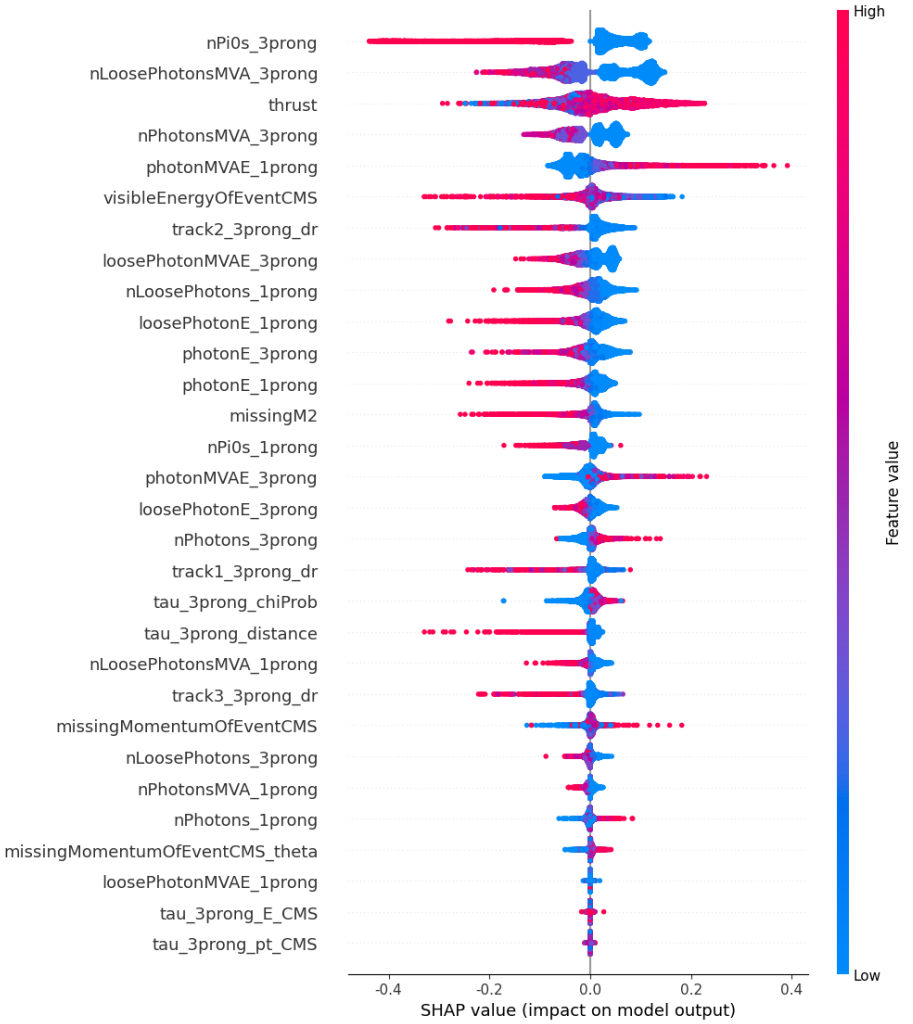


Figure 6.1: Beeswarm plot of the TabNet Shapley values. Shown is the distribution of the calculated Shapley value for each of the input feature values. The color of the data points indicate the feature value. Blue corresponds to low feature values, red corresponds to high feature values. The features are displayed in the order of their importance.

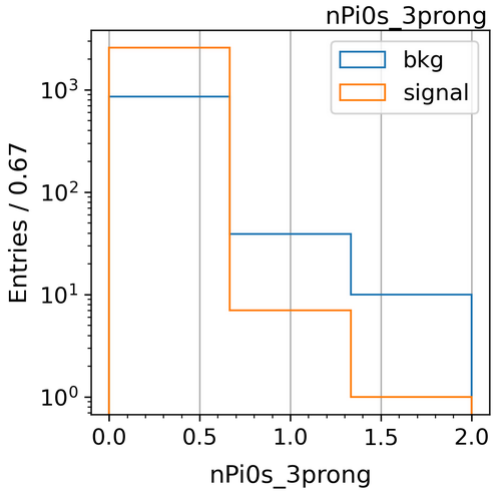


Figure 6.2: Distribution of the number of π^0 for signal (orange) and background (blue) events.

From the distribution in the number of π^0 on the three prong side, shown in figure 6.2 it is clear that almost all signal events have $nPi0s\_3prong = 0$. Comparing this with the corresponding Shapley values in the first line of figure 6.1, low numbers of π^0 on the three prong side correspond to a positive Shapley value and therefore push the prediction towards signal. This makes sense as there is no π^0 in the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay. Larger numbers of π^0 on the three prong side lead to negative Shapley values and therefore push the prediction towards background.
The feature importance given by the Shapley values is calculated by taking the mean of the absolute of the Shapley values per feature. The feature importance is shown in figure 6.3 and corresponds to the ordering of features in figure 6.1. Overall the feature importance is decreasing continuously with the first features having the largest feature importance and some features having almost no feature importance. The three most important features are the number of π^0 on the three prong side, the loose and MVA cut photons on the three prong side and the thrust. The three least important features are the energy of the loose and MVA cut photons on the one prong side, the energy of the τ on the three prong side and the transverse momentum of the τ on the three prong side. All three features have a feature importance of almost zero, which means they do not contribute to the prediction of TabNet. Between the three most important features and the following features a gap in the feature importance is visible. A second gap is visible between all features and the three least important features.

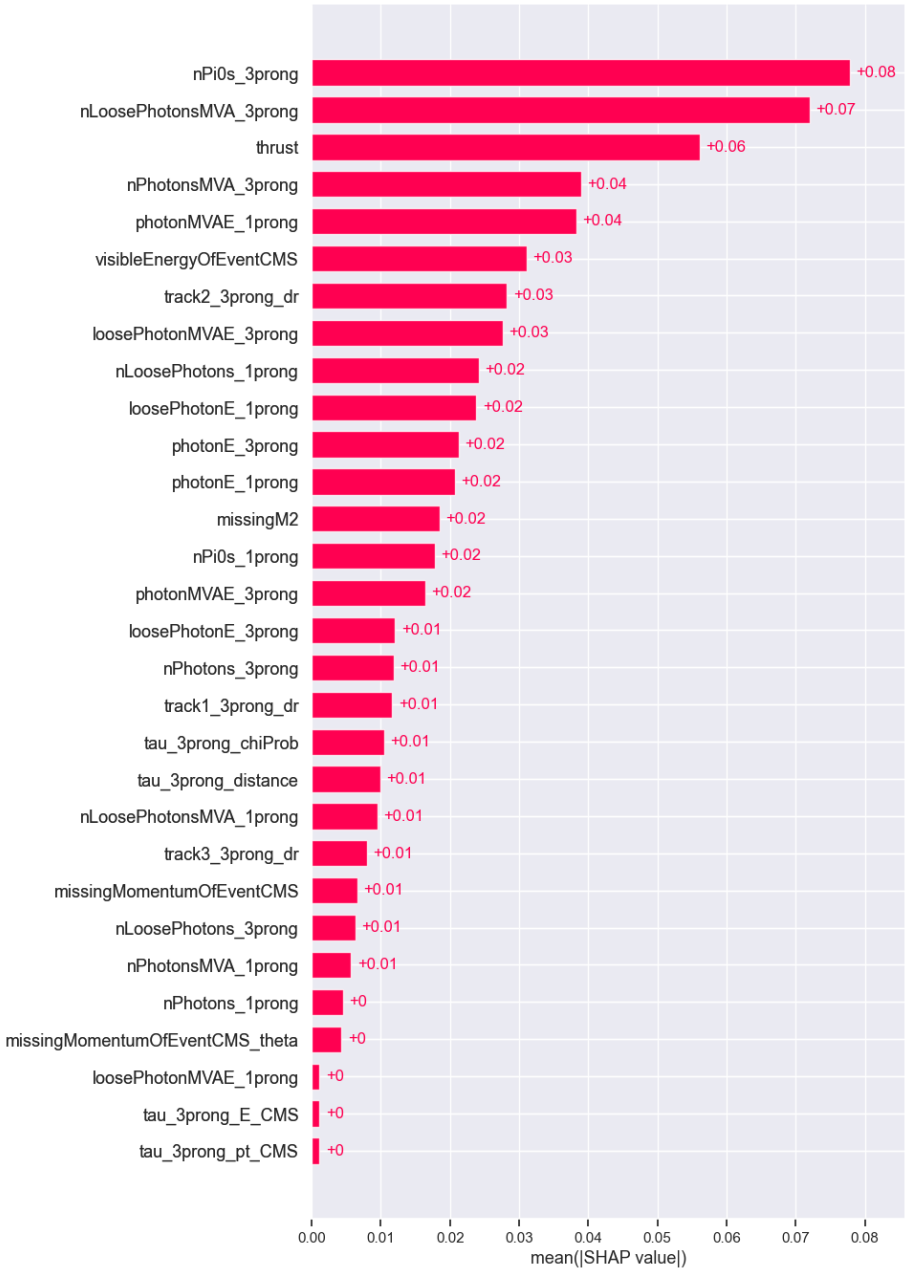


Figure 6.3: Feature importance of TabNet calculated via the mean of the absolute Shapley values per feature.

The most important feature is the number of π^0 on the 3 prong side. This feature belongs to the γ and π^0 rejection features (see section 3.1.1). As there

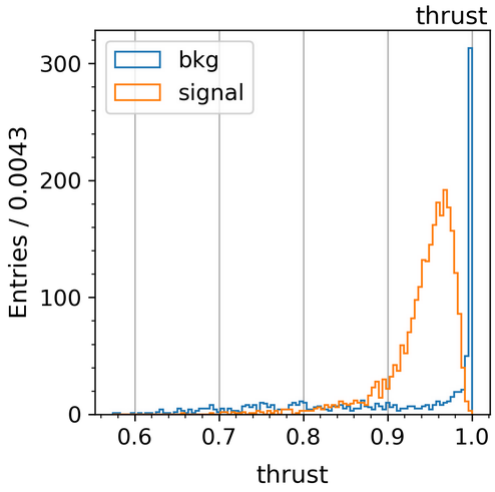


Figure 6.4: Distribution of the thrust values for signal (orange) and background (blue) events.

are no π^0 in the signal $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay this is a good separator between signal and background, explaining its high feature importance.
The *thrust* is also a good indicator between signal and background, as τ leptons have a mass of 1.777 GeV [17] which is small compared to the CMS energy of 10.580 GeV. Therefore the τ leptons have a large momentum in the CMS. Consequently the event-shape is jet-like with a thrust value in the range of 0.85 and 0.99 as shown in the orange histogram in figure 6.4. In contrast, background events strongly peak at thrust values of 1 corresponding to Bhabha events or have a broad bump of low thrust value from e.g. $B\overline{B}$ events.
The loose and MVA cut photons on the three prong side also belongs to the γ and π^0 rejection features. As this is the softest cut it includes photons of the 200 MeV cut on the three prong side, the 100 MeV cut on the three prong side and the MVA cut. Therefore the information given to TabNet with the other cuts on the three prong side contain redundant information, as photons can already be included in the loose and MVA cut photons on the three prong side. This is supported by the correlation of the features shown in figure 6.5. Green corresponds to a positive linear correlation, brown corresponds to a negative linear correlation and white corresponds to no linear correlation.

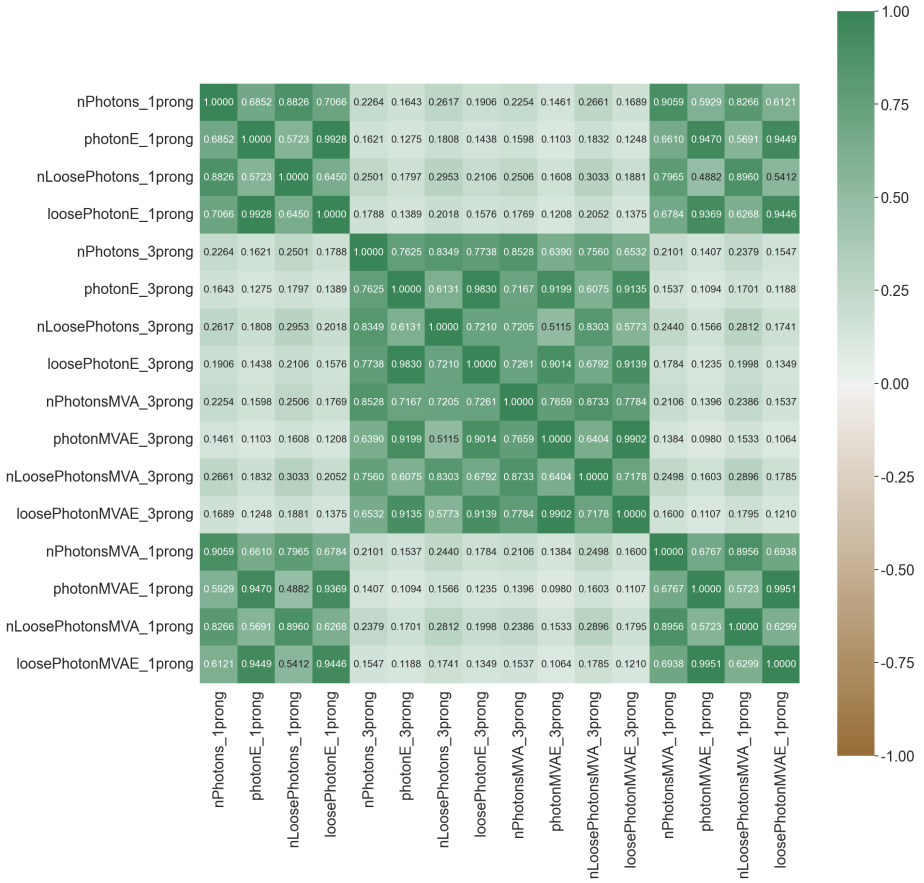


Figure 6.5: Correlation of the γ and π^0 rejection features.

The green blocks of the γ and π^0 rejection features on the one and three prong sides show, that these features have a positive linear correlation, respectively and therefore contain redundant information. Between the prongs there was almost no linear correlation, as the correlation values are low. The most important feature on the one prong side was the energy of the MVA cut photons. This feature was the fifth most important feature overall.

The kinematic feature can be divided into two subgroups. The first group includes the visible energy of the event in the CMS, the squared missing mass, the missing momentum of the event in the CMS and the angle theta of the missing momentum of the event in the CMS (see section 3.1.1). The most important feature of this group (see figure 6.3) is the visible energy of the event in the CMS. Figure 6.6 shows the linear correlation of the kinematic features of this group.

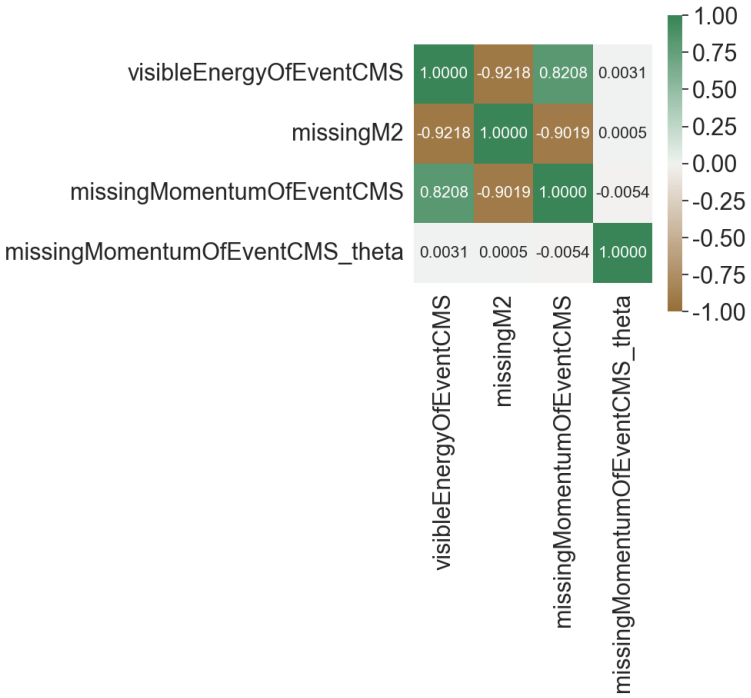


Figure 6.6: Correlation of the first group of kinematic features.

The visible energy of the event in the CMS has a positive linear correlation with the missing momentum of the event in the CMS and a negative linear correlation with the squared missing mass. The squared missing mass also has a negative linear correlation with the missing momentum of the event in the CMS. The only feature not linearly correlating with any other feature of this group nor any other input feature (see figure A.4) is the angle of the missing momentum in the CMS. This feature has the least feature importance out of this group. This means that either the information obtained from this feature is not crucial for the prediction or the information of this feature is included in features that correlate on higher orders.

The second group of kinematic features include the energy of the τ on the three prong side and the transverse momentum of the τ on the three prong side. Both features have the lowest feature importance for the TabNet model and have a positive linear correlation of 0.7147 with each other, but no other linear correlation with other input features. This means that these two features were not important for TabNet to make predictions.

The last category of features are the vertex reconstruction features (see section 3.1.1). They only have a slight linear correlation shown in figure 6.7 and do not correlate with any other input feature. The most important feature of

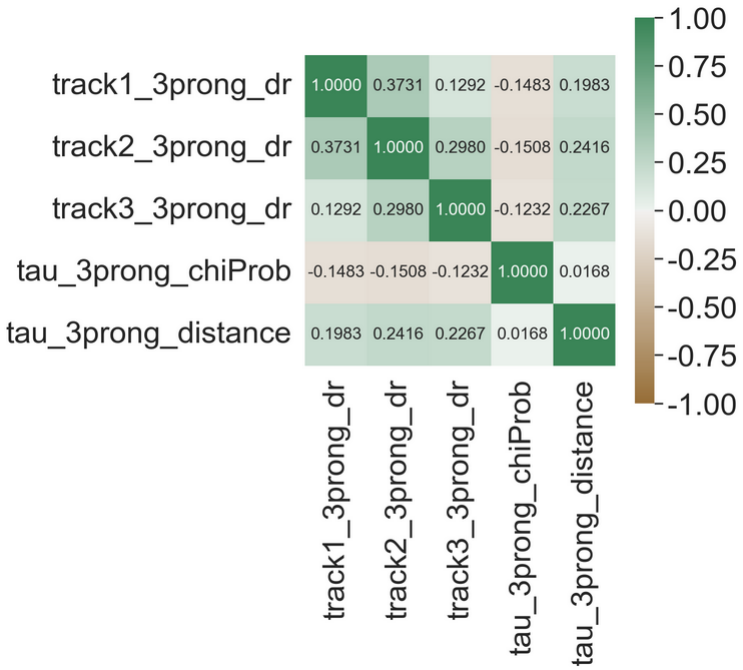


Figure 6.7: Correlation of the vertex reconstruction features.

this category is the second track on the three prong side. It is the seventh most important feature overall, while the first track on the three prong side is the 18th most important feature overall, which is the second most important feature of this category. As they do not strongly linearly correlate they seem to be less important for the predictions of TabNet than the thrust and γ and π^0 rejection features.

In conclusion TabNet makes its predictions using almost all features, where every feature has a different feature importance for TabNet. The number of π^0 and the thrust value are good separators for signal and background. This is why they are part of the top three most important features. The second most important feature also accounts for the γ and π^0 rejection features, as they contain redundant information that is also given in the number of loose and MVA cut photons. The least important category of features are the vertex reconstruction features. The kinematic features include the least important features, that have almost a feature importance of zero.

6.2 Boosted Decision Tree vs TabNet Shapley Values

Figure 6.8 shows the feature importance of the established BDT. The feature importance of the established BDT follows a smooth curve and shows no gaps. The redundancy of the γ and π^0 rejection features is clearer shown as many of these features show a small feature importance. The most important features of the γ and π^0 rejection features are the number of π^0, which is the fourth most important feature overall and the energy of loose and MVA cut photons on the three prong side followed by the number of loose and MVA cut photons on the three prong, which are the sixth and seventh most important feature, respectively. The most important γ and π^0 rejection feature on the one prong side is the number of loose cut photons on the one prong side.
The *thrust* is also the third most important feature for the established BDT, which shows again that this feature is a good separator for signal and background.
The first group of kinematic features consisting of the visible energy of the event in the CMS, the squared missing mass, the missing momentum of the event in the CMS and the angle theta of the missing momentum of the event in the CMS has a higher feature importance overall. The visible energy of the event in the CMS is again the most important feature of this group and the fifth most important feature overall. The next two features with the highest feature importance of this group are the missing momentum of the event in the CMS and the angle theta of the missing momentum of the event in the CMS. Both features had the lowest feature importance of this group for TabNet, with the angle theta of the missing momentum of the event being the fourth least important feature almost having a feature importance of 0. For the established BDT the squared missing mass was the least important feature of this group but is still the 14th most important feature overall, ranking this group of features higher in feature importance compared to its feature importance for TabNet.

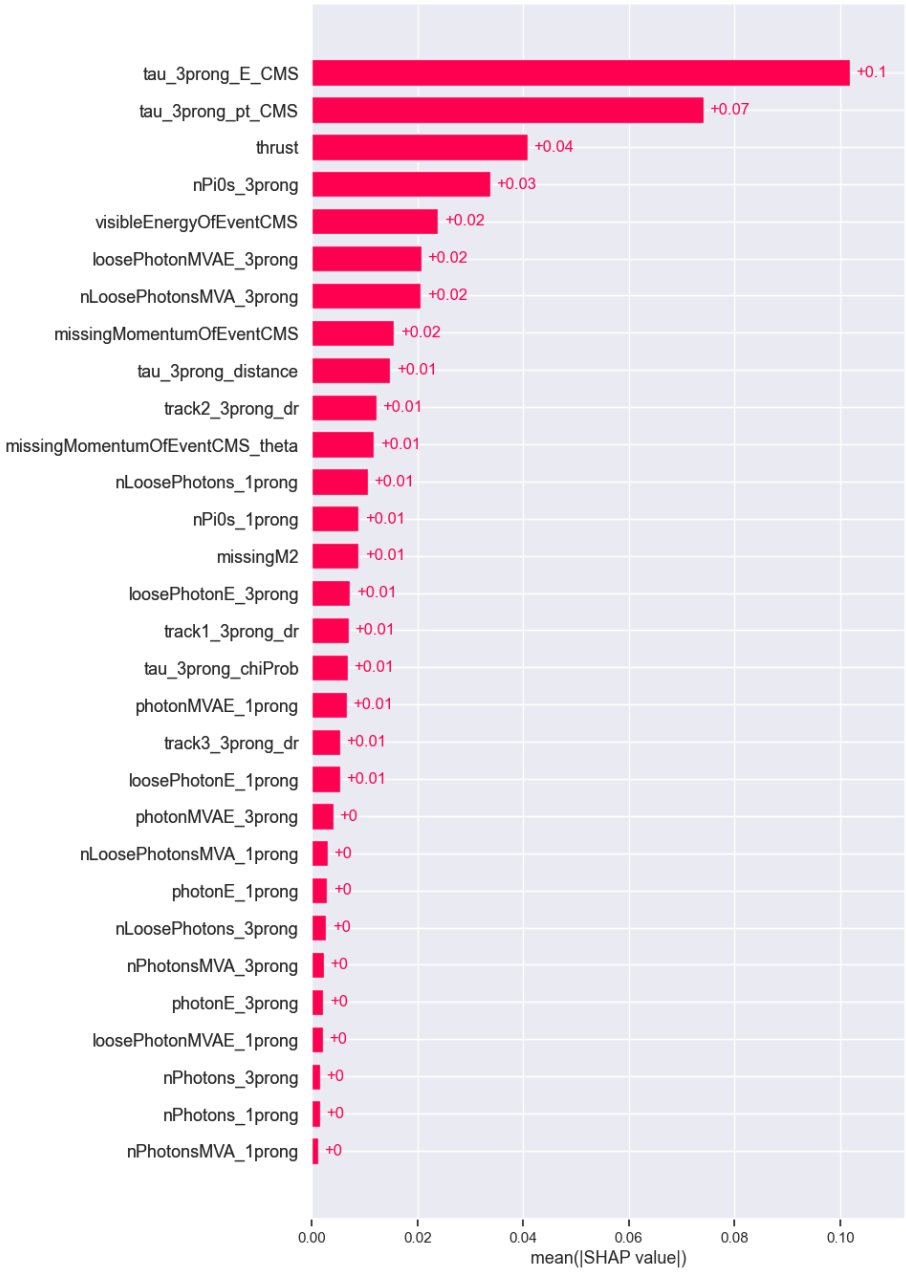


Figure 6.8: Feature importance of the established BDT calculated via the mean of the absolute Shapley values per feature.

The second group of the kinematic features consisting of the energy of the τ on the three prong side and the transverse momentum of the τ on the three

prong side are the most important features for the established BDT.

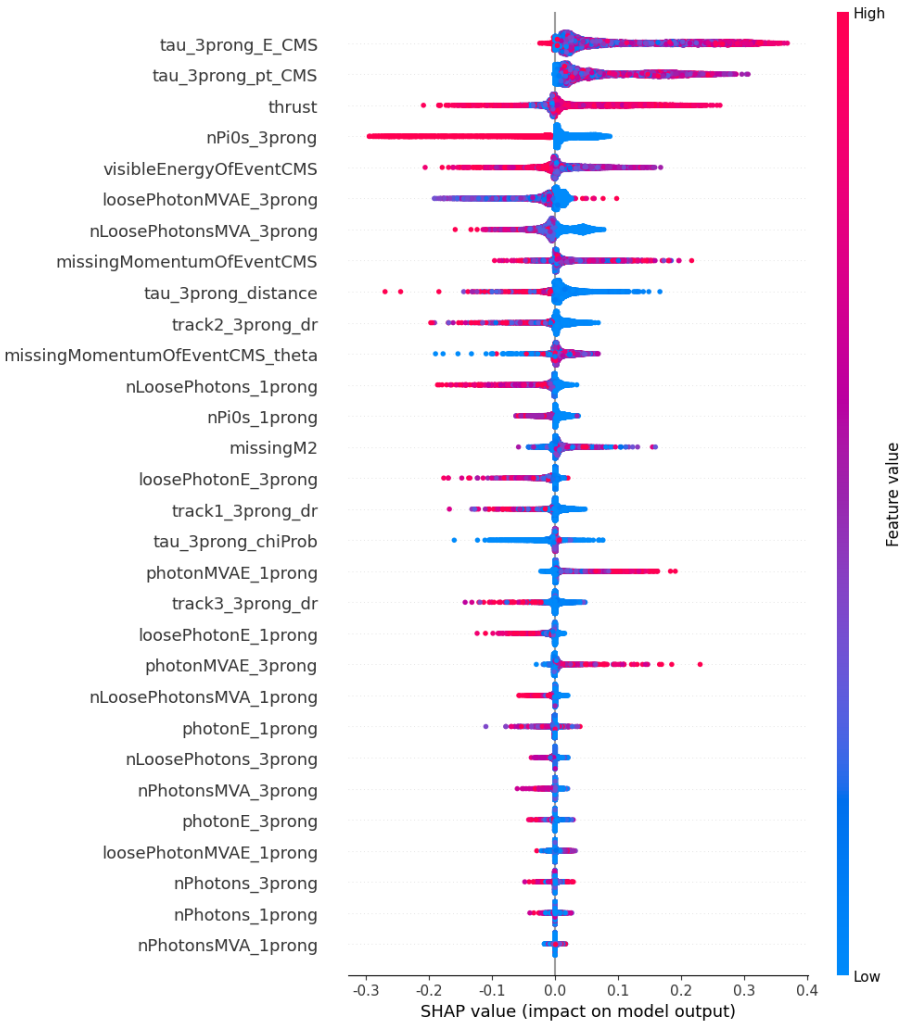


Figure 6.9: Beeswarm plot of the established BDT Shapley values. Shown is the distribution of the calculated Shapley value for each of the input feature value. The color of the data points indicates the feature value. Blue correspond to low feature values, red corresponds to high feature values. The features are displayed in the order of their importance.

Compared to TabNet where they had no feature importance and did not contribute to the prediction (see section 6.1). Figure 6.9 shows the beeswarm plot of the established BDT. Almost every value for the energy of the τ on the three prong side and the transverse momentum of the τ on the three prong side pushes

the prediction towards signal, with a small portion of high values of the energy of the τ on the three prong side pushing the prediction towards background. This distribution looks odd compared to the TabNet distribution shown in figure 6.1 and the other features shown in figure 6.9. The expectation is a distribution that has values pushing the prediction to signal and background not only to signal. This may be caused by the assumption made when calculating the Shapley values (see section 4.3), as e.g. the established BDT is not linear.
The most important vertex reconstruction feature is the distance of the τ on the three prong side. It is the ninth most important feature overall, immediately followed by the second track on the three prong side. The least important feature of this category of features is the third track on the three prong side, being the 19th most important feature overall. Compared to TabNet this category of features is more important to the established BDT, as the second most important feature of this category for TabNet was the 18th most important feature leaving a big gap between the two most important features.

In conclusion the feature importance of TabNet and the established BDT only agree on the importance of the thrust values. When ranking the feature importance not based on the single features but base on the categories of features the most important category is the event shape. The second most important category of features are the kinematic feature for the established BDT and the γ and π^0 rejection features for TabNet. The third most important category for the established BDT and for TabNet are the vertex reconstruction features. The least important category for the established BDT are the γ and π^0 rejection features and the kinematic features for TabNet.

6.3 Intrinsic Interpretability of BDT and TabNet

The established BDT and TabNet also have an intrinsic feature importance, which is not common in the case for DL algorithms like TabNet. The BDT measures the feature importance by either counting the times a feature is used to split the data set or by weighting the counts of splitting with the information gained through this feature. TabNet provides interpretability by aggregating the attention masks and by calculating the mean of the attention masks to get a global feature importance. The intrinsic intepretability of TabNet is not shown as there seems to be a bug. When changing the thrust value for a signal event to a thrust value of a background event the prediction changes from signal to background, but the global feature importance of the thrust provided by TabNet is 0 and consequently the prediction should not be altered. Therefore the intrinsic interpretabilty of TabNet is not to be trusted, until this bug is fixed.
Besides the bug of TabNet comparing the intrinsic interpretability of the established BDT and TabNet is not meaningful as the definition of interpretability and feature importance are different. They can only be compared qualitatively.

Chapter 7

Conclusion and Outlook

This thesis studied whether TabNet, a DL algorithm, can outperform the established BDT in selecting the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay in Belle II data. We also studied the way TabNet makes its predictions, how the prediction making differs from the established BDT and derived the importance of the input features from this with the help of the SHAP library [27].

The optimization of the TabNet performance was split into two parts. The first part studied the impact of preproccessing and certain hyperparameter values on the TabNet performance by manually adjusting them. The second part conducted a high-dimensional automated hyperparameter optimization using the Optuna library [2].
The first step in the manual hyperparameter study of TabNet was to establish a suitable scaler for the data. Two models with different scalers were studied i.e. StandardScaler and the MinMaxScaler of the sklearn.preprocessing library [22]. Additionally, a model with no scaling was studied as the authors of ref. [4] state that preprocessing is not needed for TabNet. Despite the statement of the authors of [4] the MinMaxScaler model yielded the best performance, with a slightly better performance than the model trained without scaled data. Therefore the data used for further training of different models was scaled with the MinMaxScaler.
Next the training sample size and along with it the *batch_size* was studied. Three *batch_sizes* of $2 - 8\%$ of the training sample size were studied, where a *batch_size* of 8% performed the best supporting the need for a large *batch_size*. After the *batch_size* was adjusted for the model trained on 1M events the model trained on 100K events outperformed the model trained on 1M events. The following analysis still used 1M events, as the other hyperparameter values were the model default hyperarameter values and from experience a larger sample size is performing better.
The next hyperparameter studied was the width of the decision output layer and the attention masks embedding, as the authors of [11] state that this contains a risk of overfitting. Indeed the models overfitted, with the model default

hyperparameter values performing the best. This led to an automated hyperparameter optimization, as the change of a single hyperparameter value did not yield a better performance than the established BDT.
During the automated hyperparameter optimization, with the Optuna [2] framework, multiple hyperparameter values were adjusted at the same time. A total of 17 trials were conducted. The three most important hyperparameters were the number of decision steps, the dimension of the decision output and attention masks embedding and the number of independent transformer blocks. All of these hyperparameters make the model more complex as they increase the parameters of TabNet. Therefore the complexity of TabNet seems to have the most influence on the performance. Additionally changing a single hyperparameter value gives wrong clues about the performance of a hyperparameter due to the interplay of the hyperparameter values. Therefore using a high-dimensioal automated hyperparameter optimization is recommended instead of changing a single hyperparameter value.
After the 17 conducted trials the performance of TabNet was still not better than the performance of the established BDT.

The feature importance analysis using the Shapley value enabling the comparison between models in terms of the way prediction were made. First the TabNet feature importance were studied. For TabNet the thrust and the number of π^0 on the three prong side provided a good separator for signal and background. The most important feature of the γ and π^0 rejection features, aside from the number of π^0 on the three prong side, were the loose and MVA cut photons. The other γ and π^0 rejection features contained redundant information and were therefore partly accounted for in the number of π^0 on the three prong side and in the softest cut of photons. The least important features that had a feature importance of almost 0 were the energy of the τ on the three prong side and the transverse momentum of the τ on the three prong side. This means they had no contribution to the prediction of the model, despite having no linear correlation with other features.
Comparing the feature importance from TabNet with the established BDT in terms of feature categories they only agree on the importance of the thrust feature, which is the only included event shape feature and the vertex reconstruction features. For TabNet the γ and π^0 rejection features were more important than the kinematic features, which was the opposite for the established BDT. Still both categories are important for both models. The two most important features of the established BDT were the energy of the τ on the three prong side and the transverse momentum of the τ on the three prong side, which had no feature importance for TabNet. The Shapley values of these two features pushed the prediction towards signal almost everytime, which should not be the case and indicate that the assumptions made during the calculation of the Shapley values may not be applicable for the established BDT, but due to time this could not be studied further.

As only 17 trials were conducted during the automated hyperparameter op-

timization and the performance improved a lot in the last few trials more trials may lead to an even larger improvement in performance. In addition to the possible improvement in performance more trials could result in an emerging cluster for combinations of hyperparameter values and make the impact of the combination of hyperparameters on the performance more clear. The behaviour of the number of shared and independent transformer blocks could also be studied further as in this thesis the same value for both hyperparameter values seemed to be favoured and showed the same behaviour of the dimension of the decision output and attention masks embedding. For that the range of the hyperparameter values need to be increased and more trials need to be conducted.
Despite the established BDT still outperforming TabNet after the conducted analysis in the event selection of the $\tau^- \to \pi^-\pi^-\pi^+\nu_\tau$ decay it can be an alternative to a BDT for selecting other processes because TabNet is still very good and close to the performance of the established BDT within the Belle II data. For each decay TabNet needs to be retrained and reoptimized using a high-dimensional automated hyperparameter optimization with larger hyperparameter value ranges as in this analysis.

Appendix A

Appendix

A.1 Hyperparameters of TabNet

Table A.1: Irrelevant hyperparameters of the TabNet model their model default hyperparameter values and their recommended range according to [11]. "−" indicates that no model default value or recommended value was given.

Model	Default value	Recommended range
cat_idx	[ ]	−
cat_dims	[ ]	−
cat_emb_dim	1	−
$clip\_value$	−	−
$group\_features$	−	−
$n\_shared\_decoder$	1	$Only for PreTrainer$
$n\_indep\_decoder$	1	$Only for PreTrainer$
$callbacks$	−	−
$pretraining\_ratio$	−	$Only for PreTrainer$

Table A.2: Relevant hyperparameters of the TabNet model their model default hyperparameter values and their recommended range according to [11]. "–" indicates that no model default value or recommended value was given.

Model	Default value	Recommended range
n_d	8	$8 - 64$
n_a	8	$n_d = n_a$
n_steps	3	$3 - 10$
γ	1.3	$1.0 - 2.0$
$n\_independent$	2	$1 - 5$
n_shared	2	$1 - 5$
$epsilon$	$1e - 15$	$Not\ to\ be\ changed$
$seed$	0	–
$momentum$	0.02	$0.01 - 0.4$
$lambda\_sparse$	$1e - 3$	–
$optimizer\_fn$	$torch.optim.Adam$	–
$optimizer\_params$	$lr = 2e - 2$	–
$scheduler\_fn$	–	–
$scheduler\_params$	–	–
$model\_name$	$'DreamQuarkTabNet'$	–
$verbose$	1	–
$device\_name$	$auto$	–
$mask\_type$	$sparsemax$	$entmax, sparsemax$
$eval\_set$	–	–
$eval\_name$	–	–
$eval\_metric$	–	–
max_epochs	200	–
$patience$	10	–
$weights$	0	–
$loss\_fn$	$Cross\ Entropy$	–
$batch\_size$	1024	$1 - 10\% of\ data$
$virtual\_batch\_size$	128	$divide\ batch\_size$
$num\_workers$	0	–
$drop\_last$	$False$	–
$warm\_start$	$False$	–
$compute\_importance$	$True$	–

A.2 Feature list

List of all features, divided into their categories, used in the analysis. Values in brackets represent multiple features. Each value in the bracket corresponds to one feature.

- **Event-shape:** $thrust$
- **Vertex reconstruction:** $track(1,2,3)\_3prong\_dr$, $tau\_3prong\_distance$, $tau\_3prong\_chiProb$
- **Kinematics:** $tau\_3prong\_E\_CMS$, $tau\_3prong\_pt\_CMS$, $visibleEnergyOfEventCMS$, $missingM2$, $missingMomentumOfEventCMS$, $missingMomentumOfEventCMS\_theta$
- γ **and** π^0 **rejection:** $nPiOs\_(1,3)proong$, $nPhotons\_(1,3)prong$, $photonsE\_(1,3)prong$, $nLoosePhotons\_(1,3)prong$, $loosePhotonE\_(1,3)prong$, $nPhotonsMVA\_(1,3)prong$, $photonMVAE\_(1,3)prong$, $nLoosePhotonsMVA\_(1,3)prong$, $loosePhotonMVAE\_(1,3)prong$

A.3 Logloss of the NoScaler Model

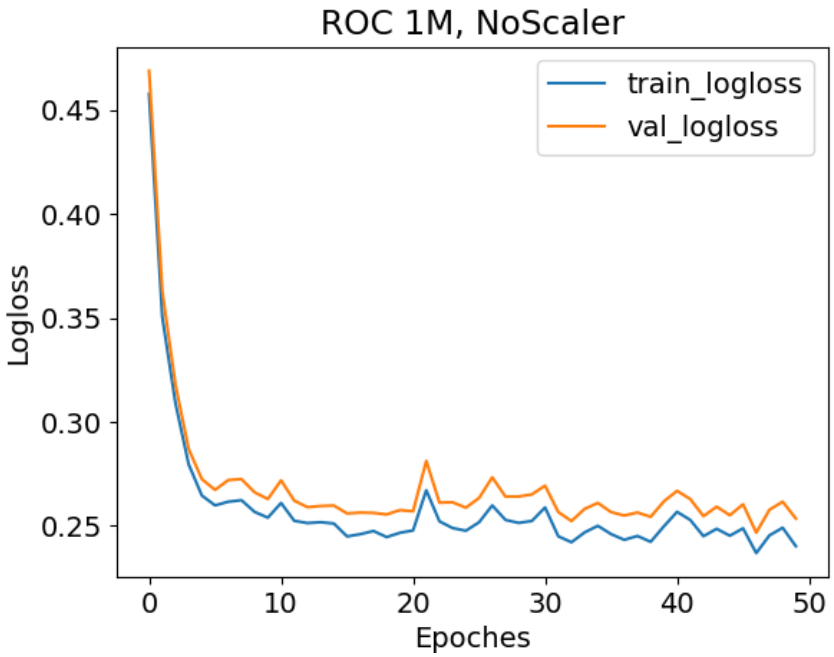
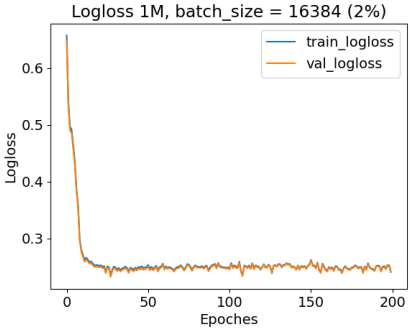
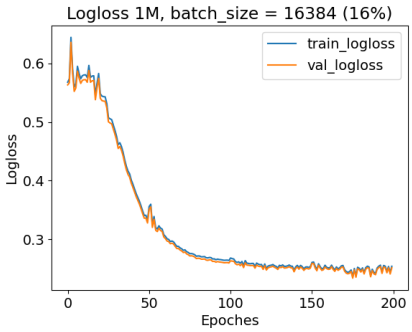


Figure A.1: Loglosses as in figure 4.1. The data was not scaled.

A.4 Logloss of *batch_size* Optimization



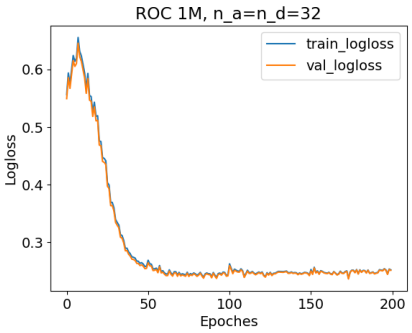
(a) Logloss as in figure 4.1. The model trained on 1M events, 200 epochs and $batch\_size = 16384$.



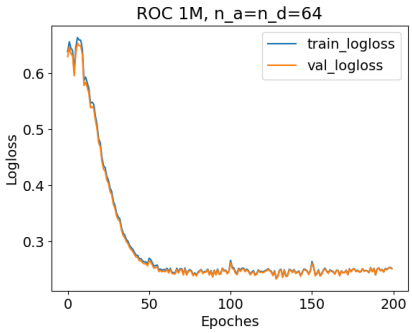
(b)

Figure A.2: Logloss as in figure 4.1. The model trained on 1M events, 200 epochs and $batch\_size$ = (a) 16384, (b)131072.

A.5 Logloss of *n_da* Optimization



(a)



(b)

Figure A.3: Logloss as in figure 4.1. The model trained on 1M events, 200 epochs, $batch\_size = 65536$ and (a) $n\_da = 64$, (b) $n\_da = 32$

A.6 Hyperparameter Values of automated hyperparameter optimization

Table A.3: Hyperparameter, maximum epochs and validation AUC of trial 4.

Trial 4:	
Hyperparameter	**Value**
n_da	40
n_steps	4
gamma	1.9
n_independent	5
n_shared	4
momentum	0.04
lambda_sparse	0.0001631418443918974
learning_rate	0.025246159579514208
batch_size	16384
virtual_batch_size	512
gamma_scheduler	0.74
max_epochs	238
val_auc	0.9542854

Table A.4: Hyperparameter, maximum epochs and validation AUC of trial 16.

Trial 16:	
Hyperparameter	**Value**
n_da	30
n_steps	8
gamma	1.6
n_independent	5
n_shared	5
momentum	0.05
lambda_sparse	0.00028398066268405317
learning_rate	0.04976252205401964
batch_size	65536
virtual_batch_size	2048
gamma_scheduler	0.99
max_epochs	402
val_auc	0.9536834

Table A.5: Hyperparameter, maximum epochs and validation AUC of trial 21.

Trial 21:	
Hyperparameter	**Value**
n_da	62
n_steps	7
$gamma$	1.7
$n\_independent$	5
n_shared	5
$momentum$	0.09999999999999999
$lambda\_sparse$	$6.571264692807959e-6$
$learning\_rate$	0.042400355924497146
$batch\_size$	16384
$virtual\_batch\_size$	2048
$gamma\_scheduler$	0.8500000000000001
max_epochs	80
val_auc	0.9542641

A.7 Correlation Matrix

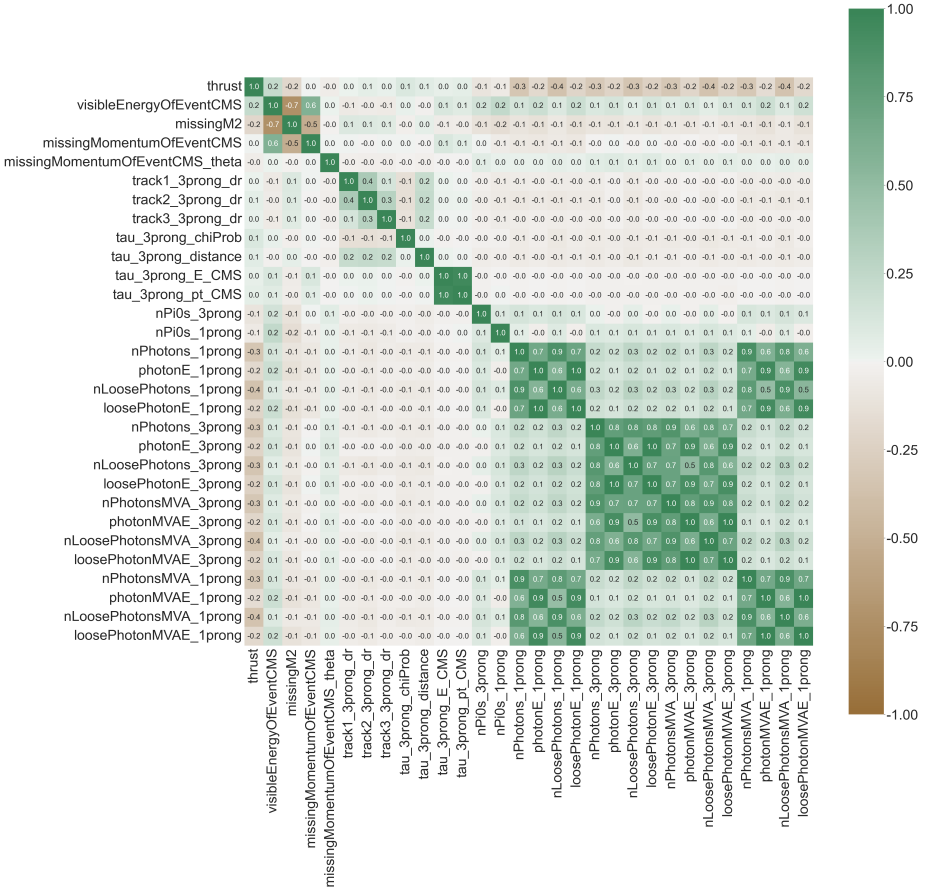


Figure A.4: Correlation of all features used in the TabNet training. Due to readability only one digit after the decimal point is given.

Acronyms

π Pion

AI Artificial Intelligence

ARICH aerogel-based proximity focusing ring imaging Cherenkov system

AUC Area Under the Curve

BDT Boosted Decision Tree

BN Batch Normalization Layer

CDC Central Drift Chamber

CMS Center of Mass System

DL Deep Learning

DT Decision Tree

ECL Electromagnetic Calorimeter

FC Fully Connected Layer

FPR False-Positive-Rate

GLU Gated Linear Unit

IP Interaction Point

K Kaon

KLM K_L^0 and Muon Detector

MC Monte Carlo

ML Machine Learning

MVA Multivariate Analysis

Neural Network Neural Network

NP New Phyiscs

PMTs Photomultiplier Tubes

PWA Partial Wave Analysis

PXD Pixelated Silicon Sensors

QCD Quantum Chromodynamics

ROC Receiver Operating Characteristic

SHAP Shapley Additive Explanations

SM Standard Model

SVD Double-Sided Silicon Strip Sensors

TOP Time-of-Propagation

TPR True-Positive-Rate

VXD Vertex Detector

List of Figures

List of Tables

Bibliography

[1] Brain John Aboze. https://deepchecks.com/a-comprehensive-guide-into-shap-shapley-additive-explanations-values/, 2023. [Online; accessed 19-August-2024].

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

[3] O. Alonso et al. DEPFET active pixel detectors for a future linear e^+e^- collider. *IEEE Trans. Nucl. Sci.*, 60:1457, 2013.

[4] Sercan O. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning, 2020.

[5] Pushpalatha C Bhat and /Fermilab. Advanced analysis methods in particle physics. *Submitted to Ann.Rev.Nucl.Part.Sci.*, 61(1), 10 2010.

[6] Robert Cattral and Franz Oppacher. Poker Hand. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C5KW38.

[7] Yann Coadou. *Boosted Decision Trees*, page 9–58. WORLD SCIENTIFIC, February 2022.

[8] DataScience-ProF. https://medium.com/@TheDataScience-ProF/understanding-log-loss-a-comprehensive-guide-with-code-examples-c79cf5411426, 2024. [Online; accessed 20-July-2024].

[9] DESY. Belle II. https://www.belle2.org/, 2024. [Online; accessed 27-Mai-2024].

[10] DESY. SuperKEKB. https://www.belle2.org/research/superkekb/, 2024. [Online; accessed 27-Mai-2024].

[11] Dreamquark. BWorld Robot Control Software. https://dreamquark-ai.github.io/tabnet/generated_docs/README.html#useful-links, 2019. [Online; accessed 22-Mai-2024].

[12] Kou. E; et al. The belle ii physics book. *Progress of Theoretical and Experimental Physics*, 2019(12), December 2019.

[13] Taisuke Yasuda ; et. al. Sequential attention for feature selection, 2023.

[14] BELLE II EXPERIMENT. `https://belle2.jp/top/#:~:text=The%20Time%2DOf%2DPropagation%20(,kind%20of%20B%20decays%20happened.`, 2016. [Online; accessed 27-Mai-2024].

[15] BELLE II EXPERIMENT. `https://belle2.jp/arich/`, 2016. [Online; accessed 27-Mai-2024].

[16] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, 2022.

[17] Particle Data Group and R L; et. al. Workman. Review of Particle Physics. *Progress of Theoretical and Experimental Physics*, 2022(8):083C01, 08 2022.

[18] Zuzana Gruberová. Tau physics at Belle II. `https://indico.cern.ch/event/949705/contributions/4555573/attachments/2371048/4049442/LP2022_tau_at_BelleII.pdf`, 2022. [Online; accessed 27-Mai-2024].

[19] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks, 2018.

[20] T. Hara; T. Kuhr and Y. Ushiroda. Belle ii coordinate system and guideline of belle ii numbering scheme. 2011.

[21] André F. T. Martins and Ramón Fernandez Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification, 2016.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Jason Brownlee PhD. `https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/`, 2020. [Online; accessed 20-July-2024].

[24] A. Pich. Tau physics. *Adv. Ser. Direct. High Energy Phys.*, 15:453–492, 1998.

[25] L. D. Landau ; E. M. Liftshitz; L. P. Pitaevskii. *Electrodynamics of Continuous Media*. New York: Pergamon Press, 1984.

[26] Andrei Rabusov. *Partial-Wave Analysis of τ-$\to\pi$-π-$\pi+\nu\tau$ at Belle*. PhD thesis, Technische Universität München, 2023.

[27] Scott Lundberg Revision. https://shap-lrjball.readthedocs.io/en/latest/index.html, 2018. [Online; accessed 20-July-2024].

[28] Rakesh Sukumar. https://medium.com/analytics-vidhya/shap-part-1-an-introduction-to-shap-58aa087a460c, 2020. [Online; accessed 20-July-2024].

[29] DATAtab Team. https://datatab.de/tutorial/roc-kurve, 2024. [Online; accessed 03-June-2024].

Acknowledgements

First of all I want to thank Prof. Dr. Stephan Paul for the opportunity to work on such an interesting topic. I also want to express my gratitude towards my advisor Dr. Stefan Wallner for guiding me through this journey and constantly giving constructive feedback on the work that was done. I learned a lot under your guidance. Also I want to thank Dr. Juliane Simoneit for helping me out in difficult times with good talks and her expertise. Many thanks also to Dino Pehratovic for helping out during late hours. Lastly I want to thank my father for the discussions and different view on the topic and my mother for enduring the (in her view boring) discussions during breakfast, lunch and dinner. This also applies to my siblings.