Karlsruhe Institute of Technology

ETP-KA/2017-30

Continuum Suppression with Deep Learning techniques for the Belle II Experiment

Dennis Weyland

Master Thesis

November 2, 2017

Institut of Experimental Particle Physics (ETP)

Advisor: Prof. Dr. G. Quast
Coadvisor: PD Dr. A. Meyer

Editing time: November 2, 2016 – November 2, 2017

Contents

1. Introduction

The human mind always tries to understand what is beyond its knowledge. To understand the universe and the process of its creation to this point, particle physics examines the tiniest building blocks of matter by conducting experiments with ever-increasing accuracies. The Standard Model of Particle Physics (SM) is a theory that describes nearly all phenomena with remarkable accuracy. It explains three of the four fundamental interactions and all known particles. The SM has been developed since the 1960s and almost all tests in high energy physics are in agreement with this theory.

The last particle predicted by the Standard Model, the Higgs Boson, was discovered at the Large Hadron Collider (LHC) in 2012 [1, 2]. However, astrophysical and cosmological measurements like the galactic rotation curves or gravity lensing effects are indications for the existence of dark matter, for which the SM does not provide a satisfying explanation. The SM is thus not an encompassing theory, but there are physical phenomena.

B-factories are experiments that investigate the decay of B mesons by collecting a large amount of data by operating on the energy of the $\Upsilon(4S)$-resonance that decays almost exclusively into two B mesons. This makes it possible to search for deviations of the SM by searching for new exotic decays and testing already known branching fractions more accurately than other experiments. B-factories such as Babar and Belle discovered CP violation in the B meson system and thus verified the theory of Makoto Kobayashi and Toshihide Maskawa, who were awarded the Nobel Prize for their theory in 2008 [3].

The Belle II experiment, which is currently under construction, is an upgrade of Belle and thus the next generation of B-factories. By accumulating 50 times more data than Belle, the Belle II experiment will open up new possibilities for analyses, that were not possible before. Besides the desired collision containing a B meson pair (signal), Belle II will also create many other events, in which this is not the case. Those events are called continuum and have to be suppressed in order to analyze the B meson decays.

For achieving a separation between continuum and signal, multivariate analysis (MVA) methods are used to classify the event as continuum or signal. The Belle II Software Framework basf2 implements such a continuum suppression by using engineered features that describe the shapes of the underlying event. Those engineered features were developed empirically using the understanding of the difference of the event shape between signal and continuum.

Recent developments in machine learning have changed the way classification processes can be carried out with huge datasets. In so-called Deep Learning techniques [4], a classifier is able to learn its own representation of the event, instead of relying on engineered features. In Belle II, Deep Learning was already successfully applied for Flavour Tagging [5]. Besides the valuable experience gained from the Deep Flavour Tagger, Deep Learning techniques are an active field of research with many innovations, which could further benefit the Belle II experiment.

In this thesis, a "deep continuum suppression" using various Deep Learning techniques is developed and compared to the traditional, Belle-inspired approaches, which were already implemented in basf2. Instead of just relying on the traditional engineered features, new features are used that describe single reconstructed tracks and clusters in the event. Besides the improvement in continuum suppression, this thesis also provides new insights in Deep Learning techniques, which were not used in the Belle II experiment before and which further increase the understanding of these technologies.

In Chapter 2 a brief introduction to the Belle II experiment is provided. Understanding the continuum is crucial, when building a continuum suppression. The continuum as well as the engineered features are explained in Chapter 3. In Chapter 4 the used MVA methods including the Deep Learning techniques are described. To compare the Deep Learning techniques with the traditional technique, Monte Carlo (MC) generated events are needed to extract the features which can be used by the classifiers. Chapter 5 introduces the used MC generated dataset and explains all features which were fed into the classifiers. Also, the choice of so-called hyper-parameters, which are crucial to the performance of a classifier is discussed. In the second half of the chapter, the classifiers using Deep Learning techniques are then compared to the traditional technique. The results are discussed at the end of the chapter. In Chapter 6 the results are summarized and an outlook on the "deep continuum suppression" and its Deep Learning techniques is provided.

2. The Belle II Experiment

The Belle II experiment at the SuperKEKB accelerator is designed to examine electron-positron collisions. The experiment is located at the High Energy Accelerator Research Organization (KEK) in Tsukuba, Japan and is currently under construction. Like its predecessor, the Belle experiment, it is a so-called B-factory, that operates mostly at a center-of-mass energy that corresponds to the mass of the $\Upsilon(4S)$-resonance to produce B meson pairs at a high luminosity. Although the Belle II experiment is an upgrade to the Belle experiment, many parts were built completely new based on the experience gained from the previous experiment. The Belle II Collaboration is responsible for designing and operating the experiment, as well as analyzing the resulting data.

To understand the underlying motivation of this experiment, Section 2.1 gives a brief introduction to the physical motivations that warrant an upgrade. A detailed overview of important physical contributions of past B-factories is described in Ref. [6]. The accelerator SuperKEKB and the detector Belle II are described in Section 2.2 and Section 2.3, while a more in-depth description of both can be found in Ref. [7]. This thesis uses features that are extracted directly after tracking and cluster reconstruction. Therefore, both processes are introduced in Section 2.4. At last, the software framework basf2, which was built for the Belle II experiment from scratch and was used and expanded in this thesis, is described in Section 2.5.

2.1. Motivation

In 2008, the Nobel Prize in physics was awarded to Makoto Kobayashi and Toshihide Maskawa for their prediction of CP violation [3]. This was achieved by the verification of Belle's [8, 9] and BaBar's [10] observation of the CP violation in the interference between mixing and decay in $B^0 \to J/\psi K^0_S$ decays.

In the same year 2008, the Belle II Collaboration was formed, as proposed in a Letter of Intent from 2004 [11]. With a 40 times higher luminosity, Belle II will carry on the legacy of B-factories and accumulate around 50 times more data than Belle [12]. Using such a bigger dataset provides access for more accurate and new analyses. The physics program of Belle II is to search for new physics in rare decays and in CP violation, as well as to further examine B anomalies and exotic states. Charmless 2-body B meson decays are an example of rare SM processes in which possible contributions from new physics could be large enough to be observed in Belle II. In this case $B \to K\pi$ decays like $B^0 \to K^0_S\pi^0$ are studied. A more detailed overview of the physics motivation can be found in Ref. [7].

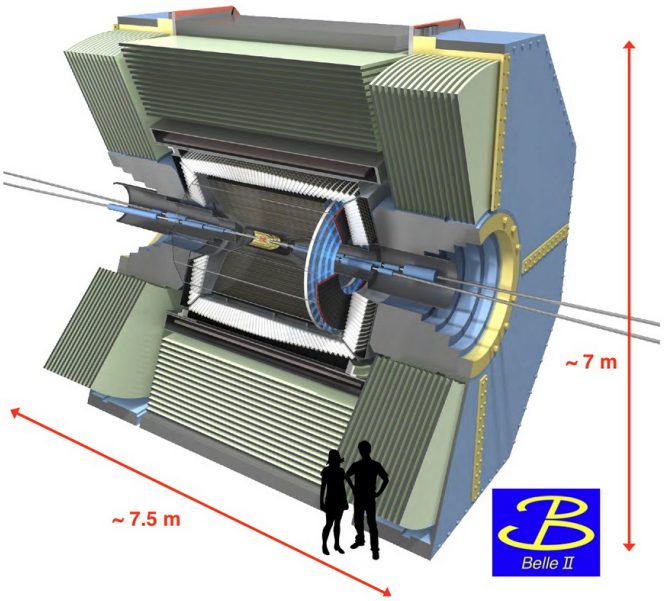

Figure 2.1.: The Belle II detector [15].

2.2. Accelerator

SuperKEKB is an upgrade to KEKB, which was the accelerator with the Belle experiment, and is built in the same tunnel as its predecessor. It is an e^-e^+-collider with an energy of 4 GeV in the low energy positron ring (LER) and an energy of 7 GeV in the high energy electron ring (HER). The accelerator is designed to run mainly at the energy of 10.58 GeV which corresponds to the mass of the $\Upsilon(4S)$-resonance [13]. Because of the asymmetry in beam energy, there is a boost of the center-of-mass system of the $\Upsilon(4S)$, which makes it possible to measure the time difference in the lifetime of the two resulting B mesons, which is necessary for the measurement of time dependent CP violation.

The huge increase in luminosity, which SuperKEKB is designed to achieve is due a new "nano-beam" scheme [14]. In this scheme, the cross section of the two beams is reduced, which makes an e^-e^+ collision more likely and increases the luminosity by a factor of 20. The remaining factor of two is realized by increasing the number of particles per beam.

2.3. Detector

The Belle II detector is presented in Figure 2.1. It is built on similar principles like its predecessor, while introducing some modifications due to new concepts for improving the quality of measurement. Below, a brief introduction of the major detector components is given. A detailed explanation of the components can be found in Ref. [7].

Magnetic Field

A superconducting solenoid provides a magnetic field of 1.5 T, which is used to determine the momentum and charge of charged particles. To achieve the homogeneity of the magnetic field, an iron yoke is used. The solenoid is located around the ECL.

Vertex Detector (VXD)

The VerteX Detector consists of a two-layer silicon PiXel Detector (PXD) and a four layer silicon strip detector (SVD). While the PXD is based in the DEPFET technology [7], the SVD is built of silicon strips with a double-sided readout. The PXD is the innermost detector component, closest to the interaction point, while the SVD is located around the PXD. Because the VXD can measure precise space points, tracks can be reconstructed with precise impact parameter information.

Central Drift Chamber (CDC)

The CDC consists of a cell-wire structure, filled with a Helium-Methane mixture. Some of its wires are arranged parallel to the beam axis and some are slightly tilted to obtain information from the particles in the direction of the beam axis. The information of the CDC is used for reconstructing tracks and extracting particle ID information from the specific energy loss due to ionization.

Electromagnetic Calorimeter (ECL)

The ECL consists of many scintillator crystals and is used for the detection of electromagnetic clusters. It is located in the barrel region and at the end cap region. Its information is used to determine the energy deposition, angular information and the PID of the incoming particles.

Particle Identification

There are multiple systems for particle identification. Their main purpose is to provide information for discriminating kaons and pions. The Time-Of-Propagation counter (TOP) is located at the outer wall of the CDC and uses Cerenkov light for detection. In the end-cap region, there is the Aerogel Ring-Imaging Cerenkov detector (ARICH), which uses the same detection principle.

K_L^0 and Muon Detector (KLM)

The KLM is the outermost detector component and consists of alternating sensor layers and iron plates. A good distinction between hadrons and muons can be achieved by determining whether there is a charged track that matches the particle measured by the KLM. These tracks belong to muons.

2.4. Tracks and Cluster Reconstruction

A particle is traversing the detector and creates hits in the CDC and energy depositions in the ECL. Many hits (energy depositions) are combined to create a track (cluster) object.

In the tracking algorithm, the trajectory of the particle is reconstructed from the hits measured in the detector along the trajectory of the particle. Because there are multiple hits originating from all sorts of particles, track finding is a process of pattern recognition. Various algorithms are used to reconstruct the tracks correctly. A reconstructed track consists of information about the position and momentum at the closest point to the interaction point in its trajectory. The curvature of the trajectory indicates the sign of the charge. Additionally, uncertainty information about the reconstruction of the track is available, as well as PID information from the various detector parts.

The ECL detects the particles by their energy deposition. Therefore, momentum information about the particle is available at the time of entering the ECL, if a particle hypothesis is applied. If no track is matched to the cluster the position resolution is very poor and therefore it is only considered, if the particle is detected in the barrel or endcap region. In addition to momentum and region, cluster objects also consist of information about the uncertainty, timing and shape of the reconstructed cluster.

2.5. Software Framework basf2

The Belle II Analysis Software Framework basf2 was written completely new for the Belle II experiment. It is designed to be used for online tasks (such as data acquisition and high level trigger) and offline purposes (such as reconstruction and analysis) [16]. While the biggest part of its code is written in C++, there is a user interface which can be accessed by the programming language Python.

The software framework also features a multivariate analysis (MVA) package. This package provides access to a wide variety of MVA methods, as well as the possibility to be compatible with every classifier, which possesses a python interface.

During the work of this thesis, I implemented several new features in the analysis package of basf2. For the MVA package, I implemented new classifier interfaces, several examples and further improvements regarding the performance of the package.

3. Continuum Suppression

Performing an analysis is largely associated with the detection of so-called background. Background is anything that does not contribute to the analysis and should be discarded. When dealing with measured data, there is no absolute proof, which parts of the data is background. Therefore, the MVA methods (see Chapter 4) use a set of input features to predict whether the event is a background event that should be discarded or whether it is a desired signal event containing a $\Upsilon(4S)$-resonance. This chapter introduces continuum as an event based background component and explains the engineered features, which are used by the traditional continuum suppression.

The chapter starts with introducing continuum as a main background component in many analyses in Section 3.1. For understanding the differences between continuum and signal events, the event shapes are explained in Section 3.2. In Sections 3.2.1 – 3.2.3, the engineered features used for continuum suppression are introduced. Finally, a short conclusion of these features is given in Section 3.2.4.

3.1. Continuum

The goal of SuperKEKB is to produce as many events at the $\Upsilon(4S)$-resonance (signal events) as possible. The $\Upsilon(4S)$ has an invariant mass of 10.58 GeV [13] and consists of a bound $b\bar{b}$ pair, which will hadronize to a pair of B mesons. But not every e^-e^+ collision results in the two desired B mesons. Events are called continuum if they do not contain a B meson pair. For example the cross section of the Bhabha scattering process $e^-e^+ \to e^-e^+$ is around 300 times higher than that of the $\Upsilon(4S)$-resonance [17]. There are other leptonic decay products like $\mu^-\mu^+$ and photons, although they do not possess such a high cross section than the Bhabha scattering. Most of these leptonic events will be cut away by the trigger. Therefore, they can be neglected as a background component in most analyses.

The remaining continuum are quark-antiquark events. In Figure 3.1 the relative cross sections of $e^-e^+ \to q\bar{q}$ are shown. The desired signal event, which leads to two B mesons, occurs only in 22.2 % of the cases. Not shown in Figure 3.1 is $\tau^-\tau^+$. Although those are leptons, they can decay hadronically and are therefore also a relevant background for many analyses. Many analyses in Belle, e.g. $B^0 \to K^0_S\pi^0$, could be vastly improved with a better separation between continuum and signal events [18].

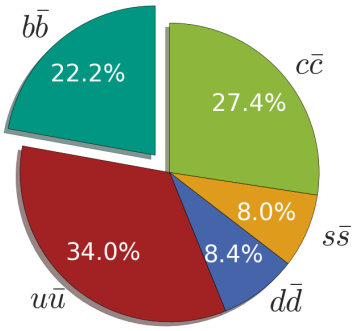

Figure 3.1.: Relative cross sections of $e^-e^+ \to q\bar{q}$ at the invariant mass of the $\Upsilon(4S)$-resonance [17]. Events with $e^-e^+ \to B\overline{B}$ come from the $\Upsilon(4S)$-resonance and are the desired signal category. The rest of the chart is continuum background.

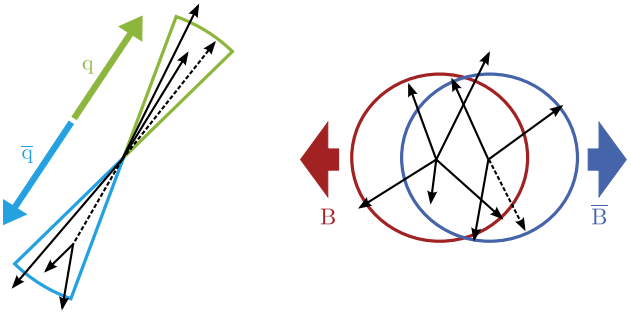

Figure 3.2.: Event shapes for continuum (left) and signal(right) events. Signal events have spherical shape, while light quark pairs have a jet like structure, because they're produced back-to-back. Adapted from [19].

3.2. Event Shapes

The event shapes of continuum and signal events are illustrated in Figure 3.2. In the center-of-mass system the two B mesons from signal events are almost at rest. Because they have spin 0, their decay products have no preferred direction, which results in an isotropic distribution of spherical shape. In continuum events the light quark pairs that are created back-to-back have much more kinetic energy, which almost corresponds to the energy of the accelerator. Therefore, the hadrons produced in the fragmentation do not deviate so much from the flight directions of the quark pairs, resulting in a jet-like structure.

Below, different features are described that characterize the differences in event shapes between signal and continuum events. They are calculated using information from final state particles found in the detector, described in Section 2.3. The final state particles can be combined to a B meson candidate, which represents one of the two B meson originating from an $\Upsilon(4S)$-resonance. The rest of event (ROE) is formed by the remaining final state particles that are not used for forming the B candidate. To visualize the distributions of these features each for signal and background, $B \to K^0_S\pi^0$ is chosen as the decay to combine the B candidate.

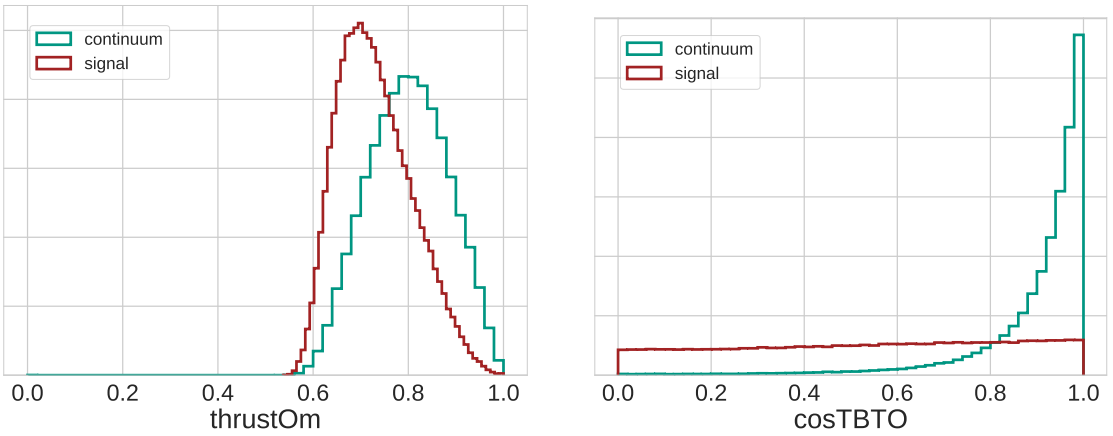

Figure 3.3.: Normalized distributions of the magnitude of the thrust axis of the ROE (thrustOm, left) and the angle between the thrust axis calculated from the B candidate and thrust axis calculated the ROE (cosTBTO, right).

3.2.1. Thrust

The concept of thrust was originally used to quantify jets [20]. The thrust axis is defined as the axis that maximizes the sum of longitudinal momenta of particles. Since the particles are divided into those which are used for reconstructing the B candidate and those in ROE, two different thrust axes can be calculated. Four different thrust related features are used for continuum suppression: the magnitude of each thrust axis; the cosine of the angle between the two thrust axes; and the cosine of angle between the thrust axis of the B candidate and the z-axis, which is the beam axis. Two of these features are shown in Figure 3.3. All features can be found in Appendix A.1.

3.2.2. Cleo Cones

Cleo Cones were introduced by the CLEO Collaboration for continuum suppression [21]. The cones measure the scalar momentum flow around the thrust axis into concentric cones in angular intervals of 10°. The distributions of the first two Cleo Cones are shown in Figure 3.4. All Cleo Cone distributions are shown in Appendix A.2.

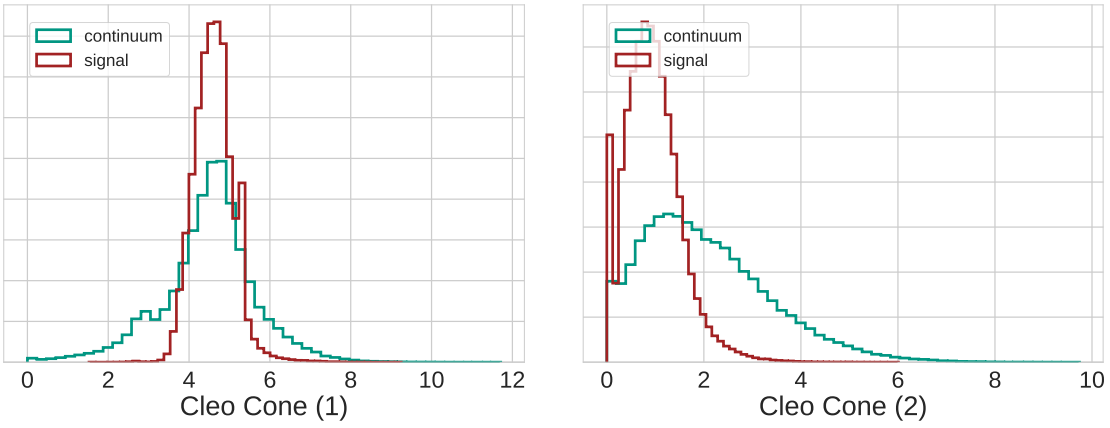

Figure 3.4.: Normalized distributions of the first (left) and second (right) Cleo Cones. Cleo Cones sum up all momenta in 10° cones around the thrust axis.

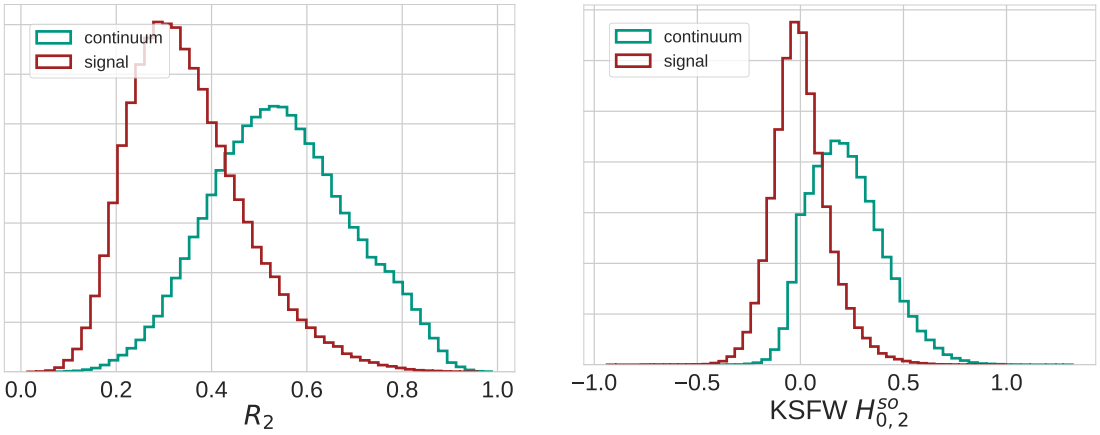

Figure 3.5.: Normalized distributions of the KSFW moments R_2 (left) and $H^{so}_{0,2}$ (right).

3.2.3. Fox-Wolfram Moments

The Fox-Wolfram (FW) moments H_l are defined as

$$H_l = \sum_{i,j}^{N} |\boldsymbol{p_i}||\boldsymbol{p_j}| P_l(\cos(\theta_{ij})) \text{ and } R_l = \frac{H_l}{H_0}, \tag{3.1}$$

where θ_{ij} is the angle between $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ and P_l is the l-th order Legendre polynomial. Those moments were originally introduced to describe event shapes in e^-e^+ annihilations [22]. A very good discriminator is R_2, which is one of the used features in continuum suppression. The feature is shown in Figure 3.5.

A more refined version of the FW moments are the Kakuno-Super-Fox-Wolfram (KSFW) moments $H^g_{c,l}$, which are described in detail in Ref. [6]. The considered particle groups for the double sum are described in the variable g. If one of the sums runs over the particles that decayed from the B candidate and the other sum runs over the particles located in ROE, the variable g is marked as "so". If both sums use the particles in the ROE, the variable g is marked as "oo". The variable c describes whether the considered particles are charged (0), neutral (1) or missing (2). It is only used with the superscript "so". In the end the possible KSFW moments are $H^{so/oo}_{0/1/2,l}$.

Also considered as KSFW moments are the transverse energy E_t, which is the scalar sum of the transverse momenta of each particle and the missing mass squared M^2_m [6]. One of the best discriminating KSFW moments is shown in Figure 3.5. All KSFW moments can be found in Appendix A.3.

In Belle, most of these features were combined into a Fisher Discriminant [23] and then fed in another MVA method [24]. For the Belle II software framework basf2, the state-of-the-art approach prior to this thesis was to insert all features directly into an MVA method.

3.2.4. Conclusion

The features, which are introduced in this Section were specifically built for continuum suppression and were refined over the years. Therefore, it is not surprising that some of these features separate signal and continuum already very well on their own. As a result using just these engineered features as input features in a MVA method results in a very good classifier.

Even with such a high baseline, continuum is still considered as the main background component for many analyses. Also, the engineered features are not built to contain all possibly useful information in the event. Therefore, a new set of features based on tracks and clusters will be compared with the engineered features in Chapter 5.

4. Multivariate Analysis Methods

Multivariate Analysis (MVA) methods are commonly used in particle physics. Most of the time they are used as binary classifiers. In this case a multidimensional feature vector will be projected into a one dimensional test-statistic. This test-statistic is designed to separate signal and background and therefore is used to increase the signal-to-background ratio in an analysis. In contrast to simple cuts on independent features, MVA methods also use information about correlations between the features. Therefore, the resulting test-statistic is in many cases much better. MVA methods have to be trained on an independent dataset to adapt its model specific parameters to a given classification task. Many MVA methods also require so-called hyper-parameters, which are not automatically tuned by the training and have to be chosen by the user.

This section starts by introducing Boosted Decision Trees (BDT) and Artificial Neural Networks (ANN) in Section 4.1 and Section 4.2. The concept of Deep Learning, which is a very active field of research in and outside of physics, is explained in Section 4.3. Relation Networks and Adversarial Networks as recent techniques developed in Deep Learning are introduced in Section 4.3.1 and Section 4.3.2. Using such techniques requires the handling of many hyper-parameters. So-called Black Box Optimization is an approach that can handle this automatically and is explained in Section 4.4.

4.1. Boosted Decision Trees (BDT)

A decision tree divides the feature vector into distinct rectangular regions to separate signal and background. In Figure 4.1 the schematic outline of this process is shown. The binary cuts are ordered hierarchically in a tree and are applied consecutively. In the end, the feature space is divided into regions with different sizes. Each region now possesses its own signal fraction, which is used as a classifier output, if the data point is in the corresponding region. The cuts of a decision tree are referred to as nodes, while the regions are referred to as leafs.

A single decision tree does not possess a good generalization capability. In other words, it is very prone to "remember" statistical features of the particular dataset used for the training and performs much worse on an independent dataset. This so-called over-training can be prevented by limiting the depth of a tree, which is the number of consecutive cuts. Such a limited tree is a so-called weak-learner and divides signal and background only poorly.

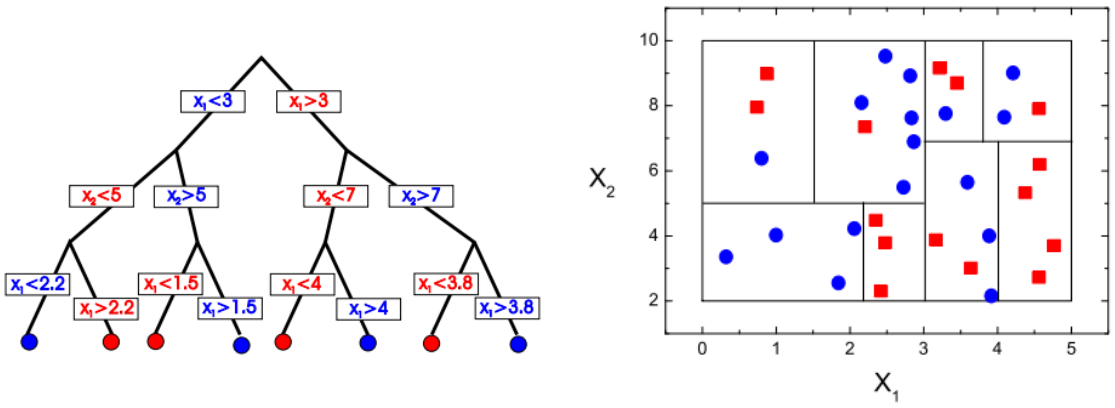

Figure 4.1.: Schematic outline of a single Decision Tree (left) and its regions inside a two-dimensional feature vector resulting from the cuts in the tree (right). Taken from [25]

While a single weak-learner does not perform well on classification tasks, many weak-learners combined can create a good classifier that is resistant to over-training. A so-called Boosting algorithm, like Gradient Boosting [26], assigns each event from the training dataset a weight to train a weak learner. Weights of wrongly (correctly) classified events will be increased (decreased) and used for training a new weak-learner. This process will be repeated a few hundred times. In the end the classifier output is defined by the weighted sum of the outputs of the individual weak-learners. The boosting algorithms build a "forest" of weak-learners and belong to the category of ensemble methods.

In the Belle II Software Framework basf2, BDTs are the most often used classifiers. The standard boosting algorithm is FastBDT [27, 28]. FastBDT also supports boosting to uniformity, which is a concept that is explained in Section 4.3.2. For the comparisons in this thesis, FastBDT will be used as a representative for BDT classifiers.

4.2. Artificial Neural Networks

Artificial Neural Networks are inspired by neural biological processing systems. One of the predecessors was the Rosenblatt Perceptron [29]. A basic ANN is shown in Figure 4.2. This type of Neural Network is called a Multilayer Perceptron (MLP) and consists of an input layer, which receive the input features, a hidden layer and an output layer that is the classifier output. While the dimension of the input and output layer is specified by the classification task, the size of the hidden layer can be adjusted. Every neuron consists of weighted sums from the outputs from the previous layer. Neurons in hidden and output layers have an activation function $A(x)$, which transforms the output of each neuron. Also, every neuron can have a bias b, which serves as an offset for the activation function. The output of a Neural Network with a single hidden layer is calculated as

$$y = A_2\left(\sum h_j \cdot w_j^{(2)} + b\right) \text{ , where } h_j = A_1\left(\sum x_i \cdot w_{ij}^{(1)} + b_j\right), \tag{4.1}$$

and i, j are the indices over the input, hidden neurons.

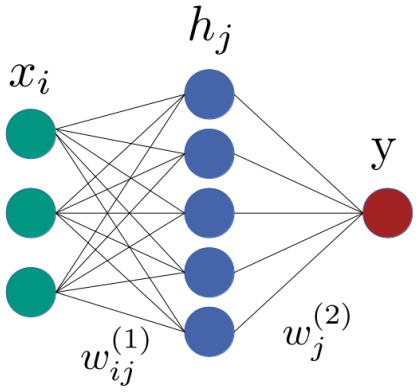

Figure 4.2.: Schematic outline of a Multilayer Perceptron (MLP). The network consists of a layer of input neurons x_i, a layer of hidden neurons h_j, a single output neuron y and weights $w_{ij}^{(1)}/w_j^{(2)}$, which are the connections between the layers. Biases and activation functions are not shown.

To be able to solve nonlinear classification problems like the XOR-Problem the activation functions from the MLP has to be nonlinear. Otherwise, the MLP is only a linear classifier and therefore not able to solve such problems [30].

To train a MLP a loss function is required, which compares the predicted result with the truth values. The gradient of the error defined by the loss function is then backpropagated through the network and the weights are updated by their gradients. By updating the weights the training tries to minimize the loss function. Using this method a Neural Network can approximate a vast number of functions and is not limited to binary classification tasks.

4.3. Deep Learning

A three-layer MLP like the one shown in Section 4.2 consists of only one hidden layer. However, one can increase the capabilities of a MLP by adding many more hidden layers. Such approaches are becoming more popular due to improvements in the available hardware for parallel computation, significantly reducing the time necessary for the training. Besides the increase in depth, there is also a tendency for bigger input feature vectors, e.g. when dealing with images [31]. The goal is to extract the required information from the raw dataset, instead of relying on engineered features, which are only representations of the raw dataset. Because every layer in a Neural Network is a representation of the input features, the number of neurons per layer in relation to a classical Neural Network has to increase as well. Using more hidden layers as well as using more neurons per layer is the foundation of Deep Learning. A schematic outline of a Deep Neural Network is shown in Figure 4.3.

In Deep Learning the underlying statistical models are much more complex than in a traditional Neural Network. Therefore, the need for regularization techniques to prevent over-training is much more important. Two very popular methods for regularization are Dropout [32] and Weight Sharing.

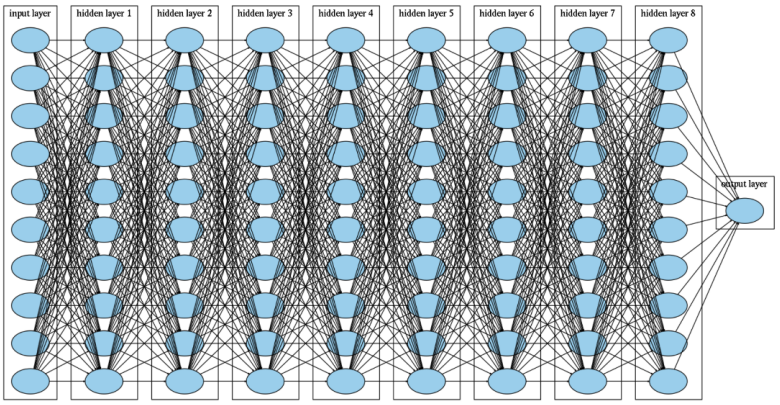

Figure 4.3.: Schematic outline of a Deep Neural Network. Taken from [33]

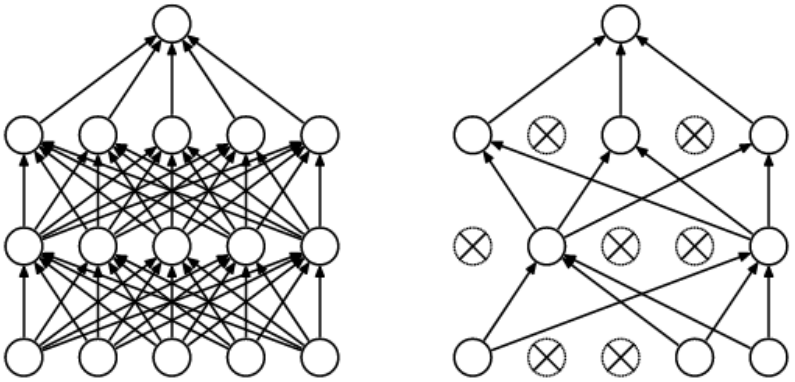

Figure 4.4.: Neural Network before (left) and after (right) applying Dropout. Taken from [32]

During training, Dropout will randomly select neurons and set their activation to zero. As shown in Figure 4.4, the number of weights used for training is reduced significantly. The selection of dropped out neurons is always changing. In the end every weight of the fully connected layer is scaled and no dropout is used for predicting the results.

Weight Sharing is another approach for regularization. Using the same weights in different parts of the network greatly decrease the number of weights and therefore complexity. This loss of complexity does not result in a loss of classification quality because the exact location of some features are unknown. The idea is to search for the same feature in different regions of the representation. The most popular approach to shared weights are Convolution Layers [34]. Another example of Weight Sharing is explained in Section 4.3.1.

Using Deep Learning techniques also requires a fitting software library. TensorFlow [35] calculates symbolic operations using data flow graphs. The graphs are designed by the user on a mathematical operations level. For this thesis, I implemented the TensorFlow interface in basf2 with and without Keras [36], which serves as a front end to TensorFlow and other libraries. In Keras the Neural Network is designed on a layer base, while retaining the freedom of "mixing in" TensorFlow code.

While in basf2 both approaches are supported, I recommend using Keras with TensorFlow as back end. Keras has many convenient functions to use for training Deep Neural Networks, which makes the code of designing such networks much shorter and more accessible. Therefore, sharing Neural Networks inside the Belle II-Collaboration becomes more transparent and easier to understand. However, for developing new types of Deep Learning techniques it is still required to write code at a TensorFlow level.

Deep Learning is already used for FlavourTagging in Belle II [5]. The objective of this thesis was to apply Deep Learning for continuum suppression and to further increase the understanding and knowledge of Deep Learning techniques and how they are useful for the Belle II-Collaboration. In the following sections, specific techniques for Deep Neural Networks, which are used in this thesis are explained. Relation Networks in Section 4.3.1 use the concept of sharing weights, while Adversarial Networks in Section 4.3.2 use the concept of discriminators.

4.3.1. Relation Networks

Relation Networks use the concept of Weight Sharing by comparing two sets of similar features pairwise [37]. A schematic outline of a Relation Network is shown in Figure 4.5. For the purpose of comparing subsets of input features, the input features have to be divided into several groups. The groups should have a representative meaning, e.g. coordinates of a particle. In this case, every feature that describe a single coordinate of a specific particle in the event is grouped together with other features that contain information about the same particle.

The groups are compared pairwise with an internal MLP shown as a black box in Figure 4.5. The shared weight approach is the key aspect of this step. A Multilayer Perceptron with enough neurons and hidden layers could learn the same functionality as the Relation Network. However, for every permutation, the MLP use a new set of parameters. For the example described above, this means that while particle A is still compared to particle B and particle B is still compared to particle C, those comparisons will be done differently, which is not reasonable in most cases. Using a new set of parameters for every permutation also hugely increases the number of parameters, which more likely leads to over-training or to no convergence at all. Therefore, Relation Networks are very good for handling large numbers of input features, which can be understood as groups, while retaining small complexity and preventing over-training.

The output is now determined by the number of permutations and the extracted features per combination. In Figure 4.5 there are three permutations resulting from three groups and two features per combination. This results in a total of six output features. However, using more groups and features might result in a vector, which is too big for feeding into another Neural Network. In this case Global Average Pooling is performed, which averages each feature over every permutation. This process is shown in Figure 4.5. Output features are marked with different colors and are reduced into one neuron per feature in the next layer. The remaining output can now be fed into a MLP, which results in the classifier output.

There is also one optional extension for Relation Networks. In this case an additional input vector provides additional features event-wise to every comparison for context. These features only change event-wise and are identical for every permutation. This idea was used to create a Relation Network, which can compare specific objects depending on a given question [37]. Those modified Relation Networks can be used to feed in additional features, which can not be divided into groups.

Relation Networks are a very recent development and I implemented them in basf2 for this thesis. In Section 5.5 the results of applying both types of Relation Networks for continuum suppression are shown.

4.3.2. Adversarial Networks

Using Adversarial Networks as discriminators in a joint training was first introduced in Generative Adversarial Networks (GAN, [39]). Those Networks are divided into two parts: The Generative Network (GN) which tries to create data out of random numbers and the Adversarial Network (AN), which tries to predict if the data is "real" or from the GN. Because of that, the AN is called a discriminator. The better the discriminator is at separating "real" and generated data during training, the more the GN will be punished in its loss function. This way, the GN is trained against the AN and tries to create data which is indistinguishable for the AN.

The concept of AN was also adapted for classification purposes [40]. Depending on the input features, the classifier might be biased towards a certain quantity, which is not allowed to be correlated to the classifier output. In most analyses it is required that the classifier output does not affect such a quantity at all. Therefore, the AN has to learn to approximate this quantity during training and to punish the classifier, if it is able to learn the shape of said quantity.

Such a Neural Network is shown in Figure 4.6. In this case there is a regular Neural Network with one red classification output. An AN is built to approximate the shape of a specific quantity using the classification output. This is done by trying to predict the parameters of a Gaussian Mixture Model [38]. In Figure 4.6 the Gaussian Mixture Model is made of two parameter sets, each representing a Gaussian distribution, which will be combined to approximate the shape of the quantity. The resulting shape will then be compared with the specific quantity during training and the classifier will be punished by the approximation capability of the AN.

The classifier is not limited to only one Adversarial Network. Multiple ANs can be built on the same output to control different quantities. In Section 5.6 each quantity is split into its background and signal distribution and therefore requires two ANs. The classifier is then compared to other classifiers, which avoid the bias on the quantities by dropping features that are correlated to them.

There is also a BDT algorithm called uBoost which provides a similar functionality [41]. Here, a metric is calculated which describes the change in shape of the quantity on different classifier output cuts. This metric will be used as an additional loss function in the boosting algorithm. In principle this metric fulfills the same role as the AN of a Neural Network classifier. The technique is also available in FastBDT and can produce similar results than the Adversarial Networks.

4.4. Bayesian Optimization

Bayesian Optimization is a technique for minimizing black box functions, where the derivatives are unknown. In contrast to other minimization algorithms, Bayesian Optimization is especially useful in dealing with functions where evaluations are quite expensive. The hyper-parameters of a classifier can be seen as the arguments of a multidimensional function that projects the hyper-parameters to a figure of merit, which is a metric for the classification capability. Therefore, the task of tuning hyper-parameters from a classifier is very fitted to be done by Bayesian Optimization [42].

Building a Bayesian Optimization algorithm requires two bits: A prior probability measure and an acquisition function. The prior probability measure is a function, which captures the prior beliefs on the function to minimize. This prior will be updated to a posterior by using data received from calculating the figure of merit. As a prior probability measure, Gaussian Processes, which are described in Ref. [43], are the most common. An acquisition function is chosen to determine which data points will be calculated next, based on the assumptions of the prior. After calculation, the prior is updated and this cycle continues. Therefore, Bayesian Optimization tries to optimize the amount of exploration and exploitation and is suited to solve the Multi Armed Bandit Problem [44].

An example of how a Bayesian Optimization operates is shown in Figure 4.7. Every plot shows an updated version with one additional calculated data point from the plot before. As a figure of merit the Area Under ROC Curve (AUC [45]) is chosen and will also be explained and used in Section 5. Figure 4.7 shows that the acquisition function peaks at the point of lowest lower bound of the model function $\mu(x)$, which is the model acquired by the prior (Step 1-3), but also takes into consideration the number of points already calculated in the local region (Step 4). Therefore, if the next point does not show an improvement in the AUC score the acquisition function will peak in another region.

For most classifiers used in Section 5, Bayesian Optimization was used for optimizing its hyper-parameters. Automatically optimizing every classifier with the exact same conditions improves the credibility of the comparisons and decreases the bias of the author. However, final evaluations of classifiers has to be done on a different dataset as the one used for calculating the figure of merit during optimization. Otherwise, there could be an optimization bias, which could vary across classifiers and reduces the credibility of the results. During this thesis I implemented Bayesian Optimization techniques for hyper-parameter optimization in basf2. For this purpose the software library scikit-optimize [46] was chosen and used for all hyper-parameter optimizations in this thesis.

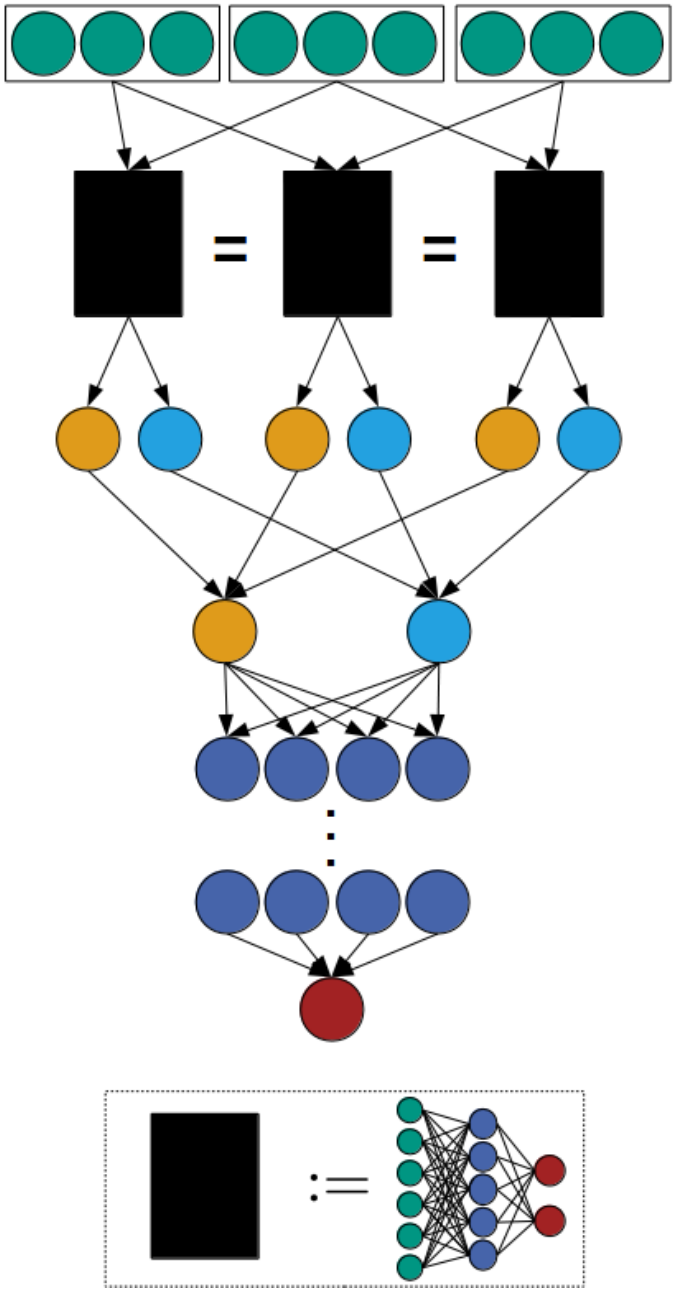

Figure 4.5.: Schematic outline of a Relation Network. From top to bottom, the nine input features are divided into 3 groups. The groups are pairwise compared with the same Neural Network with shared weights. This Neural Network extracts two features for every comparison. For each feature the average over all permutations will be calculated and fed into a Multilayer Perceptron.

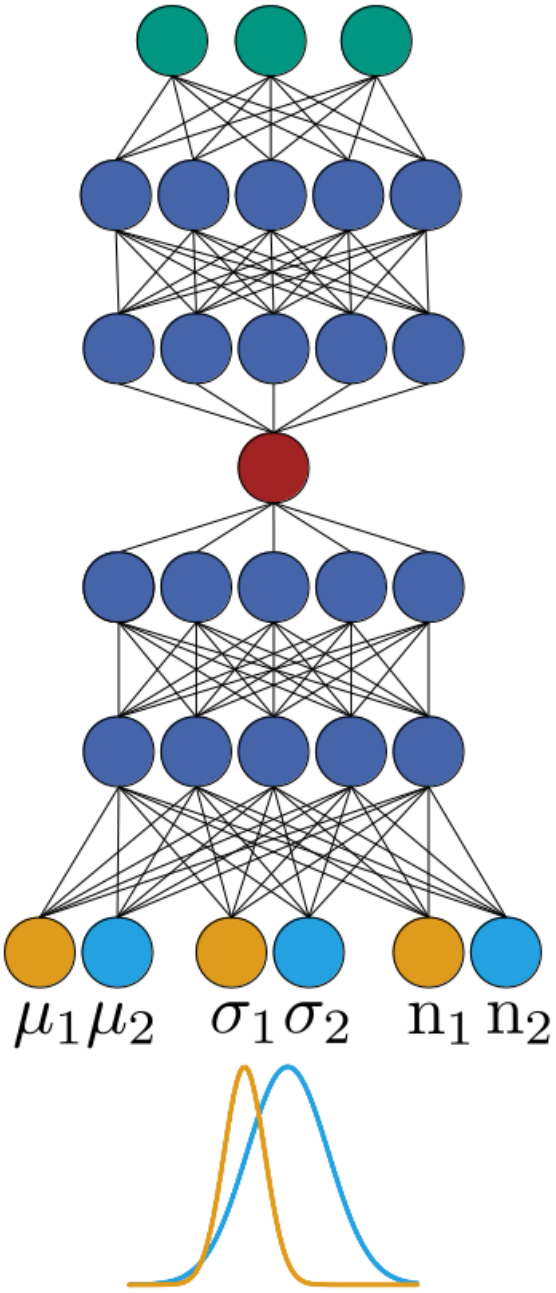

Figure 4.6.: Schematic outline of an Adversarial Network from top to bottom. The structure is a MLP until the red classifier output. Below the classifier output a discriminator network is built, which approximates a quantity with a Gaussian Mixture Model [38], with means μ, standard deviations σ and normalization factors n. During training this approximation also affects the MLP.

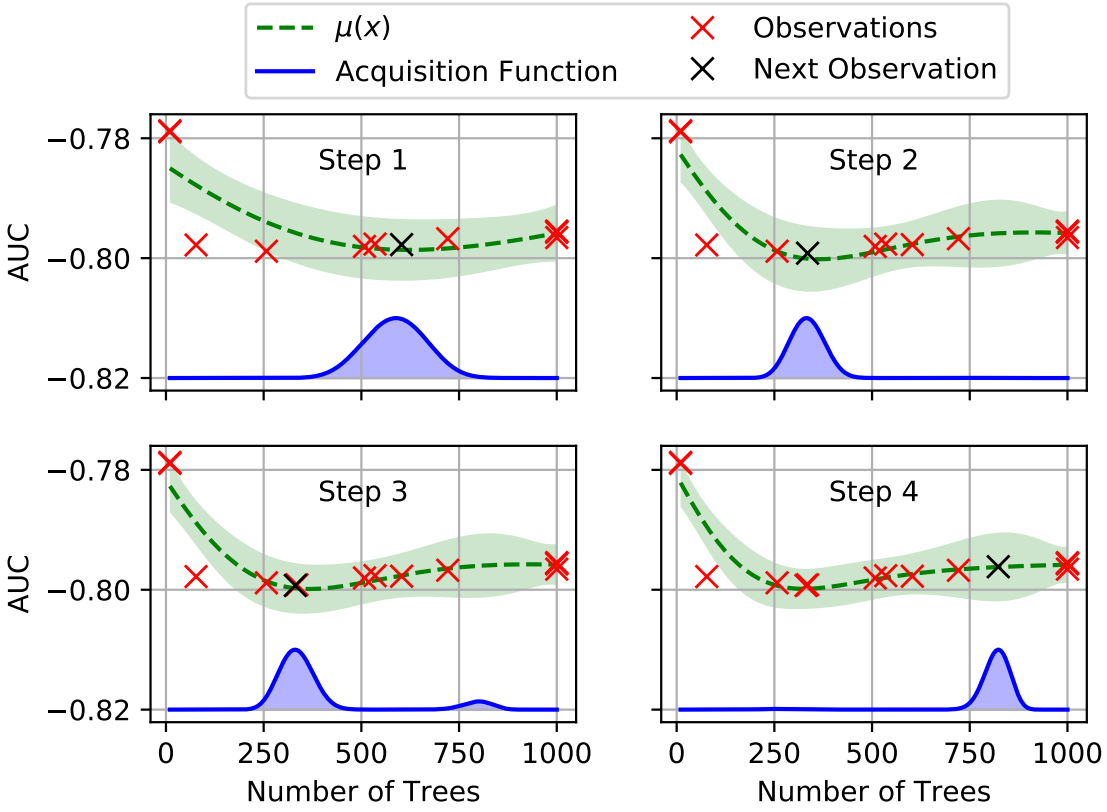

Figure 4.7.: Example of using Bayesian Optimization for tuning the hyper-parameter "Number of Trees" for a BDT classifier. The model function $\mu(x)$ is an approximation of the true hyper-parameter function with respect to the figure of merit AUC [45]. $\mu(x)$ is calculated on the observation points and is updated with every new point. The Acquisition function determine the next location of observation, where the figure of merit will be calculated. The y-axis of the acquisition function is arbitrary.

5. Deep Learning for Continuum Suppression

Continuum suppression classifiers were steadily improved in recent years [24]. This was achieved either by engineering better input features used by the classifiers or by changing the classifiers themselves. The features of the state-of-the-art continuum suppression, which is referred in this thesis as traditional approach, are explained in Section 3.2 and are fully illustrated in Appendix A. Any classifier used for continuum suppression could only be as good as those features that represent the event. Deep Learning techniques that can process much larger input spaces do not have to rely on the representation of the engineered continuum suppression features. Instead, they use information obtained from tracks and clusters and extract the relevant characteristics for continuum suppression themselves. This chapter compares various Deep Learning techniques with traditional classifiers.

The chapter starts by introducing the benchmark dataset in Section 5.1. The features and their composition used for the classifiers will be explained in Section 5.2. Choosing the right hyper-parameters is crucial to the optimal performance of the classifier. Bayesian Optimization can improve the classification result significantly. The whole process of choosing the hyper-parameters is described in Section 5.3. In Section 5.4 comparisons between Deep Neural Networks and traditional classifiers are shown. Section 5.5 demonstrates that Relation Networks (explained in Section 4.3.1) further improve the results of Deep Learning classification. Adversarial Networks (explained in Section 4.3.2) are an elegant technique for avoiding correlations between classifier output and other quantities. Section 5.6 shows how this benefits the continuum suppression. Finally, all results are discussed in Section 5.7.

5.1. Introducing the Benchmark Dataset

Belle II is currently under construction at the time of this thesis and no data measured by Belle II is available. Therefore, the dataset is only based on Monte Carlo (MC) simulation. The Belle II Software Framework basf2 is capable of generating MC simulated events. For the simulation of physical decay chains EvtGen [47] is used. The detector simulation is done with the software package GEANT4 [48]. In the Belle II-Collaboration MC events are generated in so-called Monte Carlo Campaigns, because the generation is very time-consuming and is used for many purposes. This thesis uses simulated events from Monte Carlo Campaign 7 phase 3 [49].

The benchmark dataset contains two types of MC: continuum and signal samples. In the continuum sample $u\bar{u}$, $c\bar{c}$, $d\bar{d}$, $s\bar{s}$ and $\tau^-\tau^+$ events are considered. Within these events the particles hadronize. In the signal sample only $b\bar{b}$ events are considered. One of the two B mesons resulting from this event has to decay as $B^0 \to K^0_S\pi^0$ (charge conjugated included), because this is the decay chosen for reconstructing the B candidate for the continuum suppression. The other B meson decays according to its branching fractions estimated based on PDG information [13]. During the MC Campaign 30 million events were generated in the signal sample and 11 billion events were created for the continuum sample.

The MC sample mentioned above contains simulated events, which are measured inside a simulated detector. However, to perform an analysis, the desired events have to be reconstructed with particle candidates, which are potential particles reconstructed from the simulated detector data. To calculate the engineered features used for the traditional continuum suppression, one reconstructed B candidate and its corresponding ROE (see Section 3.2) is required. For this thesis the desired B candidate is reconstructed using the following decay:

$$B^0 \to K^0_S(\to \pi^+\pi^-)\ \pi^0(\to \gamma\gamma).$$

While in the signal sample only correctly reconstructed B candidates are considered, the amount of candidates in the continuum sample are limited by pre-cuts that accept only continuum events, which look like signal events and are therefore more difficult to classify as continuum. Those pre-cuts are shown in Table 5.1. In addition to the cuts on the B candidate, there are also cuts for the tracks and cluster information inside the ROE. Both types are cut away, if they have a center-of-mass momentum (see Section 3.2) greater than 3.2 GeV. Also, every track, which is not connected to a hit inside the Central Drift Chamber (see Section 2.3) and every cluster information, which possesses a momentum smaller than 0.05 GeV are thrown away.

Applying the selections for reconstructing the B candidate and building the corresponding ROE leads to a benchmark dataset that contains 4 914 670 events. 56.83 % of these events are from the signal sample and the rest is from the continuum sample. To perform the classification tasks, the dataset is divided into three parts, as shown in Figure 5.1. All classifiers are trained with the dataset `Train`. The figure of merit calculation for hyper-parameter optimization, described in detail in Section 5.3, uses the `Opt` dataset. Finally, each result shown in this thesis is based on the evaluation of the trained classifier on the dataset `Val`.

Table 5.1.: Pre-selection cuts on the different particles for building B candidates. The charge conjugated particles are implicitly included. GoodGamma is a momentum cut on γ, which is cluster region specific. ChiProb is a probability based on the fit performed during reconstruction and piid is the pion identification probability. M_{bc} is the beam constrained mass and $\triangle E$ is the energy difference between the candidate and the theoretical mass M. Values for M, M_{bc} and $\triangle E$ are listed in GeV.

target	cut
γ	goodGamma=1
π^0	$0.115 \leq M \leq 0.152$
π^+	chiProb > 0.001
π^+	piid > 0.5
K_S^0	$0.48 \leq M \leq 0.516$
B^0	$5.2 < M_{bc} < 5.3$
B^0	$-0.3 < \triangle E < 0.3$

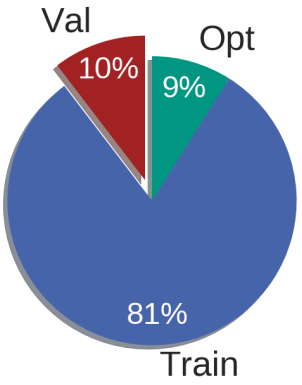

Figure 5.1.: Relative size of the different benchmark dataset parts. Final Evaluation is done on 10 % (`Val`) of the dataset. This dataset will only be used at the end for creating the results. The rest is split in the dataset used for training the classifier (`Train`) and the dataset used for calculation the figure of merit during hyper-parameter optimization (`Opt`). All parts posses the same signal-to-background ratio.

5.2. Choosing Input Features

Choosing the input features is crucial to every classification task. They are a representation of the event, which is meant to have the required information to solve the classification task. While each classifier is limited to the amount of information in the input features, classifiers can be highly dependent on how this information is represented in the features. Also, a small set of features, where each one already separates signal and background very well, is much easier to train, but there might be missing additional information that would be contained in a larger set of features, where each one is not so meaningful. Therefore, this section explains in detail the different input features and their composition in the classifiers.

For the comparison in Section 5.4 three sets of input features are chosen: Those sets are hierarchically ordered, with each set adding additional features to the set before: The Engineered (E) features (Section 5.2.1), the Detector Level (DL) features (Section 5.2.2) and the Vertex (V) features (Section 5.2.3). While the E features already describe the event, the DL and V features are built on a cluster and track basis. Therefore, Section 5.2.4 explains how the corresponding tracks and clusters are grouped to represent the whole event. After introducing all features, Section 5.2.5 introduce the different composition of features used for the comparison in Section 5.4. Before entering the classifier the features are preprocessed, which is discussed in Section 5.2.6.

5.2.1. Engineered (E) Features

Engineered features are created specifically for the purpose of the classification task at hand. Many of these features are also suitable for stand-alone cuts and were refined over the years. The idea was to reduce the features used for continuum suppression to a small number, which can be fed into a classifier.

The design of the E features is based on the understanding of the differences in event shapes of continuum and signal. In Section 3.2 both the event shapes and all engineered features are explained. A more detailed explanation can be found in Ref. [11]. Summing them up will result in 30 input features, which were used by the traditional approach for classification [6], which was state-of-the art for basf2 before this thesis. The distribution of these features are fully shown in Appendix A. Some of these features separate signal and background very well and using them already leads to a very good classification result.

5.2.2. Detector Level (DL) Features

In contrast to the E features, which represent the whole event, the DL features represent only a track or a cluster. Below the different features are explained, which are used for describing tracks and clusters. In the end there are twelve features used for tracks and ten features used for clusters.

Momentum (Clusters and Tracks)

The momentum features chosen for the feature sets contain the magnitude p, the azimuth angle ϕ and the polar angle $\cos\theta$, as well as the uncertainties of the track at the nearest point on the trajectory to the interaction point.

Because the B meson pair decays in an isotropic distribution of spherical shape, the z-axis is also rotated to the thrust axis of the B candidate. The rotated B candidate now always lies in the center of the p-ϕ-plane, which makes feature extraction across different events easier for the classifier. This rotated coordinate system is inspired by the Cleo Cones (see Section 3.2.2) and is called the thrust frame.

ECL Cluster specific Features

These features are specifically made for gathering information about a cluster in the ECL (see Section 2.3). In total there are four features: Timing, Number of Hits, E9E21 and Region.

Timing is used to distinguish event-related clusters from clusters that are caused by the induced background and do not occur at the same time as the event. Both the Number of Hits and E9E21 are characteristics that describe the shape of the cluster to determine whether the particle causing the cluster came from the direction of the interaction point. While the Number of Hits indicates the number of activated crystals of the cluster, E9E21 is an energy ratio of the innermost crystals (3x3) to a larger area of the crystals (5x5 minus 4 corners). The Region describes, if the cluster was detected in the forward endcap, barrel or backward endcap region.

Track specific Features

The track specific features involve PID for kaons, electrons, muons and protons, to determine which kind of particle belongs to the track. Additionally, the χ^2 probability of the track as well as Number of Hits inside the CDC (see Section 2.3) are chosen as features to determine the uncertainty of the reconstruction of the track.

5.2.3. Vertex (V) Features

Vertex features are additional features for describing tracks. The V features consist of the distance and differences between azimuth (polar) angle to the point of interaction, calculated at the nearest point on the trajectory to the point of interaction.

Using V features can increase the classification result but can create unwanted correlation between the classifier output and $\triangle z$. The quantity $\triangle z$ is the B vertex-difference in the boosting direction (see Section 2.2). This quantity is important for time dependent CP violation (see Section 2.1) and therefore is not allowed to be correlated to the classifier output in such analyses. The problem of correlation between the classifier output and $\triangle z$, as well as a more detailed explanation of the quantity is discussed in Section 5.6.

Because of these correlations the Vertex features can not be used for every continuum suppression and are therefore treated specifically. If the V features are used, however, they are calculated in the same coordinate and thrust frame as the momentum features.

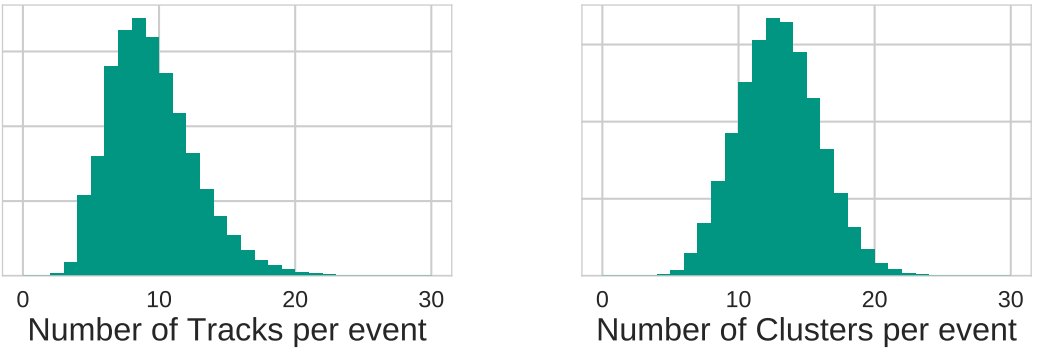

Figure 5.2.: Distribution of number of Tracks (left) and number of Clusters (right) per event. For the selection of the clusters, the goodGamma cut (see Table 5.1) was applied.

5.2.4. Grouping Tracks and Clusters for Event Representation

Every event has different numbers of tracks and clusters, which is outlined in Figure 5.2. However, the classifiers are required to receive a fix amount of input features, and so a fixed amount of tracks and clusters, per training. Therefore, a suitable grouping of tracks and cluster representing the event is necessary.

The process of grouping clusters and tracks is shown in Figure 5.3. Every event representation contains 20 clusters and 20 tracks. They are divided according to whether they are used to reconstruct the B candidate or whether they are in the ROE. Tracks are additionally divided by charge. Therefore, both informations are encoded into the structure of the feature set and do not have to be fed in as additional features for every track and cluster. In order to be independent of the B candidate's decay, the number of tracks (clusters) belonging to the B candidate and the number of tracks (clusters) belonging to the ROE are the same.

In those divisions, clusters and tracks are ordered by highest magnitude of momentum in the center-of-mass frame. If an event does not posses enough clusters or tracks in one category the features in the blank spots will be filled with zeros. Therefore, every event has the same number of features, which is required for most classifiers.

5.2.5. Feature Sets for the Comparison

In Section 5.4 three different feature sets are used for the comparison. The E features are the features used in the traditional approach and therefore serve as the baseline set containing 30 features. In the second set the features are complemented by the DL features without Vertex information (E+DL) resulting in 470 features. The V features are added in the last set (E+DL+V) for a total of 530 features.

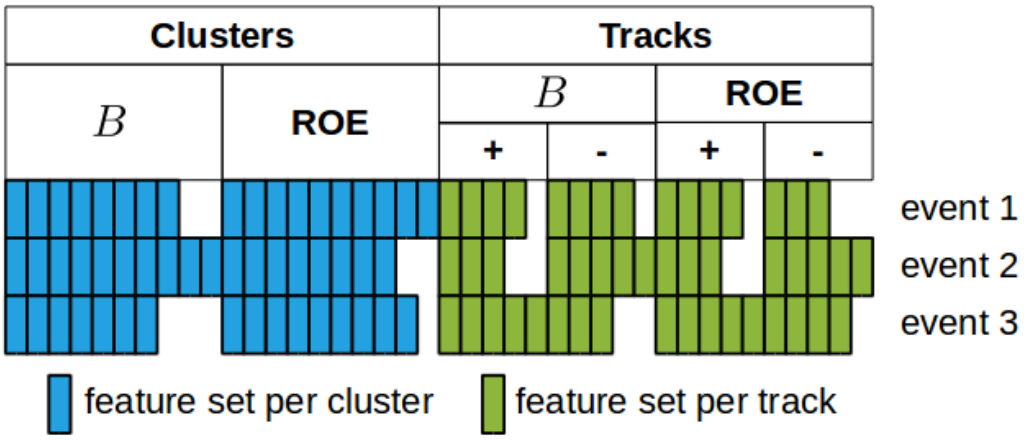

Figure 5.3.: Schematics of the grouping of tracks and clusters. Clusters are shown on the left and are divided into whether they were used to reconstructing the B candidate or are in the Rest of Event (ROE). At maximum ten Clusters per category will be chosen per event. If there are not enough clusters, they will be filled up with blank spots containing zeros. On the right the same procedure is shown for tracks. Because tracks are additionally divided by charge, there are five tracks per division.

5.2.6. Preprocessing

Preprocessing is an important step to improve classifier results by transforming the features to be better suited for the classifier. Feeding in different features with values using different orders of magnitude results in a distorted feature space, where values have a different impact on the classification output based on their absolute values instead of their information about continuum. This can be prevented by normalization, which transforms the space of each feature into a desired region.

By using binned features instead of continuous ones, the training will be more regularized. Minor, non-relevant differences in each feature are prone to over-training without having additional information, although this effect is rather small. Binning can also be distorted by outliers, which could limit the interesting range of the feature into a few bins, which results in a loss of information.

In this thesis equal frequency binning was used, which solves the problem of having outliers in the feature space. This binning approach chooses the binning boundaries in such a way that every bin has the same amount of entries, resulting into a flat distribution. The FastBDT algorithm already applies equal frequency binning before training. For Deep Neural Networks equal frequency binning was implemented by myself.

5.3. Hyper-Parameters

Hyper-parameters are parameters which have to be set by the user and are not automatically chosen during training. Choosing the right hyper-parameters is crucial for a successful classification and can make the difference between a very good classification result and a worthless one. To evaluate different hyper-parameter sets a figure of merit is needed, which evaluates the classification result. For every combination of classifier (BDT and DNN) and feature set Bayesian Optimization (Section 4.4) is performed.

The figure of merit is explained in Section 5.3.1. The explanations of the hyper-parameters as well as the results of the Bayesian Optimization for BDT and DNN are shown in Section 5.3.2 and Section 5.3.3.

5.3.1. Figure of Merit

The figure of merit is a metric which evaluates the quality of a classifier. For this purpose the classifier is trained on the `Train`-Part of the benchmark dataset and evaluated on the `Opt`-Part. As figure of merit the integral of the Receiver Operating Characteristic curve (ROC, [50]) called area under ROC curve (AUC, [45]) is used.

The ROC curve projects background rejection, which is the fraction of background thrown away by a cut on the classifier output, over signal efficiency, which is the fraction of signal remaining in the benchmark dataset after a cut on the classifier output. Therefore, every point on this curve represents a cut on the classifier output. In Section 5.4 ROC curves are shown, which compare the classifiers and feature sets.

For the AUC, a value of 0.5 means no classification ability at all, because the chance of predicting if an event belongs to signal or background is 50 %. An ideal classifier, which classifies every event correctly possesses an AUC of 1.

5.3.2. BDT Hyper-Parameters

BDT are known as very robust classifiers, which means that the classification result should not vary much by changing the hyper-parameters. FastBDT offers five different hyper-parameters which are explained in Section 5.3.2.1. The results of Bayesian Optimization based on the different feature sets are discussed in Section 5.3.2.2.

5.3.2.1. Description

As an ensemble method, BDTs are a combination of many single Decision Trees. Therefore, the first hyper-parameter is the Number of Trees (NoT) building the classifier. The depth of the trees (D) is the information about how many consecutive cuts are performed in a single tree. To control the number of bins in the preprocessor, the Number of Cut Levels (NoCL) is used. Depending on this hyper-parameter 2^n bins are used per feature. In FastBDT, every tree is trained on a subset of the events. The relative size of this subset is determined by the Rand Ratio (RR). As the final hyper-parameter the Shrinkage (Shr) is chosen, which regularizes the update rule of the boosting algorithm similar to a learning rate found in Neural Networks.

Table 5.2.: Boundaries for hyper-parameters chosen for optimizing BDTs with Bayesian Optimization. Although the optimization was done for each feature set, the boundaries of the hyper-parameters were always the same.

Hyper-Parameter	Type	Boundaries
Number of Trees (NoT)	Integer	[10, 1000]
Depth of Trees (D)	Integer	[2, 8]
Number of Cut Levels (NoCL)	Integer	[4, 12]
Rand Ratio (RR)	Real	[0.01, 1]
Shrinkage (Shr)	Real	[0.01, 0.3]

Table 5.3.: Hyper-parameter sets for the BDTs. The standard parameters of FastBDT are chosen as the Educated Guess before Bayesian Optimization is applied. For every feature set, the final hyper-parameter set was chosen based on the results of the optimization. The hyper-parameter sets and their figure of merit for every BDT optimization is shown in Appendix B.1.

	NoT	D	NoCL	RR	Shr
Educated Guess	200	3	8	0.5	0.1
Final Set	1000	8	12	0.7	0.3

5.3.2.2. Hyper-Parameter Optimization

The FastBDT algorithm already has five different hyper-parameters, which can be directly used for the hyper-parameter optimization. In Table 5.2 the boundaries of these hyper-parameters chosen for the optimization are shown. The optimization is done for each feature set with the same boundaries.

An educated guess is given for each optimizer at the start of the training. After this, there are 20 iterations, where the optimizer predicts four new hyper-parameter sets and is then told the results of the computations of the figure of merits. This results in 81 hyper-parameter sets calculated for the optimization for each feature set. In Appendix B.1 the ten best sets for each optimization as well as the educated guess is shown.

Using FastBDT with around 500 input features makes the algorithm very slow. One training could take up to 24 hours and therefore it was not possible to finish the optimization for the feature sets E+DL and E+DL+V. Only 37 hyper-parameter sets for the second feature set and 25 hyper-parameter sets for the third feature set were calculated.

As shown in Appendix B.1 the best hyper-parameter sets for each feature set led to very similar performance. Therefore, only one hyper-parameter set was chosen for every feature set. The choice was made by approximating the trend of the best results rather than taking only the best result. This Final Set and the Educated Guess is shown in Table 5.3. It is noticeable that most of the hyper-parameters become very large and reach the boundaries of the optimization. While the Number of Cut Levels might not affect the classification result very much, the results are surprising for the Number of trees, Depth and Shrinkage. Because the Number of Trees and the Depth of the Trees directly affects the training duration a further upper increase of the hyper-parameter boundaries seems unreasonable, due to the training duration of the optimization. This leads to the conclusion that the FastBDT algorithm is not suitable for dealing with large feature sets.

5.3.3. DNN Hyper-Parameters

Deep Neural Networks can vastly differ in their design. It is impossible to take every possible Neural Network into consideration while performing a hyper-parameter optimization. Therefore, most of the hyper-parameters have to be chosen based on experience before the optimization. In this optimization no special topologies like Relation Networks are considered. Instead, the Neural Networks are all MLPs, which vary in their number of hidden layers and neurons. A key aspect of a Neural Network is the number of parameters (weights and biases) resulting from the topology. It is a good approximation of the complexity of the Neural Network. The hyper-parameters chosen before the optimization and the hyper-parameters tuned by the optimization are explained in Section 5.3.3.1. In Section 5.3.3.2 the results of the optimization for each feature set are discussed. Based on those results a further investigation of certain hyper-parameters was performed in Section 5.3.3.3.

5.3.3.1. Description

To limit the near infinite amount of possible hyper-parameters that could be optimized, most of them have to be chosen before the optimization. Therefore, in this optimization all Neural Network consists of an input layer, which contains all the features from the used feature set. All hidden layers are MLP layers, which use the hyperbolic tangent as activation function. The output layer, which contains only one neuron, uses the sigmoid function as activation function to produce a classifier output between zero and one.

The Number of Hidden Layers (NoHL) and the Number of Neurons per Layer (NoN) are hyper-parameters used for the optimization. In this optimization every hidden layer in a Neural Network is required to have the same number of neurons. There are two additional hyper-parameters for regularization. After every hidden layer, Dropout can be performed. The hyper-parameter Dropout (Drop) determines the probability of each individual neuron dropping out. For every Dropout layer this probability is the same.

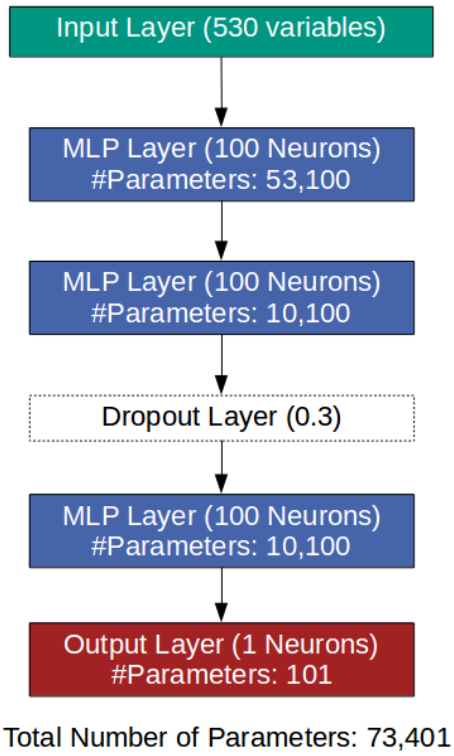

Figure 5.4.: Example of a Neural Network with the Parameter Set: NoHL: 3, NoN: 100, Drop: 0.3 and RD: True. The number of parameters #Parameters are the number of weights and biases used per layer.

The second hyper-parameter for regularization is Restrain Dropout (RD). It is a boolean hyper-parameter, which can set the Dropout after the first and last hidden layer to zero. An example of such a Neural Network with an arbitrary set of hyper-parameters (NoHL: 3, NoN: 100, Drop: 0.3, RD: True) is shown in Figure 5.4. To get a sense of the complexity of the Neural Network the number of parameters (weights and biases) needed by the layers are presented. The four hyper-parameters and their boundaries are shown in Table 5.4. Although the goal was to find the best topology for a Deep Neural Network, the optimization can also try Shallow Neural Networks with only one hidden layer. Hyper-parameters for weight decay and different activation functions were examined in earlier iterations before, but could not improve the classification.

Table 5.4.: Boundaries for hyper-parameters chosen for optimizing DNNs with Bayesian Optimization. Although the optimization was done for each feature set, the boundaries of the hyper-parameters were always the same.

Hyper Paramater	Type	Boundaries
Number of Hidden Layers (NoHL)	Integer	[1, 10]
Number of Neurons(NoN)	Integer	[50, 1000]
Dropout (Drop)	Real	[0, 0.5]
Restrain Dropout (RD)	Categorical	[False, True]

In addition to the topology of the Neural Network, there are many other aspects to consider. For the equal frequency binning in the preprocessing process 100 bins per feature are used. The minimizer responsible for decreasing the loss function uses the Adam algorithm [51] with the standard arguments implemented in Keras. Class weights are used to take into account the imbalance between signal and background. The training is done in batch sizes of 500 events and every event is used for training one time per epoch. To determine the end of the training, 10 % of the `Train`-Dataset is not used for training. Instead, it is used for evaluating the loss function independently. If the loss function calculated on this independent dataset does not decrease for ten epochs, the training is finished. This procedure is called Early Stopping.

5.3.3.2. Hyper-Parameter Optimization

The Bayesian Optimization for the Neural Network was performed in the same way as with the BDTs. The idea of the Educated Guess in Table 5.5 was to start with a relatively small topology without regularization. Because these Neural Networks take only up to two hours to be trained, all 81 trainings could be performed for every feature set.

In Table 5.5 the chosen topology for each feature set are shown that is based on the best ten results per feature set listed in Appendix B.2. It is very noticeable that trainings without regularization are strongly preferred. Besides this, there is no clear trend on the remaining two hyper-parameters. Therefore, the topologies chosen for the different feature sets are quite different. For the feature set E the Neural Network is relatively small, which reflects the number of parameters. The topologies for feature sets E+DL and E+DL+V differs very much: The first one is a Shallow Neural Network with only one hidden layer containing the maximum number of neurons and the other uses the minimum number of neurons with five hidden layers. Such vast differences raise the suspicion that the hyper-parameter space should be more constrained before applying the Bayesian Optimization.

In contrast to BDTs, Neural Networks need a lot more expert knowledge to tune the hyper-parameters. Even with this constrained topology, there was no clear optimal topology. Because of this unstable result, a second study was performed to explicitly compare shallow with deep networks that decides which topology leads to the best performance.

Table 5.5.: Hyper-parameter sets for the DNNs. The Educated Guess is chosen before Bayesian Optimization. For each feature set, the final set was chosen based on the results of the optimization. The hyper-parameter sets and their figure of merit for every DNN optimization is shown in Appendix B.2. #Parameters is the number of total parameters (weights and biases) used by the Neural Network. Because #Parameters also depends on the input features, the number from the Educated Guess varies depending on the feature set.

	NoHL	NoN	Drop	RD	#Parameters
Educated Guess	4	100	0	True	33,501/77,501/83,501
Final Set (E)	2	160	0	True	30,881
Final Set (E+DL)	1	1000	0	True	472,001
Final Set (E+DL+V)	5	50	0	True	36,801

5.3.3.3. Shallow or Deep Neural Network

To compare Shallow Neural Networks with Deep Neural Networks the number of total parameters (containing weights and biases) is interesting. While the number of total parameters grows linearly with the number of hidden layer, the growth is quadratic with the number of neurons per layer. In Figure 5.5 the AUC scores are shown for a Shallow Neural Network with only one hidden layer and a Deep Neural Network with 50 neurons per layer. The shared x-axis represents the total number of parameters ranging from 26 600 to 50 000 and the trainings are evaluated with the `Opt`-Dataset on feature set three (E+DL+V). To see which topology is the best one, each training is performed five times with the same hyper-parameter set and the median of the AUC score is chosen, along with the best and worst result as boundaries. Because of the results in Table 5.5, no Dropout regularization is considered.

The Deep Neural Network with three hidden layers has the best performance in this comparison. While the Shallow Neural Network is relatively constant around the number of neurons and the boundaries are closer to the median, the classification quality of the Deep Neural Network drops significantly with seven or more hidden layers. This drop in performance is also present, when evaluating the Deep Neural Networks with the `Train`-Dataset (not shown in the Figure) and is therefore not due to over-training.

Not shown is a big Shallow Neural Network with many more neurons like the one in Table 5.5. Therefore, the best Deep and Shallow Neural Network from Figure 5.5, as well as Shallow Neural Networks with many more neurons are shown in Table 5.6. Although the bigger Shallow Neural Networks have many more parameters than the Deep Neural Network, the performance is still not better than the Deep Neural Network.

Considering the results from the last two sections, the Deep Neural Network for the comparison in Section 5.4 has three hidden layers containing 50 neurons for feature set two and three. For the feature set E, which has significantly fewer features, the result from Table 5.5 is used. In all Neural Networks no regularization is performed.

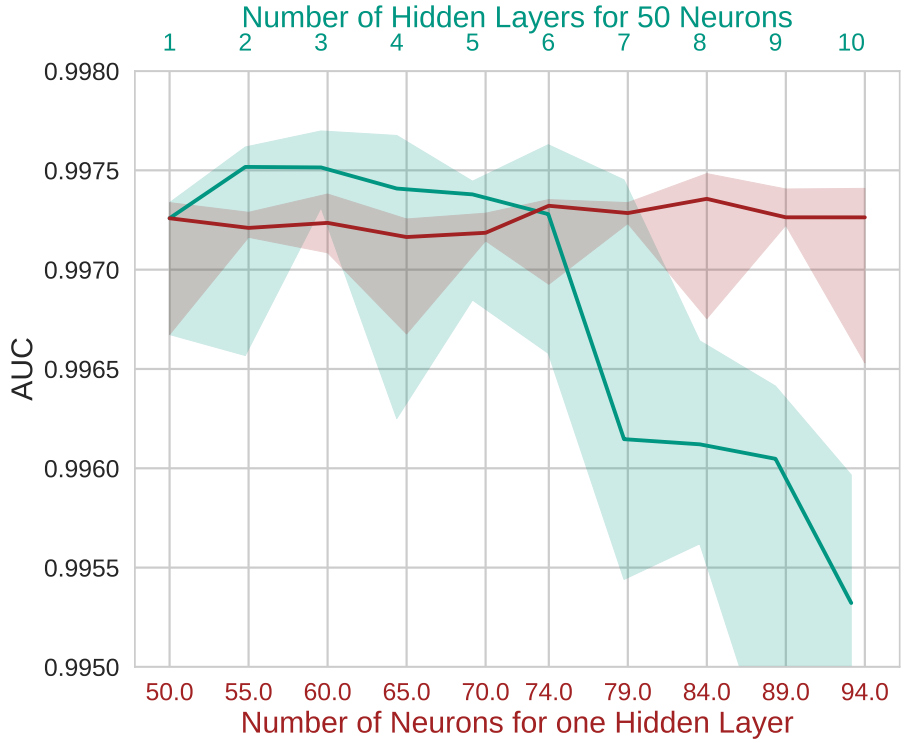

Figure 5.5.: Comparison of a Shallow Neural Network with only one hidden layer and a Deep Neural Network with 50 Neurons per layer. The x-axis also represents the number of total parameters from the Networks, ranging from 26 600 to 50 000. Each Training was done five times on the `Train`-Dataset and was evaluated using the `Opt`-Dataset. To show the best topology, the Median of the AUC scores, as well as the best and worst result as boundaries, are shown.

Table 5.6.: Comparison between the best Deep and Shallow Neural Networks from Figure 5.5 and bigger Shallow Networks with many more parameters.

NoHL	NoN	Median of AUC	#Parameters
3	50	0.99751	31 701
1	84	0.99736	44 689
1	250	0.99733	133 001
1	500	0.99732	266 001
1	1000	0.99724	532 001
1	1500	0.9973	798 001

5.4. Comparison of Traditional & Deep Learning Approaches

This section compares two kinds of improvements for the continuum suppression: Changing the feature set used for training and changing the classifier. In this section only Deep Neural Networks with no special topology are used. The setup of this comparison and the metrics of evaluation are explained in Section 5.4.1. In Section 5.4.2 the results are presented.

5.4.1. Methodology of the Comparison

The comparison is done with two classifiers, each trained on three different feature sets. The traditional approach BDT (E), which is the FastBDT algorithm trained on the feature set E, was state of the art before this comparison. Therefore, it serves as the baseline.

To compare the general classification capability the ROC curve [50] is chosen. Also, to sum up the curve into one scalar value, the integral of the curve (AUC, [45]) is chosen.

Not every region in the ROC curve is interesting for the continuum suppression. Usually the continuum suppression is applied at an early step of an analysis and therefore should not reduce the amount of signal events significantly. Because of this, only cuts which retain high signal efficiencies will be considered. Also, for this comparison it is interesting how the classifiers perform relative to the baseline BDT (E). Therefore, a new metric is created: the Relative amount of Background on a 98 % signal efficiency cut (RB(98)). This metric cuts on the classifier output on values where only 2 % of the amount of signal is thrown away. The amount of background remaining after this cut is shown relative to the amount of background of the baseline using the same procedure. As an example an RB(98) of 60 % means that the user can expect only 60 % of the background compared to the baseline classifier, loosing 2 % of signal in both cases.

As the last metric for comparison, the training time is chosen. This should only serve as an approximation, because training time is hardware dependent and BDTs and DNNs were trained on different hardware. For the training of the DNNs GPUs were used to speed up the training, while for BDTs only CPU based trainings are possible. Also, the time includes the reading time of the files containing the input features and the training time is very dependent on the chosen hyper-parameters. Nonetheless, the training time is important because up to this comparison every classifier was trained many times to get a better understanding of the classifier and to tune its hyper-parameters. Therefore, a shorter training time means additional possibilities to tune the classifier.

5.4.2. Results

The ROC curves and their AUC scores are shown in Figure 5.6. Looking at the different feature sets, the classifiers are strongly feature dependent. With each additional feature set, the classification result significantly improves. Apart from the first feature set E, there are hardly any differences between BDTs and DNNs.

In Table 5.7 the RB(98) scores and the training time is shown. The RB(98) scores further confirm the huge increase in classification capability using the new feature sets. With the feature set E+DL the amount of background is only around 20 % relative to the amount of the traditional approach BDT (E). Including the Vertex features the background is additionally halved and the amount of background is now only 10 % of the amount of the traditional approach.

The big difference between DNNs and BDTs are the training times in Table 5.7. While the slowest DNN needs approximately one hour for training, the BDTs with the new feature sets (E+DL and E+DL+V) need over a day. The time differences are due to the complexity of the models. For the DNN a rather small topology with few parameters is chosen, but for the BDT a huge number of Trees with a big depth is chosen. The BDT classifier has to train 1000 Trees, in which the last layer alone contains $2^8 = 256$ bins. Also, the training time of the BDT grows linearly with respect to the number of input features. In addition to this the DNNs are trained on a GPU, which isn't possible for BDTs using the FastBDT algorithm.

In summary, changing the feature sets has a huge impact on the training performance. While BDTs can achieve similar results to DNNs, the DNN only takes up a fraction of the BDT's training time.

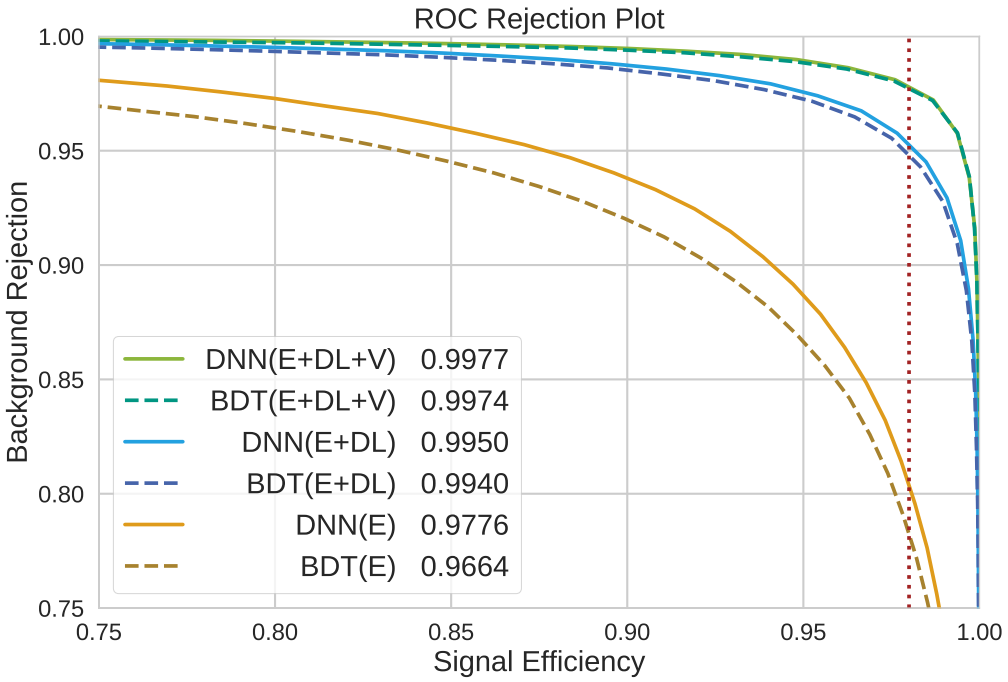

Figure 5.6.: ROC curve of the BDTs and DNNs for each feature set. Each training is performed five times and the best result is used for this plot. The corresponding AUC score is listed in the legend. The 98 % signal efficiency cut used for Table 5.7 is shown in red. For this evaluation, the `Val`-Dataset was used.

Table 5.7.: This table shows the Relative amount of Background on a 98 % signal efficiency cut (RB(98)) and the time needed for training for each classifier in Figure 5.6. As a baseline, the traditional approach BDT based on the first feature set is chosen.

Classifier	RB(98) in %	Training Time in h:min
DNN (E+DL+V)	9.81	1:01
BDT (E+DL+V)	10.12	26:26
DNN (E+DL)	21.65	0:33
BDT (E+DL)	23.24	25:42
DNN (E)	90.35	0:54
BDT (E)	100	1:39

5.5. Relation Network

Relation Networks are a very recent development in Deep Learning [37]. With this technique feature groups can be compared pairwise using shared weights, and therefore reduce the number of parameters (see Section 4.3.1). This section describes how Relation Networks can improve the classification result using the classifiers with the third feature set (E+DL+V) as comparison. It starts by explaining the setup of the used Relation Networks in Section 5.5.1. The improvements compared to the BDT and DNN are shown in Section 5.5.2.

5.5.1. Setup

To have a sound comparison, the datasets used for training and evaluation as well as the metrics are identical to Section 5.4.1. Because Relation Networks are a very recent development and there is not much information about their capabilities and preferred topologies, a Bayesian Optimization was not performed. Instead of fine-tuning their hyper-parameters, the focus was to get experience with these networks and find a working topology.

The chosen design of the Relation Network is shown in Figure 5.7. This shows the Relation Network RN (E+DL+V), where the E features are fed event-wise into every comparison. The Relation Network RN (DL+V) has the same structure, but omits the engineered information (yellow part in the figure). Global Average Pooling, which takes the average value over all permutations of the Relation layer output, as well as the shown MLP and Dropout layer after pooling are topologies inspired by Ref. [37].

During this thesis a third Relation Network was tested. It is like the one in Figure 5.8, but extends the input of the first MLP layer by feeding in the 530 input features directly. Hence, the Network was a DNN, using the output of the Relation Network as additional input. The relational part of the network was trained first, before connecting it to the whole network. Although this Relation Network should have the potential to have a better classification capability than the other ones, no better classification performance could be achieved. Therefore, it is not considered for the comparison.

5.5.2. Results

The ROC curves and their AUC-Score are shown in Figure 5.8. Both Relation Networks have better classification capabilities than the BDT and DNN. The modified Relation Network with the E features performs better than the Relation Network without those features. In Table 5.8 the RB(98) scores confirm these results. Using RN (E+DL+V) reduce background around 15 % more than using DNN (E+DL+V). It is interesting that RN (E+DL+V) has a shorter training time than RN (DL+V) despite using more features. Therefore, using the E features as event information inside the Relation Network leads to faster convergence and a better classification result.

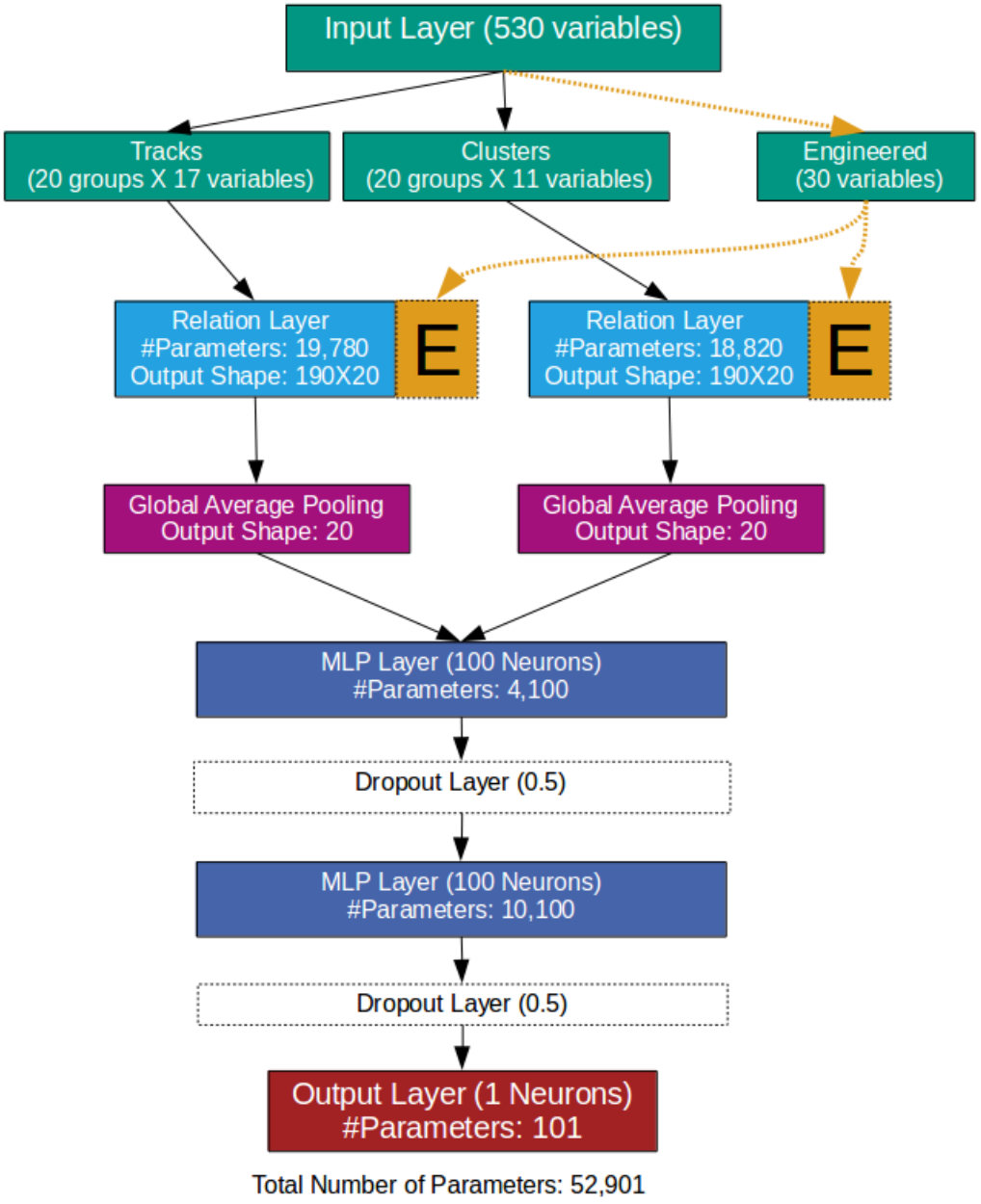

Figure 5.7.: Relation Network RN (E+DL+V) used for the comparison. The input features are split in their different groups, before fed into the Relation layers. Because the order of the tracks and clusters is irrelevant to the Relation layer the charge and ROE information are added as features. Global Average Pooling takes the average value over every permutation of the Relation layer output. After that, the output is fed into a MLP. In case of the Relation Network RN (DL+V) the yellow part of the network, where the E features are fed as event information into the Relation layers, are removed.

Table 5.8.: This table shows the Relative amount of Background on a 98 % signal efficiency cut (RB(98)) and the time needed for training for each classifier in Figure 5.8. As a baseline, the traditional approach BDT based on the first feature set E is chosen (see Section 5.4.1).

Classifier	RB(98) in %	Training Time in h:min
RN (E+DL+V)	8.13	19:43
RN (DL+V)	8.61	33:53
DNN (E+DL+V)	9.81	1:01
BDT (E+DL+V)	10.12	26:26

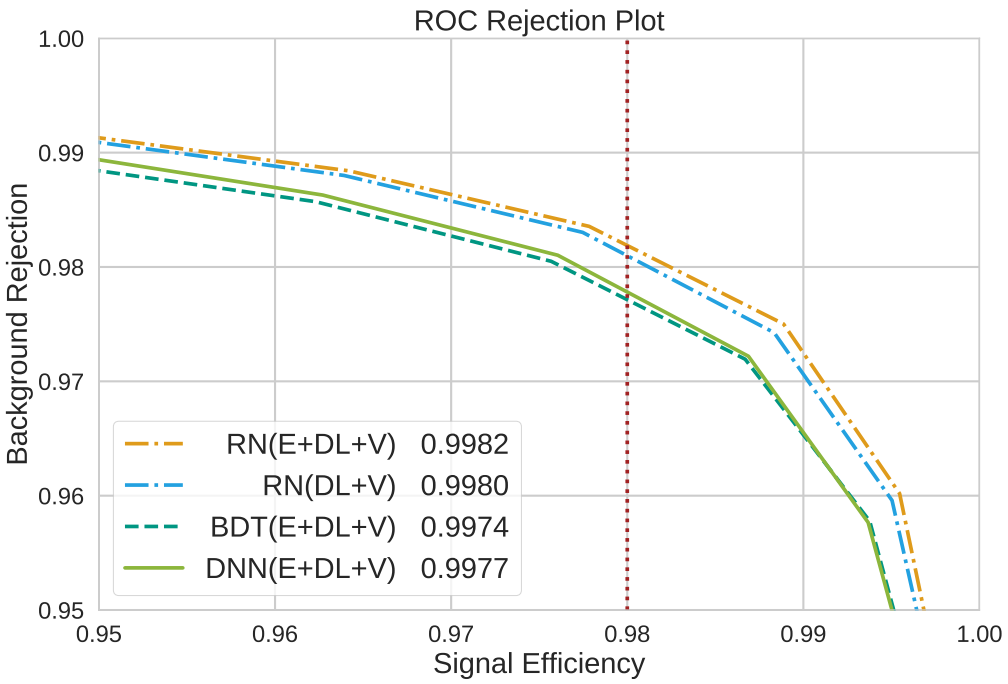

Figure 5.8.: ROC curve of the RNs and the best DNN and BDT from Figure 5.6. Each training is performed five times and the best result is used for this plot. The corresponding AUC score is listed in the legend. The 98 % signal efficiency cut used for Table 5.8 is shown in red. For this evaluation, the `VAL`-Dataset was used.

5.6. Adversarial Network

Adversarial Networks are an active field of research of Deep Learning, which is especially true for Generative Adversarial Networks (GAN, [39]). Their concept is explained in Section 4.3.2. In this thesis, Adversarial Networks are used to prevent correlations between the classifier outputs and $\triangle z$, which is a quantity explained in Section 5.6.1. The topology and the additional hyper-parameters of the used Adversarial Network is explained in Section 5.6.2. Finally, the classification results as well as the corresponding correlations between the classifier outputs and $\triangle z$ are presented in Section 5.6.3.

5.6.1. Correlation between Vertex Features and $\triangle z$

The quantity $\triangle z$ describes the difference in vertex position of the two B-mesons in boost direction. A schematic of this is shown in Figure 5.9. The measurement of $\triangle z$ is crucial in analyses, where time-dependent CP violation (see Section 2.1) like in Ref. [19] is measured. The shape of $\triangle z$ must not change after a cut on the classifier output of the continuum suppression, which means that the classifier output and $\triangle z$ have to be uncorrelated.

The influence of the V features can result in a correlation between the classifier output and $\triangle z$. To confirm this hypothesis, the impact from the classifier output on the shape of $\triangle z$ is examined on three different classifiers: a random classifier whose classifier output is a random series of numbers, a DNN (E+DL) and a DNN (E+DL+V) from Section 5.4. The random classifier is obviously not correlated at all to $\triangle z$ and serves as the offset, which determines how much fluctuation can be expected in the evaluation.

To measure the impact of the classifier output on $\triangle z$, a Classifier Output Dependent (COD) normalized distribution of $\triangle z$ is chosen. In this thesis, only the signal distribution of $\triangle z$ is shown, because there are no significant correlations in the background distribution of $\triangle z$. For the random classifier this distribution is presented in Figure 5.10. The distribution of $\triangle z$ without applying a cut on the classifier output is shown as a black line. The sharp peak at zero is due to ROE vertex fits, which fail to converge. In a physics analysis, these would nominally be removed at the reconstruction level. However, this example focuses on the changes in the distribution and not the distribution itself, so these candidates are not discarded. For every possible quantile cut on the classifier output, the distribution is drawn in the respective color of the cut, if the shape deviates from the black line. If one cuts away 20 % of the signal, the deviation in the distribution are shown in the color red (value 20 at the color bar). Since in most analyses the continuum suppression is done at an early step, high quantile cuts are not reasonable and thus the colors representing high quantile cuts fade to white. On the other hand, red deviations from the distribution are significant, because then the distribution is affected even by low quantile cuts.

In addition to the COD distribution, there is also a flatness score that determines how flat the distribution is. To determine the score, the distribution is binned by equal frequency and divided into several distributions using different quantile cuts. Then, the squared differences between the distributions are summed up and normalized. The closer the value is to zero, the flatter the distribution and the less correlated the classifier output is to $\triangle z$.

In Figure 5.11 the COD distribution of $\triangle z$ with the classifier DNN (E+DL) is shown. There is not much difference between Figure 5.10 and Figure 5.11, which indicates that the classifier output from DNN (E+DL) is not correlated to $\triangle z$ as suspected. However, this changes with the addition of the V features in Figure 5.12. Using the classifier DNN (E+DL+V) results in a significant bias in the signal distribution of $\triangle z$. This is also reflected by the flatness score at the top of the distribution, which is four times higher than the score from the random classifier.

Comparing the classifier outputs from DNN (E+DL) and DNN (E+DL+V) indicates, that the distribution of $\triangle z$ is biased by using the V features. However, using the V features also increases the classification results as outlined in Section 5.4. This raises the question, if the Adversary Network is able to reduce this correlation while retaining a classifier performance above that of the classifier DNN (E+DL).

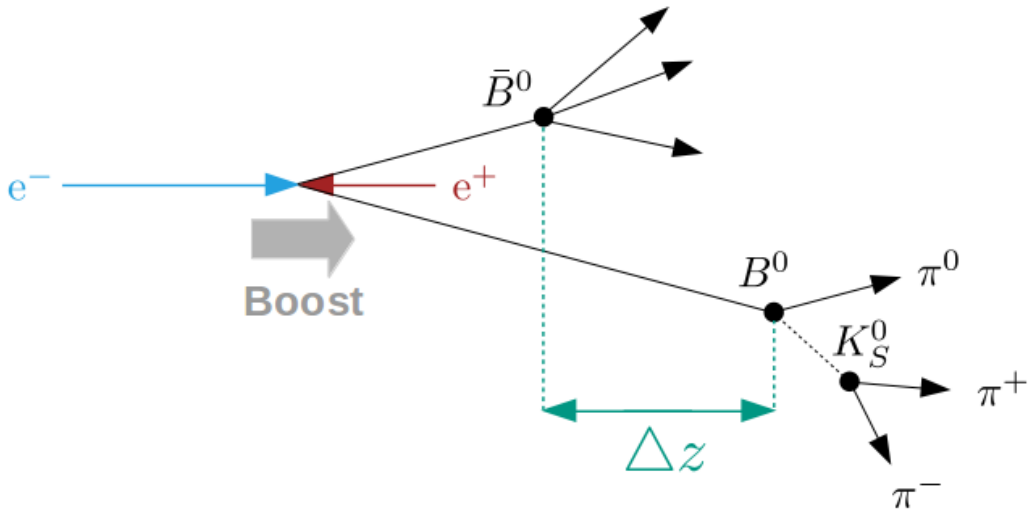

Figure 5.9.: Schematics of $\triangle z$, which is the vertex difference of the two B-mesons in the boost direction.

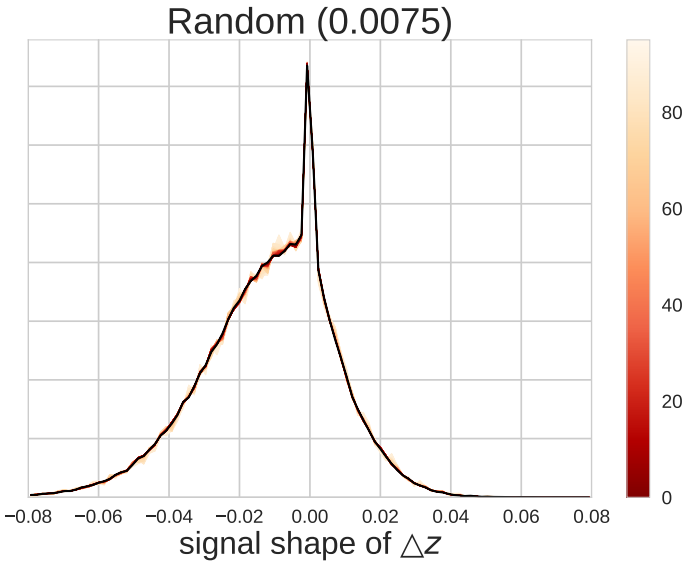

Figure 5.10.: Classifier Output Dependent (COD) normalized signal distribution of $\triangle z$ with a random classifier. The distribution without applying a classifier cut is drawn as a black line, while the different quantile cuts are drawn as deviations from the black line in their respective colors. Significant deviations are drawn in red, while not important deviations from high quantile cuts fade to white. The value at the top represents the flatness of the distribution. It is clear, that the random classifier has no correlation at all to $\triangle z$.

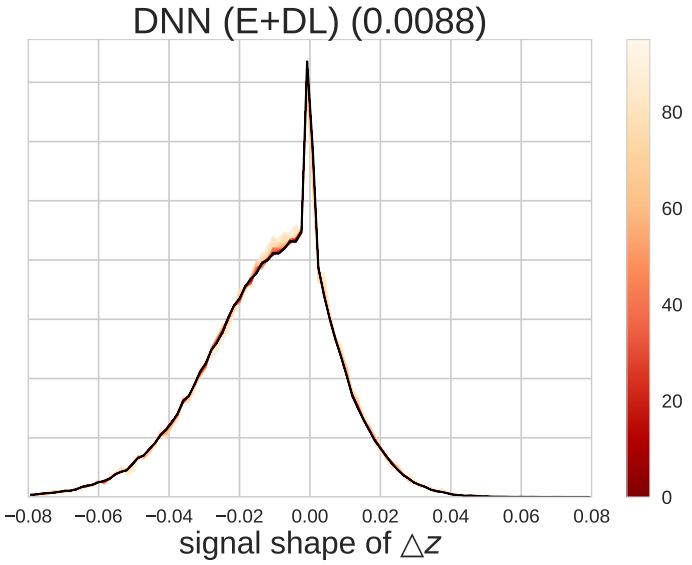

Figure 5.11.: COD signal distribution of $\triangle z$ with the classifier DNN (E+DL) from Section 5.4. For details of this representation method see Figure 5.10. The classifier output has close to none correlations to $\triangle z$ considering the offset shown in Figure 5.10.

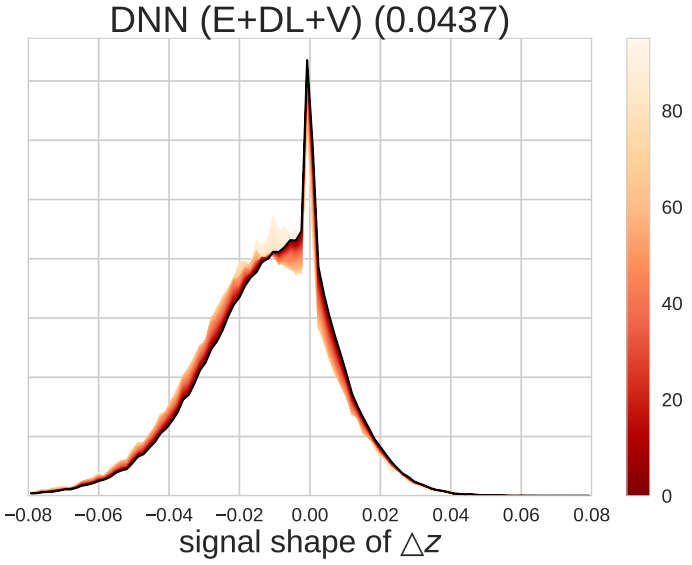

Figure 5.12.: COD signal distribution of $\triangle z$ with the classifier DNN (E+DL+V) from Section 5.4. For details of this representation method see Figure 5.10. The classifier output has significant correlations with the signal distribution of $\triangle z$ in contrast to that in Figure 5.11.

5.6.2. Structure of the Adversarial Network

The Adversarial Network consists of two parts: the classifier and the discriminators. Besides the addition in the loss function from the discriminators, the classifier is built identically to the classifier DNN (E+DL+V). The discriminators are built like in Figure 4.6 of Section 4.3.2. The first two MLP layers have the same amount of neurons, as the input layer (530). After these layers, four Gaussians are built for the Gaussian Mixture Model, which tries to predict $\triangle z$. Using this topology, two discriminators are built to predict separately the signal and background distribution of $\triangle z$.

During every step of training the classifier, the discriminators have to be able to approximate the distribution of $\triangle z$ as good as possible, although the classifier output, which serves as the input to the discriminators, can change with each training step. Therefore, the discriminators always need to be close to the minimum of its own loss function during training of the classifier. To accomplish this, the discriminators are trained ten steps after each step of the classifier training. This results in an increase in training time by a factor of ten.

To determine on how much the discriminators are affecting the loss function of the classifier, the hyper-parameter λ is introduced, which weights the loss function of the discriminators before combining it with the loss function from the classifier. This hyper-parameter regularizes the increase in flatness against the decrease in classification performance and has to be explored for every new task. For this thesis, different values were tested and a value of 0.4 for λ was chosen.

5.6.3. Results

In order to demonstrate the potential of Adversarial Networks, one was trained for this thesis using the feature set E+DL+V. The correlation between its classifier output and $\triangle z$ is shown in Figure 5.13, while its performance compared to DNN (E+DL) and DNN (E+DL+V) is presented in Figure 5.14.

Figure 5.13 demonstrates, that the Adversarial Network significantly reduce the correlation between the classifier output and $\triangle z$. The flatness score is only a third of the score from DNN (E+DL+V) and now much closer to DNN (E+DL). Also, the red deviations vanished almost entirely. Looking at the ROC curve in Figure 5.14, the Adversarial Network performs only slightly worse than DNN (E+DL+V) and is still a big improvement over the classifier without V features. Therefore, the Adversarial Network is able to significantly reduce correlations while retaining most of the classification performance.

In Section 4.3.2 there is also a technique introduced which solves the same problem as the Adversarial Networks, in which the FastBDT is punished by a flatness loss similar to the flatness score. While in this case FastBDT can produce similar results as the Adversarial Networks, the technique is strongly dependent on the meaningfulness of the used metric. In contrast to that, the discriminator is a trained Neural Network, which tries to approximate $\triangle z$ itself and therefore is not dependent on a handcrafted metric.

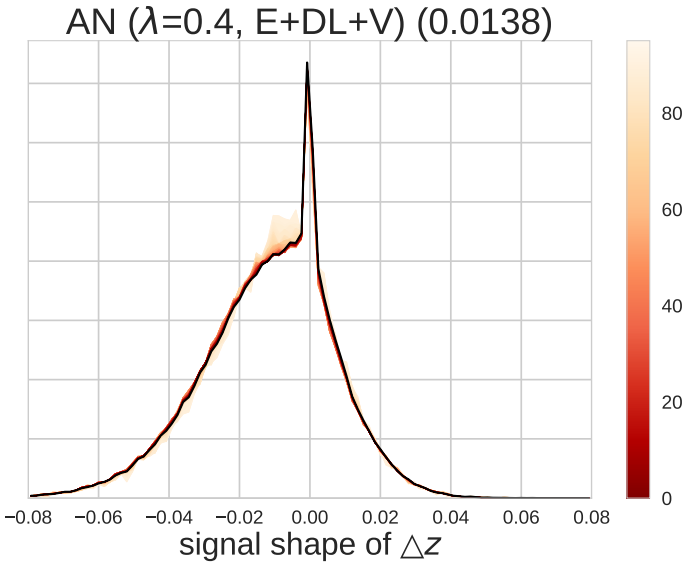

Figure 5.13.: COD signal distribution of $\triangle z$ with an Adversarial Network's classifier output. For details of this representation method see Figure 5.10. The classifier output has significantly fewer correlations to the signal distribution of $\triangle z$ than DNN (E+DL+V) in Figure 5.12.

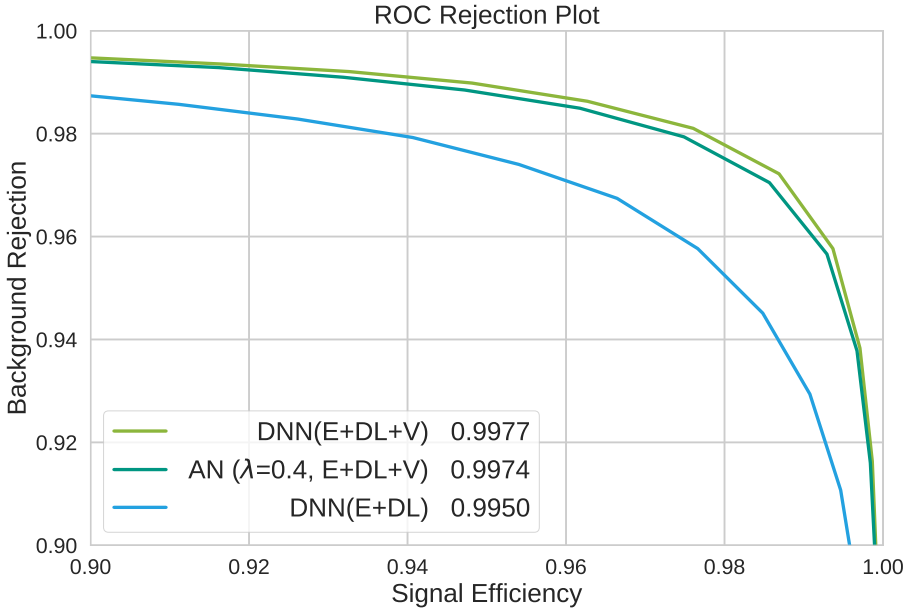

Figure 5.14.: ROC curves of DNN (E+DL) and DNN (E+DL+V) from Section 5.4 and an Adversarial Network AN trained with the same topology in the classifier part.

5.7. Discussion of Results

This thesis started with a Boosted Decision Tree, which classified continuum background based on the 30 E features. By introducing new and much bigger feature sets, that rather focus on using all information from the detector than to find a compact engineered representation of the event, the continuum background could be reduced from a RB(98) score of 100 % to a score around 10 %. Although those improvements can be accomplished with DNNs as well as BDTs, DNNs have a significantly faster training time when dealing with large amounts of input features.

Besides DNNs that exclusively use MLP layers more complex topologies were examined like Relation Networks. Those Networks could further decrease the amount of continuum by 15 % to a RB(98) score of 8.13 %. Because Relation Networks are a very recent development, it is likely that the full potential of this topology was not exhausted during this thesis. More research in the hyper-parameters of the Relational layer and the composition between Relational and MLP layers, such as feeding the input features before and after the Relational layer, could improve the continuum suppression significantly.

Using all kinds of features can lead to unwanted correlations between the classifier output and other quantities like $\triangle z$. Instead of dropping features in the input features, this thesis showed that Adversarial Networks are able to reduce the correlations significantly while achieving a higher classification performance. Adversarial Networks can be built after every classifier based on Neural Networks like DNN or Relation Networks. Although there is a similar technique available for BDTs, this technique is dependent on a metric, which tries to describe the correlation between the classifier output and $\triangle z$. In contrast, Adversarial Networks try to directly predict the shape of $\triangle z$ and therefore are not dependent on a scalar representation.

The use of Deep Learning techniques in continuum suppression does not only improve the classification results, they also open up new possibilities to known problems. Because this thesis covers rather new techniques like Relation and Adversarial Networks, the full potential of these might not be fully understood at this point. Further examinations, which increase the experience with these techniques, might significantly benefit the continuum suppression. During this thesis, all the discussed Deep Learning techniques were implemented as examples into the software framework basf2 and are ready to be used for analysis.

6. Conclusion & Outlook

In this thesis Deep Learning techniques were applied to the task of continuum suppression. By introducing new feature sets and using Deep Neural Networks the amount of continuum was reduced to a fraction of 9.81 % compared to the 100 % using the engineered features with a BDT, which was state of the art before this thesis. With the use of a Relation Network the amount of continuum was further reduced to a fraction of 8.13 %. Additionally, Adversarial Networks proved capable of removing correlations between the classifier output and $\triangle z$, without discarding the correlation-inducing input features.

While the hyper-parameters of the DNNs and BDTs were researched using Bayesian Optimization, this could not be applied to Relation and Adversary Networks. These approaches take far more training time than the average DNN and instead of thousands of CPUs, only four GPUs were available as shared resources during this thesis. Therefore, the hyper-parameters of the Relation and the Adversarial Networks could only be studied to a working minimum. A further refinement of the topology and hyper-parameters of these techniques can result into better and more robust classification results. With more GPUs the number of trainings in parallel can be increased and a more detailed hyper-parameter optimization for Relation and Adversarial Networks will be possible.

One of the limiting factor in this thesis was the amount of continuum, because the reconstruction of $B^0 \to K^0_S \pi^0$ already discards most of it. Although the remaining amount was sufficient to train and evaluate the classifiers, the performance of the baseline BDT (E) was so good that limiting the training to continuum with events that are difficult to classify could improve the performance of the classifiers. A new comparison with another B candidate that does not discard most of the continuum in building the candidate can lead to new insights.

During this thesis only Monte Carlo generated events were used for the comparison. Therefore, the difference between measured data and generated events could not be considered during this thesis. The Software Framework basf2 contains a package, that converts Belle measured data into basf2, which makes it possible to test the deep continuum suppression on measured data. Using Belle data also opens up the possibility to do a full analysis using the deep continuum suppression. In this process the impact of Adversarial Networks reducing unwanted correlations to quantities crucial to the analysis can be examined.

Continuum suppression is only one part of an analysis. There are other tasks like suppress wrongly reconstructed candidates in a $\Upsilon(4S)$ event or Flavour Tagging. In the past, each task was handled separately. With Deep Learning it is possible to combine those tasks into one classifier with multiple outputs. The classifier would then be trained on multiple targets, which can improve the overall performance across the various tasks.

Finally, it is important not to consider Deep Learning as a complete set of techniques. Any innovation in Deep Learning could have the potential to significantly improve the work of particle physicists. Like the Standard Model, Deep Learning is a field of research in which the human mind tries to understand what lies beyond its knowledge.

Bibliography

[1] **ATLAS**, G. Aad *et al.*, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," *Phys. Lett.* **B716** (2012) 1–29, arXiv:1207.7214 [hep-ex].

[2] **CMS**, S. Chatrchyan *et al.*, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," *Phys. Lett.* **B716** (2012) 30–61, arXiv:1207.7235 [hep-ex].

[3] M. Kobayashi and T. Maskawa, "Cp-violation in the renormalizable theory of weak interaction," *Progress of Theoretical Physics* **49** no. 2, (1973) 652–657. +http://dx.doi.org/10.1143/PTP.49.652.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning,".

[5] J. Gemmler, "Study of b meson flavor tagging with deep neural networks at belle and belle ii," ms, Karlsruhe Institute of Technology (KIT), 2016. https://ekp-invenio.physik.uni-karlsruhe.de/record/48849. Karlsruhe Institute of Technology (KIT), Masterarbeit, 2016.

[6] A. J. Bevan, B. Golob, T. Mannel, S. Prell, B. D. Yabsley, *et al.*, "The physics of the b factories," *The European Physical Journal C* **74** no. 11, (Nov, 2014) 3026. https://doi.org/10.1140/epjc/s10052-014-3026-9.

[7] **Belle-II**, T. Abe *et al.*, "Belle II Technical Design Report," arXiv:1011.0352 [physics.ins-det].

[8] **Belle Collaboration**, K. Abe *et al.*, "Observation of large *CP* violation in the neutral B meson system," *Phys. Rev. Lett.* **87** (Aug, 2001) 091802. https://link.aps.org/doi/10.1103/PhysRevLett.87.091802.

[9] **Belle**, K. Abe *et al.*, "Observation of mixing induced CP violation in the neutral B meson system," *Phys. Rev.* **D66** (2002) 032007, arXiv:hep-ex/0202027 [hep-ex].

[10] **BABAR Collaboration**, B. Aubert *et al.*, "Observation of *CP* violation in the b^0 meson system," *Phys. Rev. Lett.* **87** (Aug, 2001) 091801. https://link.aps.org/doi/10.1103/PhysRevLett.87.091801.

[11] **SuperKEKB Physics Working Group**, A. G. Akeroyd *et al.*, "Physics at super B factory," arXiv:hep-ex/0406071 [hep-ex].

[12] C. Schwanda, "Superkekb machine and belle ii detector status," *Nuclear Physics B - Proceedings Supplements* **209** no. 1, (2010) 70 – 72. http://www.sciencedirect.com/science/article/pii/S0920563210004482. Proceedings of the Third Workshop on Theory, Phenomenology and Experiments in Heavy Flavour Physics.

[13] **Particle Data Group**, C. Patrignani *et al.*, "Review of Particle Physics," *Chin. Phys.* **C40** no. 10, (2016) 100001.

[14] P. Raimondi. 2nd superb workshop, frascati, 2006. http://www.lnf.infn.it/conference/superb06/talks/raimondi1.ppt.

[15] "Desy belle and belle ii.". http://belle2.desy.de/sites2009/site_belle2/content/e127118/SuperKEKB:BelleII.jpg. Accessed: 2017-09-29.

[16] A. Moll, "The software framework of the belle ii experiment," *Journal of Physics: Conference Series* **331** no. 3, (2011) 032024. http://stacks.iop.org/1742-6596/331/i=3/a=032024.

[17] P. Urquijo and T. Ferber, "Overview of the belle ii physics generators," *BELLE2-NOTE* **H-2015-006** (2016) . https://docs.belle2.org/record/282/files/BELLE2-NOTE-PH-2015-006-v2.pdf.

[18] **The Belle Collaboration**, Y.-T. Duh, T.-Y. Wu, P. Chang, G. B. Mohanty, Y. Unno, *et al.*, "Measurements of branching fractions and direct *cp* asymmetries for $b \to k\pi$, $b \to \pi\pi$ and $b \to kk$ decays," *Phys. Rev. D* **87** (Feb, 2013) 031103. https://link.aps.org/doi/10.1103/PhysRevD.87.031103.

[19] M. Röhrken, "Time-dependent cp violation measurements in neutral b meson to double-charm decays at the japanese belle experiment,". https://ekp-invenio.physik.uni-karlsruhe.de/record/48212.

[20] E. Farhi, "Quantum chromodynamics test for jets," *Phys. Revi. Lett.* **39** (Dec., 1977) . https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.39.1587.

[21] D. M. Asner, M. Athanas, D. W. Bliss, *et al.*, "Search for exclusive charmless hadronic b decays," *Phys. Rev. D* **53** (Feb, 1996) 1039–1050. https://link.aps.org/doi/10.1103/PhysRevD.53.1039.

[22] G. C. Fox and S. Wolfram, "Observables for the analysis of event shapes in e+e- annihilation and other processes," *Physical Review Letters* **41** (1978) 1581–1585.

[23] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics* **7** no. 2, (1936) 179–188. http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[24] M. Prim, "Angular analysis of b→ πk* decays and search for cp violation at the belle experiment,". https://ekp-invenio.physik.uni-karlsruhe.de/record/48306. KIT, Diss., 2013.

[25] G. Bohm and G. Zech, *Einführung in Statistik und Messwertanalyse für Physiker*. DESY, 2006. https://books.google.de/books?id=isVrtwAACAAJ.

[26] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *Ann. Statist.* **29** no. 5, (10, 2001) 1189–1232. https://doi.org/10.1214/aos/1013203451.

[27] T. Keck, "Fastbdt." https://github.com/thomaskeck/FastBDT.

[28] T. Keck, "Fastbdt: A speed-optimized multivariate classification algorithm for the belle ii experiment," *Computing and Software for Big Science* **1** no. 1, (Sep, 2017) 2. https://doi.org/10.1007/s41781-017-0002-8.

[29] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review* (1958) 65–386.

[30] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[31] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed., pp. 396–404. Morgan-Kaufmann, 1990. http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research* **15** (2014) 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

[33] T. Keck, "Multivariate classification." Cern school of computing, 2016.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[35] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. http://tensorflow.org/. Software available from tensorflow.org.

[36] F. Chollet *et al.*, "Keras." https://github.com/fchollet/keras, 2015.

[37] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. P. Lillicrap, "A simple neural network module for relational reasoning," *CoRR* **abs/1706.01427** (2017) . http://arxiv.org/abs/1706.01427.

[38] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[39] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *ArXiv e-prints* (June, 2014) , arXiv:1406.2661 [stat.ML].

[40] G. Louppe, M. Kagan, and K. Cranmer, "Learning to Pivot with Adversarial Networks," arXiv:1611.01046 [stat.ME].

[41] J. Stevens and M. Williams, "uBoost: A boosting method for producing uniform selection efficiencies from multivariate classifiers," *JINST* **8** (2013) P12013, arXiv:1305.7248 [nucl-ex].

[42] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," *ArXiv e-prints* (June, 2012) , arXiv:1206.2944 [stat.ML].

[43] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[44] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. Amer. Math. Soc.* **58** no. 5, (09, 1952) 527–535. https://projecteuclid.org:443/euclid.bams/1183517370.

[45] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition* **30** no. 7, (1997) 1145 – 1159. http://www.sciencedirect.com/science/article/pii/S0031320396001422.

[46] 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." https://github.com/scikit-optimize/scikit-optimize.

[47] D. J. Lange, "The EvtGen particle decay simulation package," *Nucl. Instrum. Meth.* **A462** (2001) 152–155.

[48] **GEANT4**, S. Agostinelli *et al.*, "GEANT4: A Simulation toolkit," *Nucl. Instrum. Meth.* **A506** (2003) 250–303.

[49] "MC7 samples for analysis users." https://confluence.desy.de/display/BI/MC7+samples+for+analysis+users. No public access.

[50] J. Egan, *Signal Detection Theory and ROC-analysis*. Academic Press series in cognition and perception. Academic Press, 1975. https://books.google.de/books?id=V40oAAAAYAAJ.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR* **abs/1412.6980** (2014) . http://arxiv.org/abs/1412.6980.

A. Continuum Suppression Features

A.1. Thrust

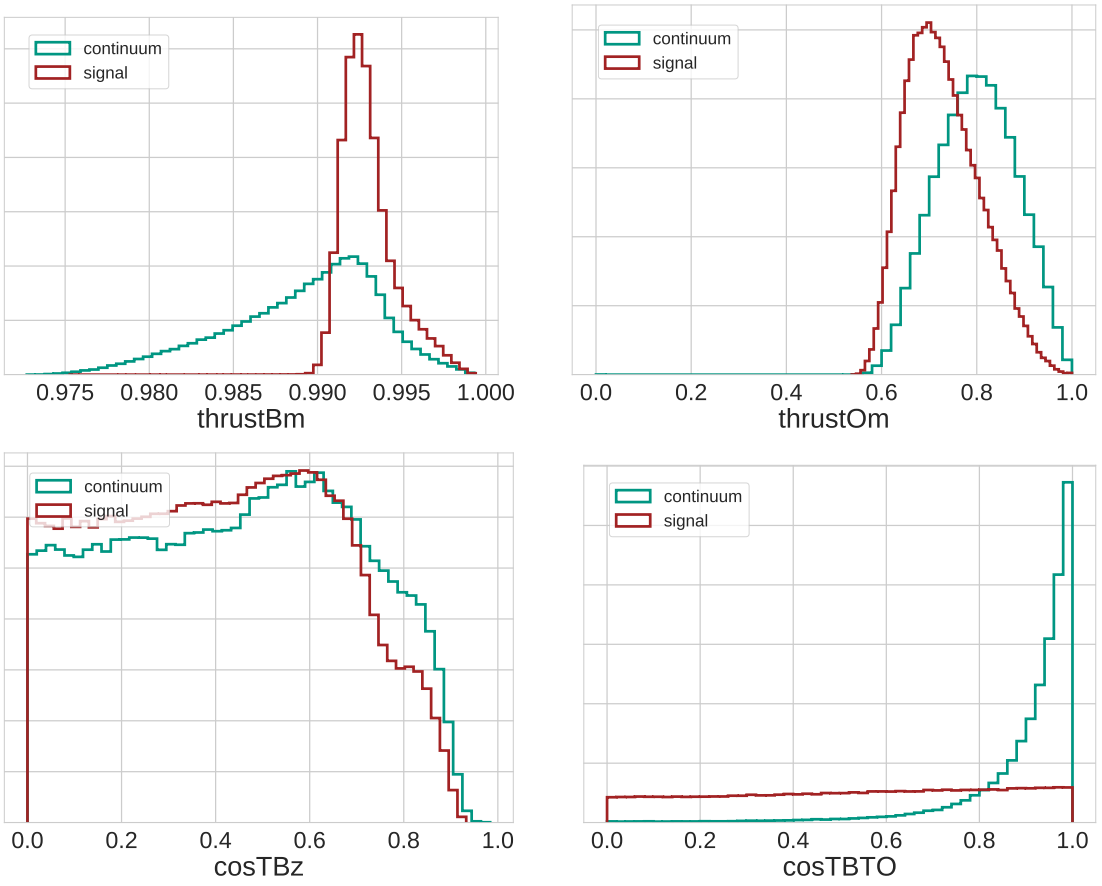

Figure A.1.: Normalized distributions of the thrust axis of the B candidate (upper left) and the ROE (upper right), as wells as the angle between the thrust axis of the B candidate and the beam axis (lower left) and the angle between the thrust axis of the B candidate and the thrust axis of the ROE (lower right). See Section 3.2.1 for details.

A.2. Cleo Cones

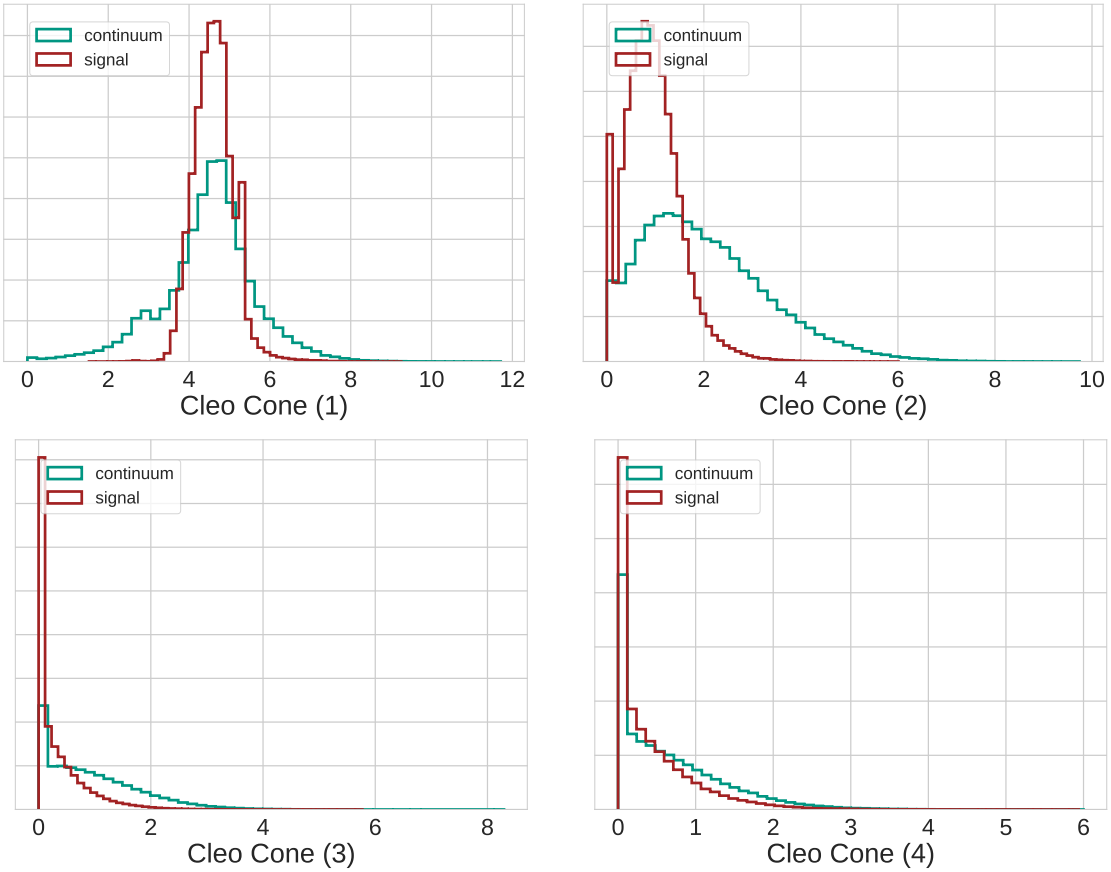

Figure A.2.: Normalized distributions of Cleo Cones 1-4. See Section 3.2.2 for details.

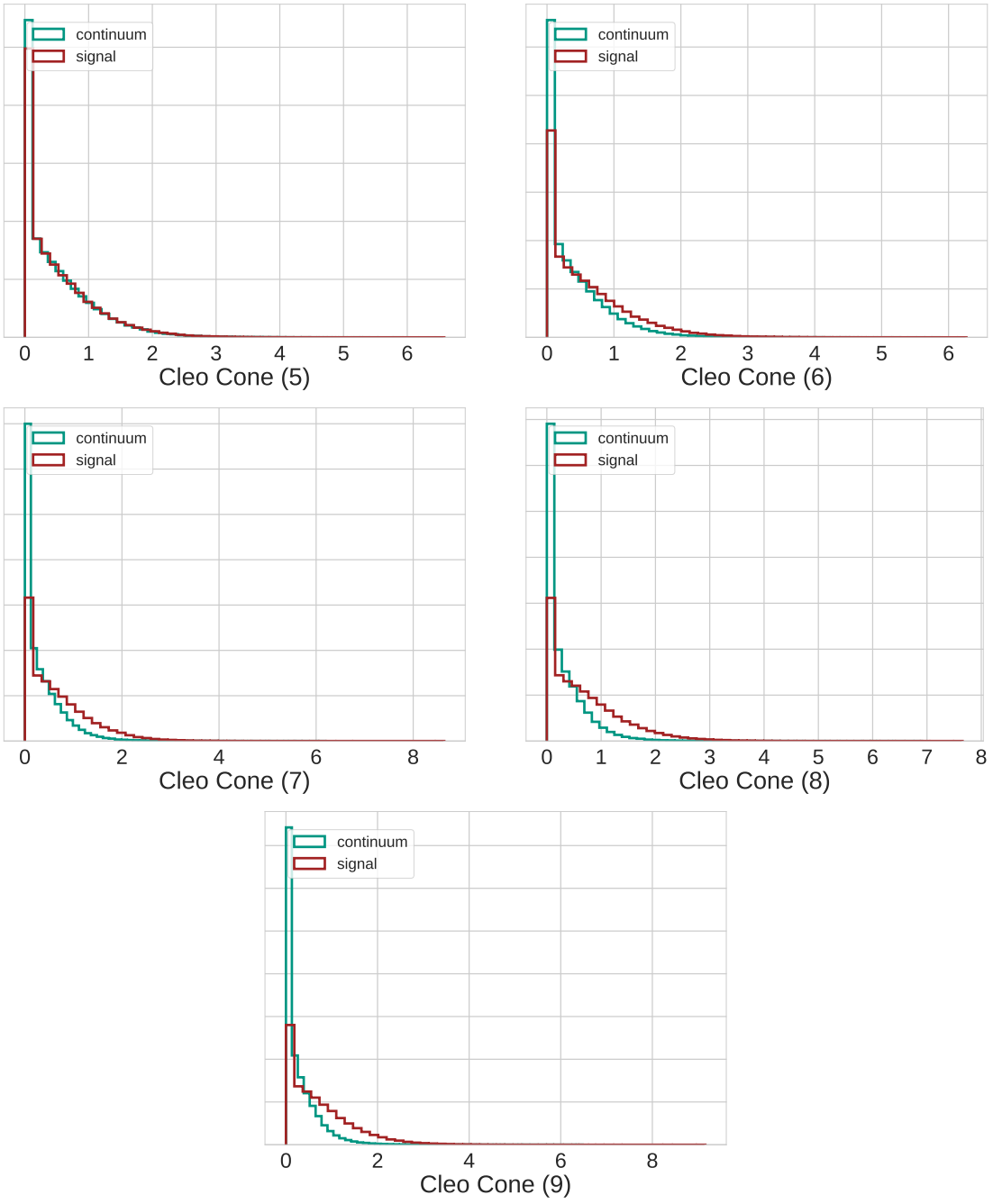

Figure A.3.: Normalized distributions of Cleo Cones 5-9. See Section 3.2.2 for details.

A.3. Fox Wolfram Moments

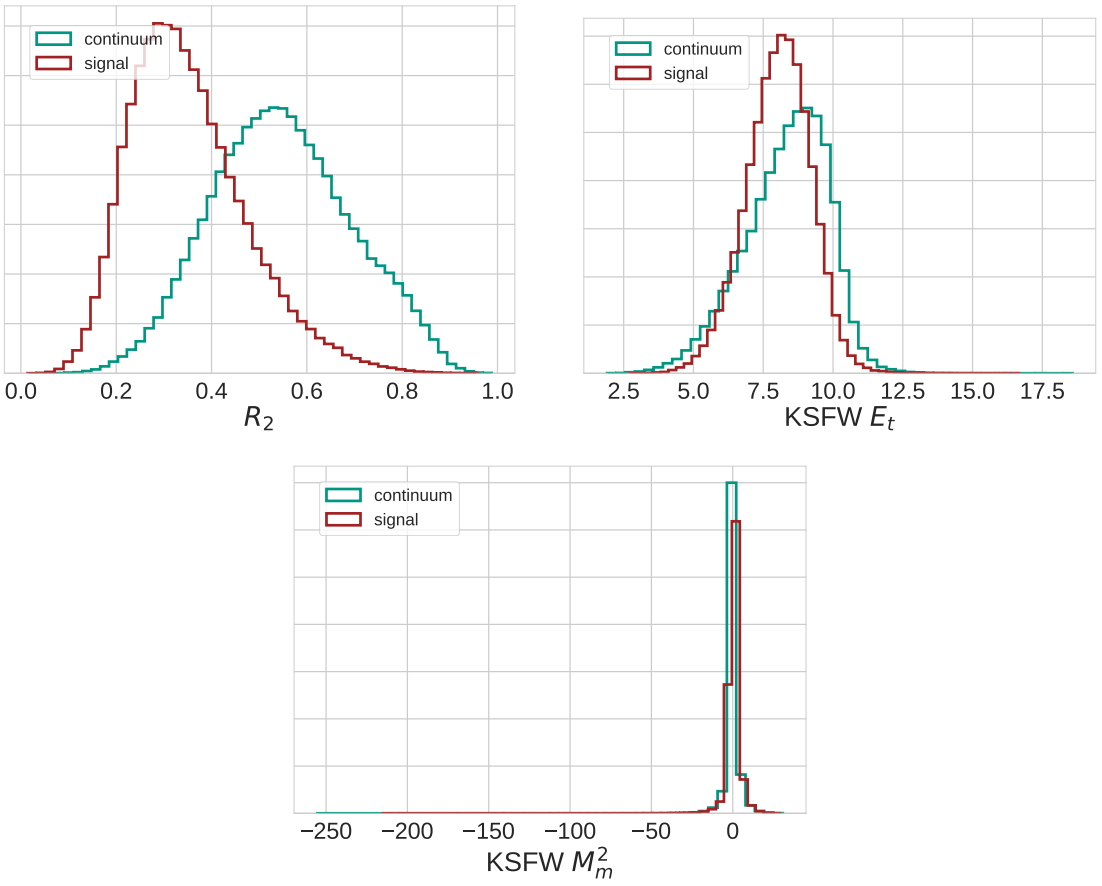

Figure A.4.: Normalized distributions of R_2 and some KSFW moments. See Section 3.2.3 for details.

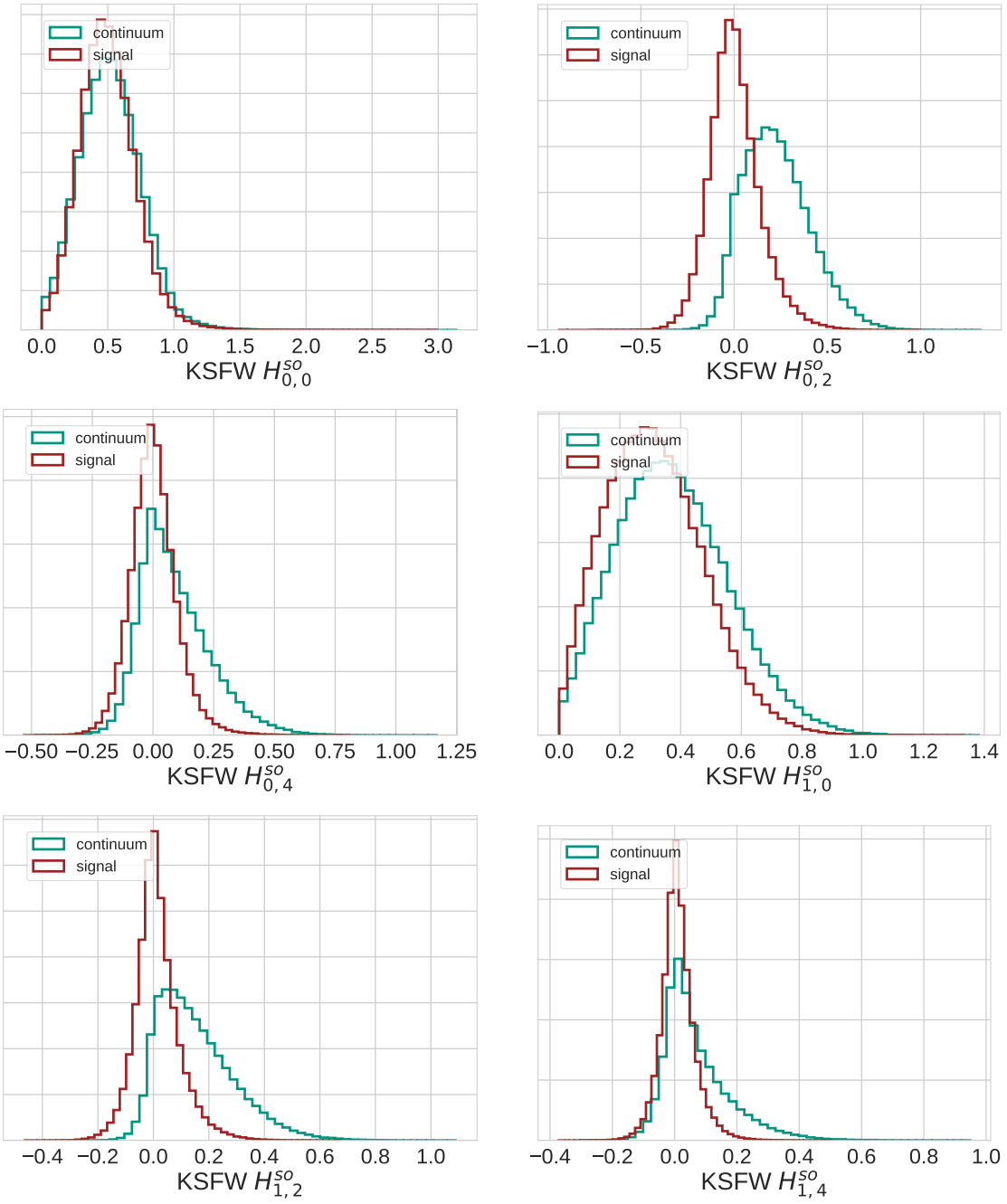

Figure A.5.: Normalized distributions of some KSFW moments. See Section 3.2.3 for details.

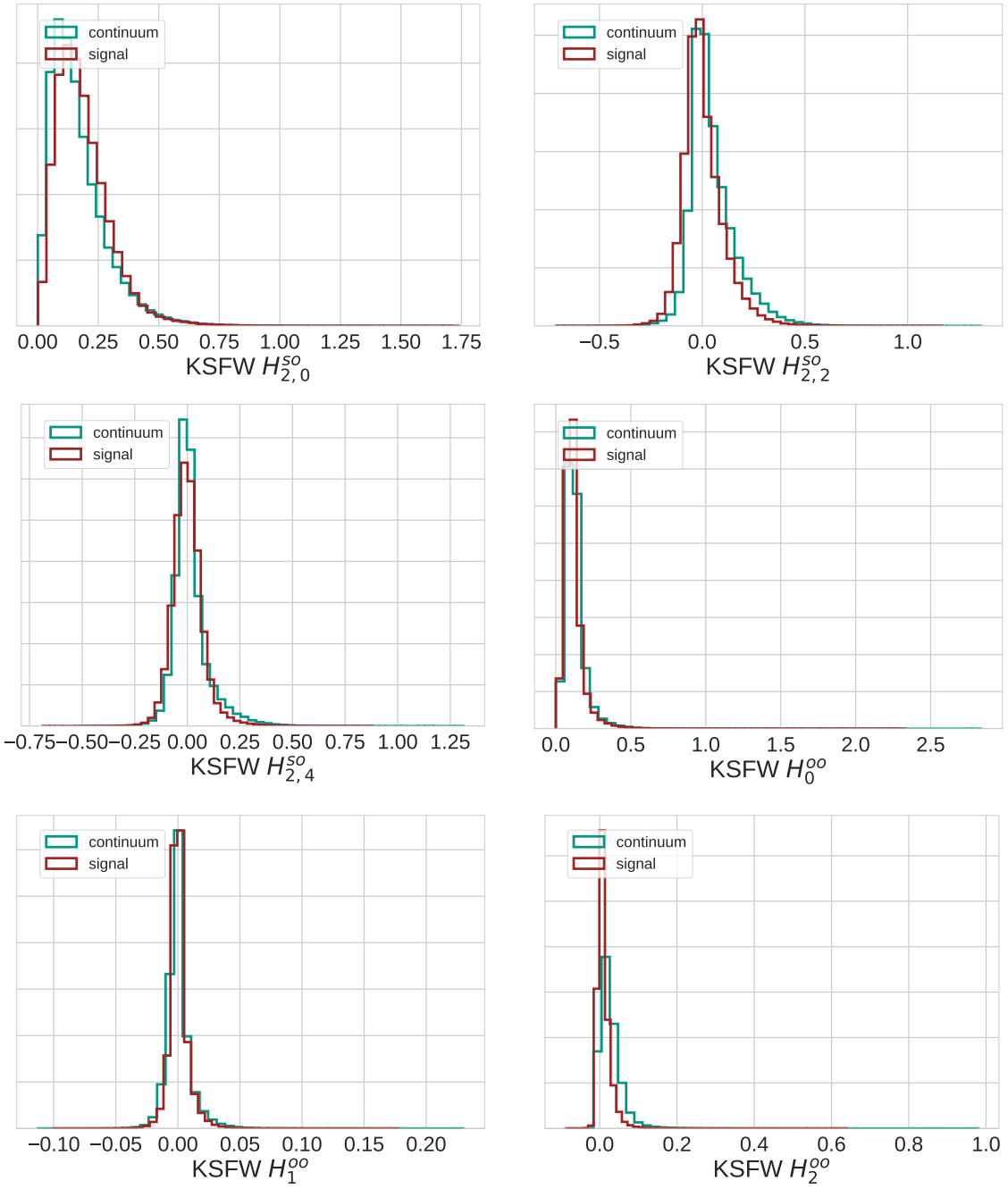

Figure A.6.: Normalized distributions of some KSFW moments. See Section 3.2.3 for details.

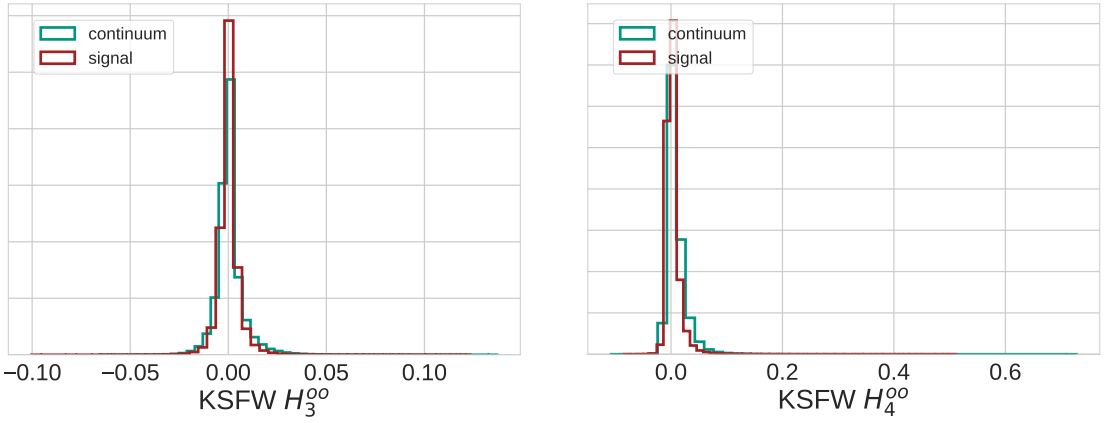

Figure A.7.: Normalized distributions of some KSFW moments. See Section 3.2.3 for details.

B. Hyper Parameter Optimization

B.1. BDT

Table B.1.: Results of the Educated Guess and the ten best hyper parameter sets from the Bayesian Optimization from BDT (E). The hyper parameters are the Number of Trees (NoT), the Depth of each tree (D), the Number of Cut Levels (NoCL), the Rand Ratio (RR) and the Shrinkage (Shr). As the figure of merit, the area under ROC curve (AUC) is chosen. For details see Section 5.3.2.

	NoT	D	NoCL	RR	Shr	AUC
Educated Guess	200	3	8	0.5	0.1	0.971
Rank 1	1000	8	12	0.694	0.135	0.9752
Rank 2	992	7	12	0.772	0.138	0.9752
Rank 3	1000	8	12	0.744	0.137	0.9752
Rank 4	536	8	12	0.785	0.160	0.9749
Rank 5	494	8	12	0.645	0.078	0.9748
Rank 6	1000	4	12	0.707	0.226	0.9747
Rank 7	1000	4	12	0.598	0.197	0.9746
Rank 8	679	5	12	0.743	0.163	0.9746
Rank 9	1000	3	9	0.703	0.300	0.9746
Rank 10	990	4	12	0.860	0.200	0.9745

Table B.2.: Results of the Educated Guess and the ten best hyper parameter sets from the Bayesian Optimization from BDT (E+DL). The hyper parameters are listed in Table B.1. For details see Section 5.3.2.

	NoT	D	NoCL	RR	Shr	AUC
Educated Guess	200	3	8	0.5	0.1	0.9863
Rank 1	1000	5	12	0.880	0.300	0.9951
Rank 2	853	5	4	0.773	0.300	0.9947
Rank 3	484	8	12	0.726	0.300	0.9946
Rank 4	595	6	12	0.718	0.151	0.9946
Rank 5	771	8	12	0.694	0.300	0.9946
Rank 6	423	6	4	0.672	0.300	0.9944
Rank 7	874	6	4	0.522	0.264	0.9942
Rank 8	241	8	12	0.387	0.119	0.9940
Rank 9	296	8	12	0.330	0.300	0.9940
Rank 10	318	8	4	1.000	0.300	0.9938

Table B.3.: Results of the Educated Guess and the ten best hyper parameter sets from the Bayesian Optimization from BDT (E+DL+V). The hyper parameters are listed in Table B.1.For details see Section 5.3.2.

	NoT	D	NoCL	RR	Shr	AUC
Educated Guess	200	3	8	0.5	0.1	0.9930
Rank 1	861	4	11	0.359	0.238	0.9977
Rank 2	956	8	11	0.693	0.278	0.9976
Rank 3	317	5	4	0.687	0.252	0.9975
Rank 4	1000	2	12	0.687	0.300	0.9972
Rank 5	996	2	5	0.656	0.254	0.9970
Rank 6	1000	2	4	0.444	0.300	0.9967
Rank 7	998	5	5	0.950	0.038	0.9967
Rank 8	1000	8	12	1.000	0.194	0.9964
Rank 9	1000	2	4	1.000	0.300	0.9960
Rank 10	1000	6	4	0.972	0.300	0.9957

B.2. DNN

Table B.4.: Results of the Educated Guess and the ten best hyper parameter sets from the Bayesian Optimization from DNN (E). The hyper parameters are the Number of Hidden Layers (NoHL), the Number of Neurons for each Hidden Layer (NoN), the amount of Dropout after each Hidden Layer (Drop) and if the Dropout after the first and last Hidden Layer should be set to zero (RD). As the figure of merit, the area under ROC curve (AUC) is chosen. For details see Section 5.3.3.

	NoHL	NoN	Drop	RD	AUC
Educated Guess	4	100	0.00	True	0.9772
Rank 1	2	159	0.00	True	0.9779
Rank 2	2	304	0.00	False	0.9775
Rank 3	2	50	0.50	True	0.9775
Rank 4	2	348	0.50	True	0.9775
Rank 5	2	50	0.00	True	0.9773
Rank 6	4	100	0.00	True	0.9772
Rank 7	4	97	0.00	True	0.9772
Rank 8	2	592	0.00	True	0.9771
Rank 9	3	493	0.00	False	0.9766
Rank 10	3	322	0.40	True	0.9766

Table B.5.: Results of the Educated Guess and the ten best hyper parameter sets from the Bayesian Optimization from DNN (E+DL). The hyper parameters are listed in Table B.4. For details see Section 5.3.3.

	NoHL	NoN	Drop	RD	AUC
Educated Guess	4	100	0.00	True	0.9948
Rank 1	1	1000	0.48	True	0.9951
Rank 2	7	50	0.00	False	0.9948
Rank 3	1	1000	0.28	True	0.9948
Rank 4	4	100	0.00	True	0.9948
Rank 5	1	966	0.49	True	0.9948
Rank 6	4	50	0.04	True	0.9947
Rank 7	3	50	0.05	True	0.9947
Rank 8	1	1000	0.11	True	0.9947
Rank 9	3	212	0.06	True	0.9947
Rank 10	1	977	0.00	False	0.9947

Table B.6.: Results of the Educated Guess and the ten best hyper parameter sets from the Bayesian Optimization from DNN (E+DL+V). The hyper parameters are listed in Table B.4. For details see Section 5.3.3.

	NoHL	NoN	Drop	RD	AUC
Educated Guess	4	100	0.00	True	0.9970
Rank 1	2	150	0.00	True	0.9976
Rank 2	5	50	0.00	False	0.9976
Rank 3	2	50	0.00	False	0.9975
Rank 4	4	50	0.00	True	0.9975
Rank 5	1	681	0.20	True	0.9975
Rank 6	1	856	0.13	True	0.9974
Rank 7	1	130	0.00	True	0.9974
Rank 8	1	382	0.00	True	0.9974
Rank 9	1	1000	0.34	True	0.9974
Rank 10	6	50	0.00	True	0.9973

Danksagung

Zu aller erst möchte ich mich bei Herrn Prof. Dr. Günter Quast für die Übernahme des Referats und Herrn PD Dr. Andreas Meyer für die Übernahme des Korreferats und für das Korrekturlesen der Arbeit bedanken.

Bei Herrn Dr. Martin Heck bedanke ich mich dafür, dass er mir das Thema dieser Arbeit angeboten hat, sowie für das Korrekturlesen der Arbeit. Für die hervorragende Betreuung und das Korrekturlesen der Arbeit möchte ich mich sehr herzlich bei Herrn Dr. Pablo Goldenzweig bedanken.

Außerdem möchte ich mich für eine hervorragende Betreuung, viele interessanten Diskussionen und das Korrekturlesen der Arbeit sehr herzlich bei Herrn Thomas Keck bedanken. Zudem möchte ich mich bei Herrn Jochen Gemmler für die unzähligen hilfreichen Diskussionen und Anregungen bedanken, von denen diese Arbeit sehr profitiert hat.

Sehr herzlich bedanke ich mich bei allen Mitgliedern der B-Physik-Arbeitsgruppe am ETP für die hervorragende Arbeitsatmosphäre und die umfangreiche Hilfsbereitschaft während der gesamten Arbeit.

Zuletzt möchte ich mich bei meiner Freundin, meinen Eltern und meiner Familie für ihre unermüdliche Unterstützung während meines gesamten Studiums bedanken.

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, November 2, 2017

. . . . . .

(**Dennis Weyland**)