Karlsruhe Institute of Technology

ETP-KA/2021-9

Deep Continuum Suppression with Predictive Uncertainties at the Belle II Experiment

Lars Sowa

Masterthesis

27th July 2021

Institute of Experimental Particle Physics (ETP)

Reviewer:	Prof. Dr. Ulrich Husemann
Second reviewer:	Prof. Dr. Günter Quast

Editing time: 22nd June 2020 – 27th July 2021

Karlsruhe Institute of Technology

ETP-KA/2021-9

Deep Continuum Suppression mit Vorhersage von Unsicherheiten am Belle II Experiment

Lars Sowa

Masterarbeit

27. Juli 2021

Institut für Experimentelle Teilchenphysik (ETP)

Referent:	Prof. Dr. Ulrich Husemann
Korreferent:	Prof. Dr. Günter Quast

Bearbeitungszeit: 22. Juni 2020 – 27. Juli 2021

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, 27. Juli 2021

. . . . . .

(**Lars Sowa**)

Contents

1. Introduction

The Standard Model of particle physics is a highly successful yet unfortunately incomplete description of nature. Thus, one of the most important challenges of modern particle physics is to test this model and to search for new physics in high precision measurements. B mesons are excellent candidates for such tests. The SuperKEKB collider in Japan accelerates electrons and positrons which collide at an energy of 11 GeV aiming to produce B meson pairs. One of the most prominent achievements at SuperKEKB was the contribution to the measurement of the complex phase ϕ of the Standard Model by the Belle collaboration [1, 2]. Today, the successor collaboration Belle II continues and expands this work.

To analyse the properties of a B meson, it gets recombined from its decay daughters. Since there are a lot of background events that mimic the signature of a B meson, it is important to suppress these events. This suppression is called Continuum Suppression and in the Belle II collaboration is performed with traditional machine learning algorithms, like Boosted Decision Trees (BDT). However, today there are newer and more promising deep learning methods available which motivate an extension of the current Continuum Suppression to a Deep Continuum Suppression (DCS). Such methods were tested in previous work, which showed, that BDT can be outperformed by Multilayer Perceptrons (MLPs) [3].

This work aims to further improve the DCS. Specifically, the MLP is used as a starting point to improve three points:

Firstly, an MLP needs a fixed order for input particles. In previous work, this is solved by sorting approaches, but these are potential sources of errors. Therefore, this thesis presents a reliable, self-attention-based input mechanism which allows for invariance under the particle order.

Secondly, to guarantee certainty about the prediction of deep learning models, there is a high interest in models with predictive uncertainties. This is addressed using the concept of Deep Ensembles [4] to predict uncertainties of continuum classifications.

Finally, the use of vertex information for the training of a model leads to a bias in certain analysis variables. Since this could lead to falsification of further studies, a Distance Correlation [5] is used to decorrelate the Continuum Suppression model from third variables.

The structure of this work is as follows: in Chapter 2 an introduction to the Standard Model is given. Chapter 3 gives an overview of the Belle II experiment including its motivation, detector, and the SuperKEKB collider. The Continuum Suppression and its current status

is explained in Chapter 4. Chapter 5 gives an overview of multivariate analysis methods and explains mechanisms to improve the DCS. These mechanisms are tested and discussed in Chapter 6. At the end, a summary and an outlook is given in Chapter 7.

2. The Standard Model

The most essential model for modern particle physics was developed in the latter half of the twentieth century and is known as the Standard Model (SM). With the SM nearly all phenomena in particle physics can be described. Therefore, deviations from the SM are usually the gateway to new physics. Therefore, it is crucial to understand the SM and to determine its parameters by experiments. An overview of the SM is given in Figure 2.1. In general one can distinguish the particles of the SM in fermions and force-carrying bosons. These and their interactions are described below.

Fermions are particles with half-integer spin and are further divided into quarks and leptons. There are three generations of quarks with one positively charged quark (up, charm, top) and one negatively charged quark (down, strange, bottom) per generation. These quark types are often referred to as flavors. Leptons are just like quarks divided in three generations. But in contrast, there is one charged and one neutral particle per generation. The charged particles are the electron, muon, and tau. Each of these has a associated neutrino partner which has no charge.

Bosons are integer spin particles and act as force mediators between particles. The SM includes three of the four fundamental forces of physics: the weak force, the electromagnetic force, and the strong force. The gravitational force is not covered in the SM.

- **The strong force** is described by quantum chromodynamics. It describes the interaction between quarks via gluons, which therefore are the mediators of strong interactions. This force is described by quantum chromodynamics (QCD). Here every quark has a color flavor assigned: red, green, or blue (including anticolors for anti-quarks). Quarks can only be stable in a group with a neutral overall color (for example red and anti-red or red, green, and blue) similar to color combinations in art. Gluons carry more than one color which allows quarks exchanging colors by the strong interaction. If all gluon exchanges in a quark group result in a neutral overall color, then the strong force holds them in a stable state which we see as particles such as protons. These color rules are called confinement. As a consequence of these rules, color charged particles cannot be isolated. If two color charged particles are removed from each others, at a certain point it becomes energetically more favorable to create a new quark-antiquark pair. If a $\Upsilon(4S)$ resonance decays in a back-to-back $b\bar{b}$ pair, then confinement enforces it to produce a $u\bar{u}$ pair for example. These can enter a stable state with the b quarks in form of a B meson pair $B\bar{B}$.

- **The electromagnetic force** couples only to charged fermions, which excludes neutrinos from electromagnetic interactions. The mediator herefore is the photon γ which is massless and can therefore travel with the speed of light. The range of electromagnetic interactions is infinite, but its strength decreases by $\propto \frac{1}{r^2}$.
- **The weak force** is exchanged by $W^{\pm}$ and Z boson couplings within ranges of $\leq 10^{-15}$ m. These bosons couple only on quarks and leptons. By providing charged $W^{\pm}$ boson couplings, the weak force is the only one, which allows quarks to change their flavor with certain probabilities. These flavor transition probabilities are given by the unitary Cabibbo Kobayashi Maskawa (CKM) matrix. In modern flavor physics, it is a central aspect to determine and validate the properties of the CKM matrix and its elements.
- The SM is completed by **the Higgs**, which is the only spin 0 boson and was recently discovered at the Large Hadron Collider in 2012 [6]. The Higgs boson is a excitation of the so-called higgs field. This field allows for massive $W^{\pm}$ and Z bosons, through mutual interactions. This is known as Higgs mechanism. In addition, fermions interact with the Higgs field, as well. This interaction is known as Higgs Yukawa coupling, which also explains massive fermions in the SM.

Currently, we assume that all particles predicted by the SM have been discovered. Nevertheless, there are open questions that cannot be explained by the Standard Model. For example, it is known that there must be Dark Matter in our Universe [8], but such particles could not be observed yet. Another actual example is that experiments measured oscillations in the neutrino sector [9, 10]. Such observations show that there is still a lot of unknown physics that has to be discovered and explained.

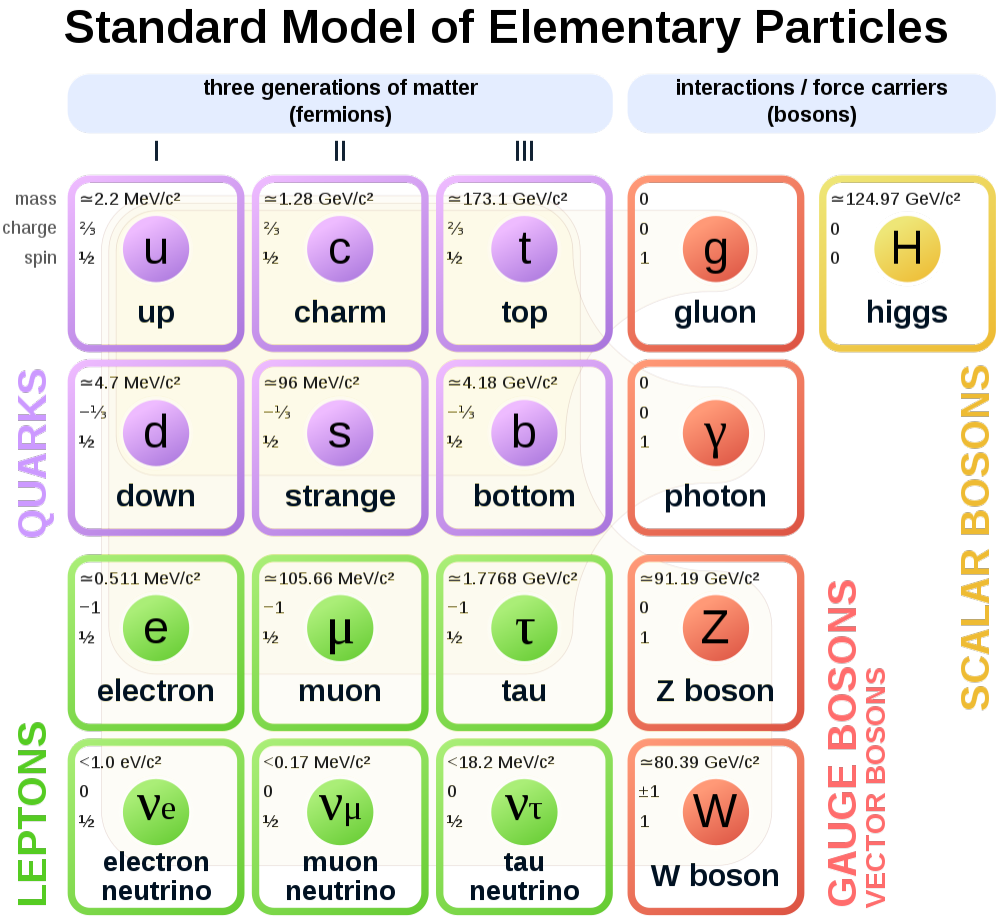


Figure 2.1.: Illustration of the Standard Model. On the left hand side quarks (purple) and leptons (green) are shown. On the right hand side, one can see vector bosons and the scalar Higgs boson. For each particle, the mass (or mass limits), the electromagnetic charge, and the spin of the particle is shown. Taken from [7]

3. The Belle II Experiment

The Belle II experiment is the successor of the Belle experiment and was founded in 2008. The experiment is part of the High Energy Accelerator Research Organization (KEK) which runs the SuperKEKB collider in Tsukuba, Japan. The Belle II experiment started its data taking in 2018 and aims to achieve an integrated luminosity of $50\,\mathrm{ab}^{-1}$ which would be 50 times more data than the Belle experiment collected.

This chapter gives a short introduction to the Belle II experiment. Section 3.1 gives a short review of motivation of the Belle and Belle II experiments. The SuperKEKB collider is explained in Section 3.2 and Section 3.3 gives a short overview over the Belle II detector. Section 3.4 gives a short insight into the Belle II analysis software framework.

3.1. Motivation

In 1964, Cronin and Fitch published a groundbreaking observation of charge parity violation (CPV) in a kaon system [11]. This result was not compatible with the SM back in the days, because it only counted two quark generations, and only three quarks were experimentally discovered (up, down, strange). This changed in 1973 when Kobayashi and Maskawa published their approach of a SM with three quark generations, which introduced a complex phase in the SM allowing for CPV [12]. After the discoveries of the charm and bottom quark, there was a high interest to verify the theory of Kobayashi and Maskawa by measuring the complex phase ϕ_1 of the CKM matrix. To do this one needs to evaluate the interference of an oscillating meson system. At that time, the B meson, consisting of a bottom quark and a lighter quark, had been recently discovered. This provides an ideal candidate to measure such interference. Among other things, the B meson has a relative long lifetime which is needed in a CPV study to resolve the path differences between the two decaying B mesons.

Taking advantage of such B mesons the Belle [1, 2] and BaBar [13] experiments were able to measure the complex phase ϕ_1 in 2008, whereupon Kobayashi and Maskawa received the Nobel prize in physics. In the same year, the Belle II collaboration was founded and the detector upgraded, aiming for 40 times higher luminosity than its predecessor. Achieving this high luminosity at a low energy scale makes the collaboration the frontier of precision measurements of rare decays. This allows for an effective search for new physics like B anomalies, axion-like particles in the dark sector, or lepton flavor violation [14, 15].

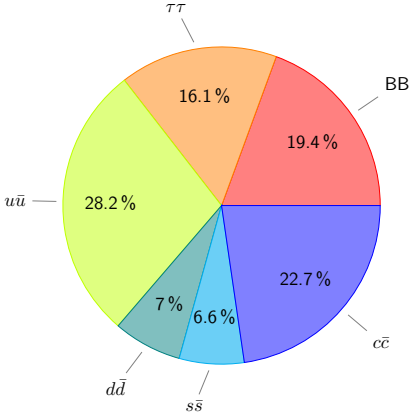


Figure 3.1.: Relative and hadronic cross sections of e^-e^+ collisions at the $\Upsilon(4S)$ resonance. Data from [18]

3.2. SuperKEKB

SuperKEKB [16] is an electron-positron collider located in Tsukuba, Japan. This collider is an upgrade of the former KEKB collider [17], which was constructed for the Belle experiment and had its first collisions in 2018. SuperKEKB is equipped with a linear collider to accelerate electrons and positrons and inject them to two storage rings. The low energy ring (LER) stores positrons with an energy of 4 GeV while the high energy ring (HER) stores electrons at 7 GeV. The beams are supposed to produce $e^- + e^+ \rightarrow \Upsilon(4S)$ mesons and therefore collide at its production threshold of 10.58 GeV [18]. Important background processes that need to be suppressed are shown in Figure 3.1 and discussed in Section 4.1. Due to the high decay width of $\Upsilon(4S) \rightarrow B\bar{B}$ above 96 %, SuperKEKB is a typical B-factory.

The asymmetry of the beam energies leads to a boosted center-of-mass (CMS) frame of the produced $\Upsilon(4S)$ which facilitates the measurement of the decay vertex difference of the B mesons.

In 2020, SuperKEKB ran at a luminosity of $3.1 \times 10^{34} \mathrm{cm}^{-2}\mathrm{s}^{-1}$ which is the highest achieved by a collider until then [19].

3.3. The Belle II Detector

The Belle II detector is built in several layers around the interaction point as illustrated in Figure 3.2. Due to the asymmetric beam energies which result in a boosted CMS reference frame, the detector design is asymmetric as well. In the following, the detector parts are described from the inside to the outside. More detailed information can be found in [20].

The two-layer pixel detector (**PXD**) is located 14 mm around the interaction point and is based on the DEPFET [21] technology. To get a high resolution of charged-particle tracks, its silicon pixels are sized between $50 \times 50\,\mu$m. The track reconstruction is additionally

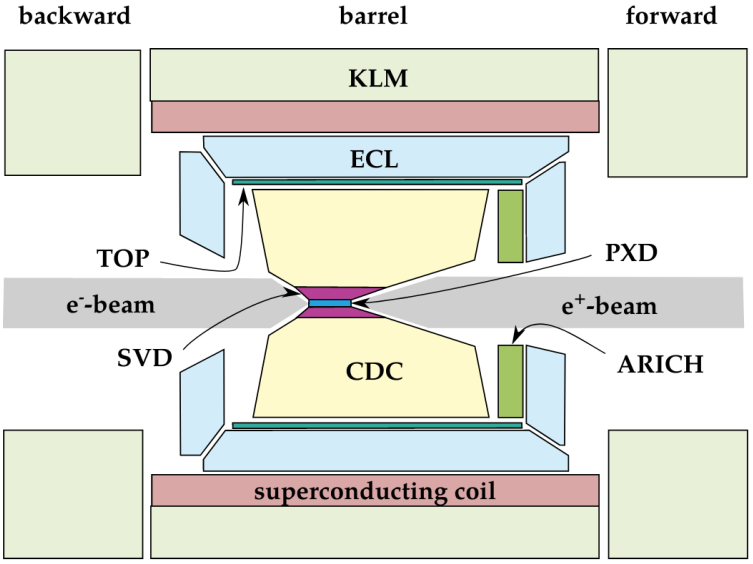


Figure 3.2.: Illustration of the Belle II detector. Taken from [22]

supported by the silicon vertex detector (**SVD**) based on double-sided strip detectors. Together PXD and SVD build the vertex detector (VXD).

The central drift chamber (**CDC**) is used to determine trajectories. To do so the detector is filled with a Helium-Methane mixture while its wires are arranged in the z-direction (axial layer). To derive information about the z component of the particles, every second layer is slightly rotated (stereo layer).

The time-of-propagation detector (**TOP**) and the Aerogel Ring-Imaging Cherenkov detector (**ARICH**) are used for the particle identification (PID), especially for the discrimination of kaons and pions. The TOP detector is in the barrel region and uses patterns in reflected Cherenkov photons for the PID. In contrast to that, the ARICH detector is located in the endcap and relies on direct measurement of the Cherenkov angle.

The electromagnetic calorimeter (**ECL**) consists of thallium-doped cesium iodide CsI(Tl) crystals to detect electromagnetic clusters from photons and electrons. To capture as many photons as possible the ECL consists of a barrel part, a backward and a forward endcap.

The ECL is wrapped in a **superconductiong coil** generating a magnetic field of 1,5 T to force charged particles in a curved trajectory which allows a deduction of their momentum and charge.

The K_L^0 and muon detector (**KLM**) is the outermost detector part consisting of alternating layers of iron plates and active material specialized to measure hadronizing K_L^0. Since muons are minimal ionizing particles they are the only charged candidates that traverse the ECL and can be identified by their charged track in the KLM.

3.4. Belle Software Framework

To analyze Belle II data usually the Belle II Analysis Software Framework II (BASF2) [23] is used. The software packages are written in C++ and Python including third-party code like ROOT [24]. The collected Belle II data during a Monte Carlo (MC) campaign or a detector run is saved as a dataset consisting of raw track, cluster, and other detector information.

In order to reconstruct particles from a dataset the user has to create a Python steering file. By importing BASF2, a path is created to which single analysis modules can be appended. For example, particle candidates can be reconstructed from the dataset and cuts can be applied. Such an analysis path gets executed by BASF2 for each event in the dataset. Thanks to this method BASF2 allows effective usage of complicated analysis techniques with minimal effort for Belle II members.

During the reconstruction, there are often many events that mimic the signal process and need to be filtered out. The main source of these are so-called continuum events that should be suppressed in analyses. To do so BASF2 includes a continuum suppression module that currently works with Boosted Decision Trees. However, in the actual machine learning field, newer and more promising methods are available. In the course of this work, some new methods are explained in Section 5.2, Section 5.5, and Section 5.6 and presented in Chapter 6.

4. Continuum Suppression

During the reconstruction of a signal process in a Belle II analysis, incorrect recombinations are unavoidable and often dominate over the actual decay process of interest. These events are called background events and have a similar detector signature to signal events. Therefore, they are difficult to suppress. Since it is the aim of a physics analysis to achieve the highest signal purity possible it is crucial to suppress these events. To do so, a so-called Continuum Suppression (CS) is performed. This chapter is based on the information of [15, 25].

4.1. Continuum

SuperKEKB runs at the $\Upsilon(4S)$ resonance to produce as many B meson pairs as possible. However, not every electron-positron collision produces an $\Upsilon(4S)$. Even after signal reconstruction, there are many events whose final states mimic the kinematic signature of a signal event and do not contain B meson pairs. It is difficult to identify these events, however, it is necessary to suppress them in order to achieve high signal purities.

The biggest background contribution comes from Bhabha scattering $e^+ + e^- \to \gamma \to e^+ + e^-$ with a cross section 300 times higher than the one for $\Upsilon(4S)$ production [15]. Bhabha scattering and other lepton pair productions like $\mu^+\mu^-$ are cut away by the trigger system with high efficiency.

Background events that cannot be suppressed by the trigger are suppressed by the Continuum Suppression. These contributions arise from $e^+ + e^- \to q\bar{q}$ ($q = u, d, c, s$) light flavor productions. As mentioned, lepton-pair-productions are negligible for the Continuum Suppression.

Taus are the heaviest leptons and the only ones who can decay hadronically, for example $\tau^- \to \nu_\tau W^-(\to q\bar{q})$. Therefore, taus contribute to the hadronic background and have to be included in the Continuum Suppression.

The total hadronic cross section of $e^+ + e^-$ collisions is shown in Figure 4.1. Figure 4.1 shows the abundance of continuum events if one compares the height of the $\Upsilon(4S)$ peak with the continuum. The relative cross sections for all events considered for the continuum components are shown in Figure 3.1.

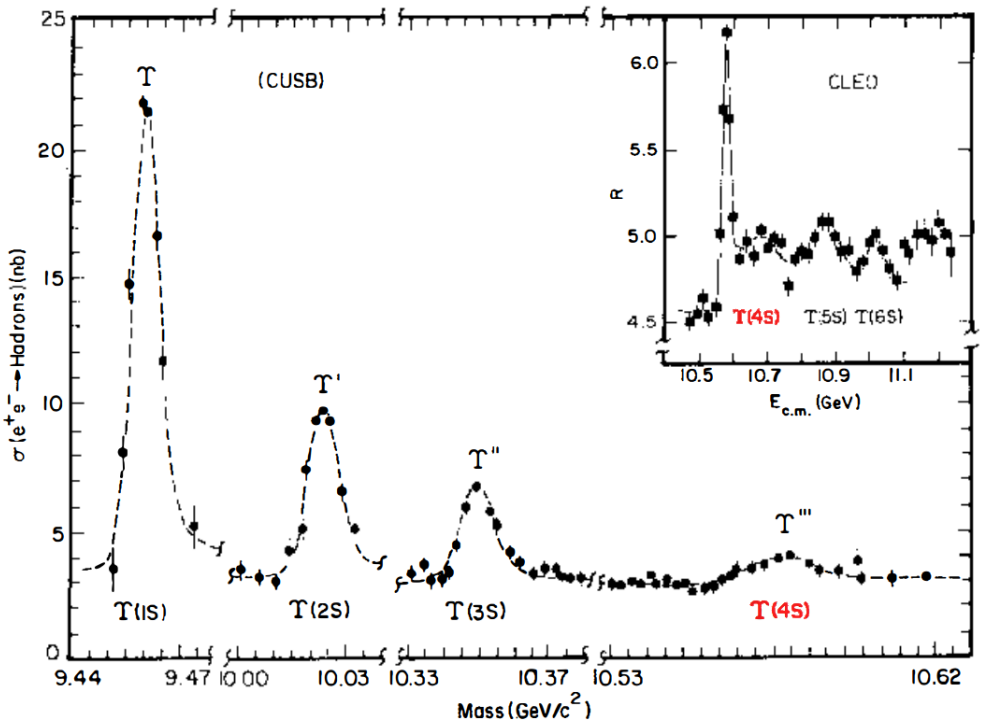


Figure 4.1.: Hadronic cross section for $e^+ + e^-$ collisions. The plot shows the bottonium resonances $\Upsilon(1S)$, $\Upsilon(2S)$, $\Upsilon(3S)$, and $\Upsilon(4S)$ (marked in red) measured by the CUSB experiment. The upper right part shows an additional plot for the resonances measured by the CLEO experiment. Here R is defined as $R = \frac{\sigma_{\text{had}}}{\sigma_{\mu\mu}}$. Adapted from [26]

4.2. Variables

At Belle II, machine learning models, most commonly Boosted Decision Trees, are used to suppress continuum events. The variables used to train such models are explained in the following sections. In general, one differentiates between engineered variables (see Section 4.2.1 and detector level variables (see Section 4.2.2). The engineered variables are handmade and are engineered, such that they describe the features of an event which allow to differentiate between continuum and $B\bar{B}$ event. The detector variables on the other hand are not engineered and contain detector data, such as momenta or angle information.

4.2.1. Engineered Variables

A significant difference between signal and continuum events lies in their event shapes. This is illustrated in Figure 4.2. At SuperKEKB, the electron-positron pairs collide at a fixed energy of 10.57 GeV. Since this energy is at the production threshold of $\Upsilon(4S)$, it is produced nearly at rest. Therefore, the decaying B mesons have relatively low momenta of $p(\text{B}) \approx 0.3\,\text{GeV}$. In contrast, the fermion pairs produced in continuum events have lower masses. Due to energy conservation, this leads to higher momenta of the daughter particles $p(\text{q}) \approx 5\,\text{GeV}$. This difference in the momenta is apparent if one compares the event shapes

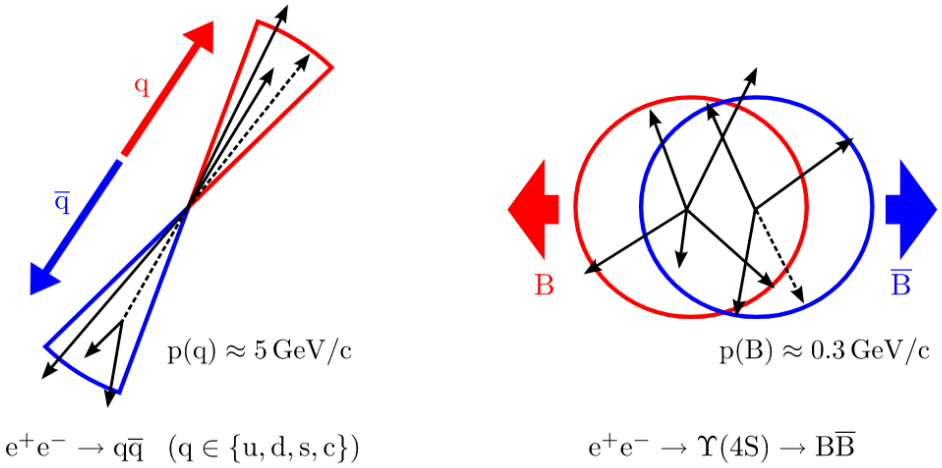


Figure 4.2.: Comparison of continuum $q\bar{q}$ (left) and signal event (right) shapes. Continuum events have a characteristic back to back shape with relative high momenta $p(\mathrm{q}) \approx 5\,\mathrm{GeV}$. Signal events usually have a spherical shape where the decay products have a momentum of $p(\mathrm{B}) \approx 0.3\,\mathrm{GeV}$. Taken from []

of signal and continuum events. Since the B mesons are nearly at rest, their daughters have no preferred direction and therefore decay spherically. In contrast, the two fermions in continuum events have a clear back-to-back structure due to their high momenta and momentum conservation.

To describe these different event shapes, the Belle and BaBar collaborations developed a set of engineered features [25]. Some of these variables relate to the reconstructed B meson. Others take only particles into account, that are not used for the signal candidate's reconstructions. These remaining particles are in the so-called Rest of Event (ROE). The following sections give a short introduction to the event shape variables.

4.2.1.1. Thrust

The thrust concept originates from jet physics [28] but is useful for B physics as well. For an ensemble of particles, the thrust $\vec{T}$ is defined as the axis which maximizes their total momentum projection, according to

$$T = \frac{\sum_{i=1}^{N} \left| \vec{T} \cdot p_i \right|}{\sum_{i=1}^{N} |p_i|}. \tag{4.1}$$

This allows calculating a separate thrust axis for the daughters of the B candidate and the corresponding ROE particles of an event. The magnitude of these two axes is a commonly used variable. Furthermore, the cosine between the thrust axis of the B candidate and the one of the ROE particles is used, as well as the cosine between the thrust axis of the B candidate and the z axis.

4.2.1.2. CleoCones

CleoCones (also CLEO Fisher discriminants) are nine variables that describe the momentum flow around the thrust axis of the B candidate. The first cone includes the momentum flow within a 10° opening angle. Each subsequent cone covers the next 10° of opening angle, resulting in a total of nine cones. The concept of CleoCones was introduced by the CLEO collaboration [29] and was originally used for charmless B decays.

4.2.1.3. Fox Wolfram Moments

Fox and Wolfram introduced in [30] the so-called Fox Wolfram Moments (FW) to describe event shapes in e^+e^- annihilations. These moments are defined as

$$H_k = \sum_{i,j}^{N} |\vec{p}_i| \, |\vec{p}_j| \, P_k(cos\theta_{ij}) \tag{4.2}$$

$$R_k = \frac{H_k}{H_0} \tag{4.3}$$

where $\theta_{i,j}$ is the angle between $\vec{p_i}$ and $\vec{p_j}$ and P_k the k-th Legendre polynomial. Since the limit for vanishing particle masses is $H_0 = 1$, the FW moments are often normalized by Equation (4.3). In the case of two strongly collimated jets, this would result in R_k values close to zero for odd values of k and to one for even values of k.

Furthermore, the Belle collaboration extended the FW moments to so-called Kakuno-Super-Fox-Wolfram (KSFW) moments $H^g_{l,c}$ [25]. The index g splits these into two categories: KSFW moments where one sum runs over reconstructed B meson daughters are indicated with $g = so$ and on the other hand moments where both sums run over ROE particle candidates are indicated by $g = oo$. Furthermore the index c denotes whether a particle is charged ($c = 0$), neutral ($c = 1$) or missing ($c = 2$). A detailed description of KSFW moments can be found in [25].

4.2.2. Detector Level Variables

The event shape variables explained in Section 4.2.1 are per-event variables. In contrast to that, detector level (DL) variables are per-particle candidate variables. Thereby, only final state particles are meant. These variables give information based on calorimeter clusters and tracks. DL variables can be separated into the following groups.

Momentum variables
: for tracks and clusters. These include the magnitude of the momentum, p, as well as the azimuth angle ϕ, the cosine of its polar angle $\cos(\theta)$, and their errors.

Cluster variables
: originate from ECL information. Namely, the number of ECL crystal hits *NHits* and the timing of the cluster *Timing*. Furthermore, the detector region of the cluster and the energy ratio between the innermost nine crystals of the cluster and the outer 21 *E9E21* are used for the Continuum Suppression. Momentum variables are also used.

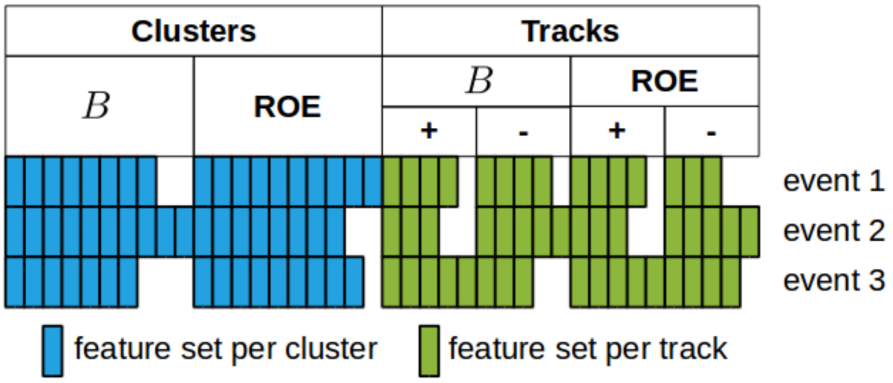


Figure 4.3.: Cluster and track candidate selection for the Deep Continuum Suppression. The feature sets of cluster and track candidates are sorted according to their momenta. For clusters, the first ten daughters of the B candidate and the first ten particles in the ROE are kept. The same procedure is applied for tracks for which the charge is also considered. Hence, the first five positive and negative daughters of the B candidate are kept. The ROE particles are handled in the same manner. Taken from [3]

Track variables
are derived from the track fit. These are the kaon, electron, muon, and proton identification probabilities *kaonID*, *electronID*, *muonID*, and *protonID*. Additionally, the number of hits in the CDC *nCDCHits* and the P-Value *pValue* of the fitted track is used. Momentum variables are also taken into account.

Vertex variables
are a subgroup of track variables and are listed separately because they can increase the accuracy of a classifier but can cause unwanted correlations between the classifier output and other vertex related variables. Available variables are the azimuth angle ϕ and polar angle θ, the azimuth angle between the vertex and the interaction point $d\phi$, the distance between the interaction point and the particle decay vertex *distance* as well as the distance between the two decaying B meson candidates.

4.3. Previous Work on Continuum Suppression

The fast-developing field of machine learning offers many techniques, which can be tested and applied for the Continuum Suppression. The current method used within the Belle II experiment is based on the FastBDT module, an implementation of Boosted Decision Trees [31]. As a result of the growing field of deep learning methods, the performance of Deep Neural Networks, as well as an Adversarial Network and a relation network was tested for a *Deep* Continuum Suppression in [3].

To perform a DCS a Deep Neural Network is used. In contrast to per-event variables (event shape variables), per-particle variables (detector level (DL) and vertex (V) variables) cannot be fed into a Neural Network in an arbitrary order. To solve this, all particle candidates

in an event are sorted according to their momenta. Then, the candidates are separated in groups for clusters and tracks, where the first few candidates are kept. The exact grouping is explained by Figure 4.3. Through this procedure, the cluster and track candidates can be fed with a fixed order into the network. This approach is performed with different variable sets: engineered (E), detector level (DL), and vertex level variables (V). A comparison between the performance of DNNs and BDT is made by the DCS and shown in Figure 4.4a. As one might expect, both DNNs and BDT perform better with more information, i.e. more input features provided. The DNNs outperform the BDT in each category, but the advantage of DNNs decreases as more variables are added to the training.

Even if the sorting approach of the DCS solves the underlying problem of an input order, it does not provide an optimal solution. A disadvantage is that it is vulnerable to variations in the particle order. Such variations may arise from detection inaccuracies or from deviating momentum distributions within the event. Furthermore, the fine groupings of the candidates harbor the risk of throwing away important candidates in one of these groups.

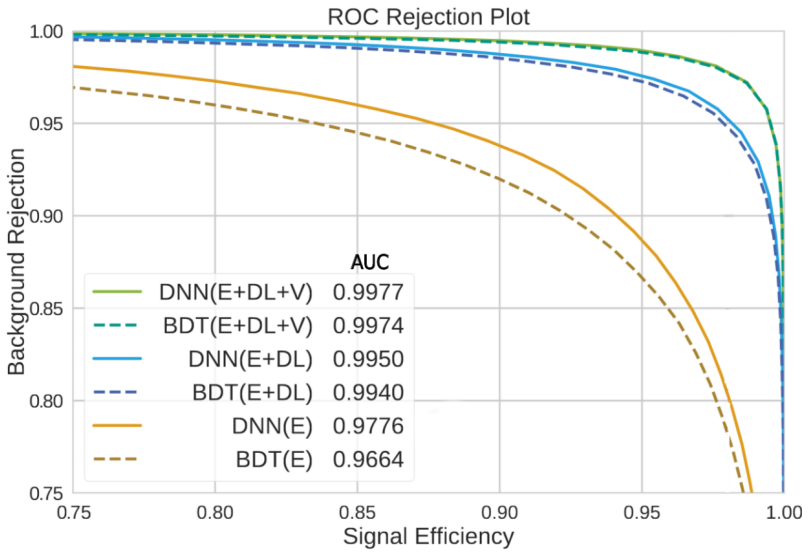
It is a well-known problem in the Continuum Suppression, that the use of vertex variables during the training leads to a bias with the classifier output. This occurs most prominently in the Δz distribution. Δz is the path difference between the two decaying B mesons within a signal event, as illustrated in Figure 4.5a. The correlation impact on the Δz distribution is plotted in Figure 4.5b. Such impacts on variables are not desirable, especially if the affected variables are of importance for the underlying analyses. In the case of Δz in CPV studies, for instance, such a bias affects the analyses result.

To address the classifier's correlation with vertex variables, [3] also investigated a decorrelation mechanism with Δz as a use case. To do so, an Adversarial Network (AN) approach is chosen. This network type origins from Generative Adversarial Networks (GANs) [32]. To train a decorrelated classifier, the AN takes the classifier output as input and tries to predict the shape of Δz. The higher the correlation between the output and Δz, the easier the prediction of the shape of Δz. If the AN is able to guess the shape of Δz, then the classifier gets penalized during the training.

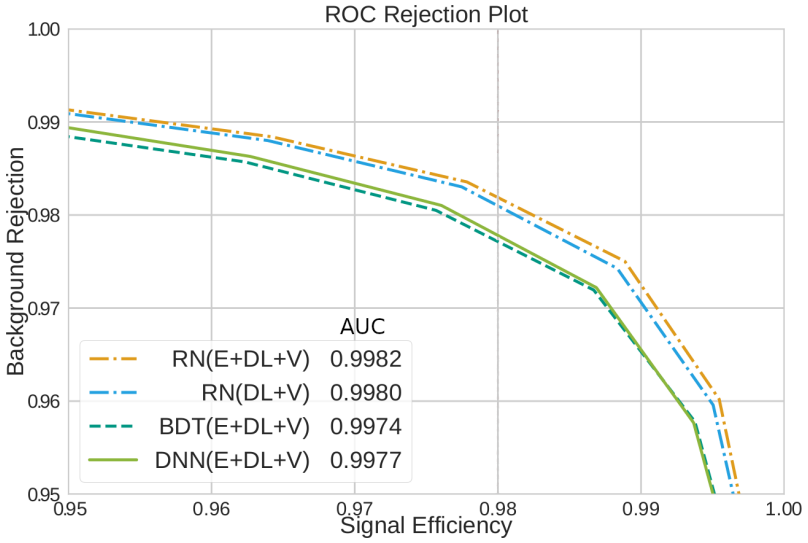
However, such a GAN comes with many additional hyperparameters which need to be adjusted in order to achieve suitable decorrelation results. Moreover, the use of a GAN during the training drives up the training time and the computing requirements. In general, simpler methods are preferable.

Finally, a so-called relation network (RN) [33] is tested in [3]. Such RNs are based on the concept of shared weights. For this, the input features have to be organized into groups with representative meaning, i.e. one feature group for each particle candidate. Each of these feature groups is set in relation to each other feature group by a Multilayer Perceptron that takes both groups as input. This procedure avoids the need to order particle candidates. The performance of RNs with different variable sets is shown in Figure 4.4b. The RN achieves the highest ROC AUC score of 0.9982 compared to the BDT and DNN using all variable sets.

In general relation networks are the predecessor of self attention approaches (see Section 5.2.1) and are now considered outdated compared to more modern methods.

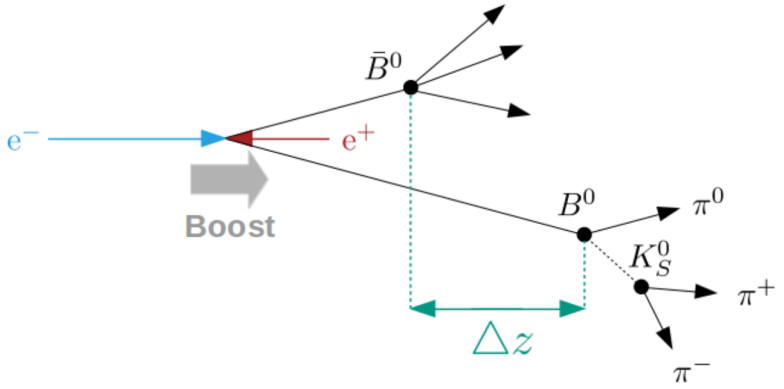


(a) Comparison between BDT and DNNs. Taken from [3]

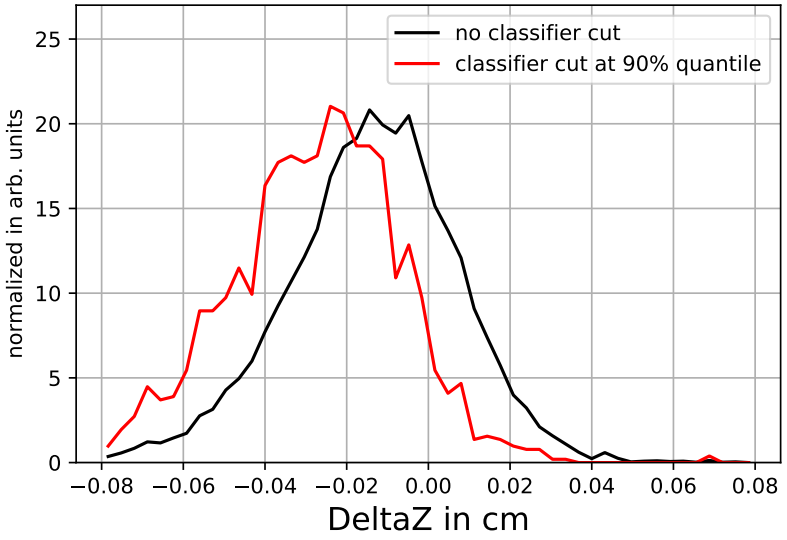


(b) Comparison between BDT, DNNs and ANs. Taken from [3]

Figure 4.4.: Comparison of Receiver Operating Characteristic (ROC) curves. The concept of ROC curves is explained in Section 6.3. Shown are curves for engineered (E), detector level (DL) and vertex (V) variable sets. Additionally the corresponding area under curve (AUC) values are given.



(a) Δz is the measured difference between the decay vertices of two B mesons. Taken from [3]



(b) Δz distribution. The black line shows Δz without any classifier cut applied. The red line shows the distribution with a classifier cut where 90 % of the signal is cut away. Both curves are normalized. As classifier a Three Streamer model (see Section 5.3) is used. Since the distributions show only signal events, the edgy distribution is due to low statistics.

Figure 4.5.

5. Multivariate Analysis Methods

In the last decades, knowledge in the field of particle physics has developed rapidly. Over time, more elaborate experiments were designed and more and more data was needed to gain new knowledge. Modern colliders are optimized to take as much data as possible and therefore make particle physics a highly data-driven field. Consequently, it is necessary to develop and test new and even better analysis methods to learn as much as possible from the collected data. In the case of Continuum Suppression, traditional machine learning approaches are the current state of the art, but there are already promising deep learning approaches (see Section 4.3). This work uses deep learning methods to give a prediction of whether an event is continuum or not. To do so, variables explained in Section 4.2 are used and evaluated by Neural Network models with complex architectures.

This section gives a short overview of deep learning and explains the analysis methods used in the scope of this work. Section 5.1 gives an overview of Neural Networks and their training. Section 5.2 explains the attention mechanism, which is used for the Three Streamer model in Section 5.3. To predict uncertainties, the concept of Deep Ensembles is described in Section 5.5. Finally, a decorrelation method is explained in Section 5.6 to provide background for mitigating bias in Section 6.7.

5.1. Neural Networks

Driven by increasing computing capacity in recent years, there was an enormous development in the field of artificial Neural Networks (NNs). Such a network mimics the arrangement of individual cells found in a brain, that are able to exchange input and output signals, like in a biological brain. This chapter gives a short introduction to artificial Neural Networks. For this, the most prominent network type, the Multilayer Perceptron, is explained in detail.

The concept of artificial neurons was introduced in the field of neuroinformatics by Frank Rosenblatt [34], who invented the so-called Rosenblatt Perceptron. For that, he assumed input values x_i for his modeled cell which are multiplied by weights ω_i. These weights control the contributions of each input x_i and have to be well-chosen (this procedure is explained for modern NNs in Section 5.1.1). Finally the weighted inputs $\omega_i \cdot x_i$ are summed, shifted by a bias β, and transformed by an activation function f. In the case of the Rosenblatt Perceptron, a simple step function (see Figure 5.2c) is used to produce either

one or zero as an output value. The final output y is

$$y = f\left(\sum_{i=1}^{N} x_i \cdot \omega_i + \beta\right). \tag{5.1}$$

In order to produce smooth output values, modern Neural Networks usually use activation functions like Rectified Linear Unit (ReLu) (see Figure 5.2b) for their neurons. Such a neuron builds the basis for most Neural Network based models and illustrated in Figure 5.1a.

A Multilayer Perceptron is organized in multiple layers consisting of artificial neurons, as illustrated in Figure 5.1b. The input layer comes first and takes measured data as input. Each neuron in the input layers produces its own output based on its weights and bias. These outputs are used as inputs for each neuron of the so-called hidden layer. An MLP can have multiple hidden layers with an arbitrary number of neurons. The values are passed analog through each layer. This feedforward structure is characteristic of MLPs, which is why their layers are also called feedforward layers. Finally one can adapt equation Equation (5.1) for the j-th neuron output $y_j^{(k)}$ in layer k

$$y_j^{(k)} = f\left(\sum_{i=1}^{N} y_i^{(k-1)} \cdot \omega_{ji}^{(k)} + \beta_j^{(k)}\right), \tag{5.2}$$

where y_j^0 are the input variables. The layer with the highest k is the output layer, which produces a final MLP output.

The structure of the output layer must be adapted to the learning task. For a multinominal classification, the MLP would have multiple output neurons, one for each class respectively. For binary discrimination, only one output neuron is needed.

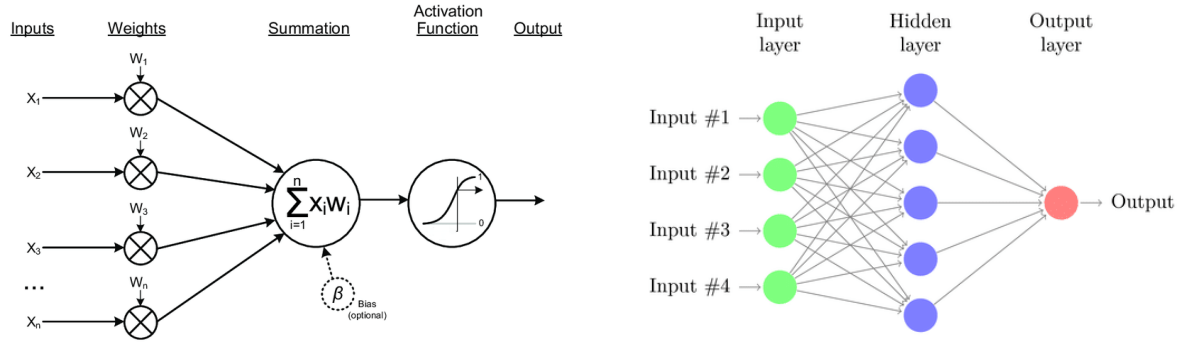
Furthermore, the activation function of the output layer must be well chosen. Classification problems need probability similar output values between one and zero: under those circumstances a sigmoid function is suitable (see Figure 5.2a). In order to be able to predict all values in the solution space, regression tasks do not need an activation function.

In order to give good predictions with an MLP, a suitable number of layers and neurons has to be chosen. Such parameters are called hyperparameters of the network. In general, the size and number of hidden layers regulate the capacity of information that an NN can extract from its input data. Especially the number of hidden layers allows a network to learn *deeper* relationships between them. Therefore a network with multiple hidden layers is also called a *Deep* Neural Network (DNN).

If the network architecture is chosen, the weights $\omega_{ji}^{(k)}$ and bias $\beta_j^{(k)}$ of the layers have to be adjusted to predict correct results. This is done during the training process, which is explained in the following section.

5.1.1. Training

The training of a Neural Network is the process that customizes the weights $\omega_{ji}^{(k)}$ and bias $\beta_j^{(k)}$ so that the model is able to predict target values $\vec{y}$ forgiven input vectors $\vec{x}$. This section gives an overview of the training based on [37, 38].



(a) Structure of a neuron. The input values x_i are multiplied with weights ω_i. These are summed up with an optional bias β and transformed by an activation function. Taken from [35]

(b) Structure of a Neural Network. Input values are fed in the neurons of the input layer. Each layer takes the output values of the previous layer as input. The last layer produces a final output value. Taken from [36]

Figure 5.1.

But before the training, it is important to split the data available into several categories: a training, test, and an optional validation dataset. The weight and bias adjustments are done with the training dataset. The evaluation of the model is done on the test dataset. It is important to separate these sets in order to ensure an uncorrelated and meaningful evaluation of the model. The validation dataset can be used to approximate the generalization capacity of the network or to tune hyperparameters.

To optimize the weights and bias of a network, a function is needed which quantifies the prediction quality dependent on the network parameters $\vec{\nu}$, i.e. weights and bias. Such functions are called cost or loss functions $C(\vec{\nu}, \vec{x}, \vec{y})$ and use the model output $\mathrm{NN}(\vec{x}, \vec{\nu}) = \hat{\vec{y}}(\vec{\nu})$ to measure the difference to the target output $\vec{y}$. An intuitive choice is the Mean Squared Error (MSE)

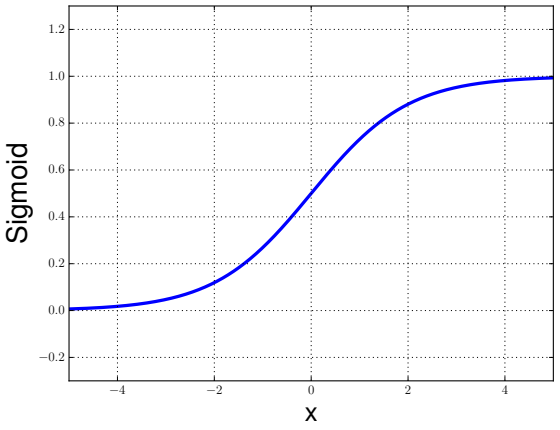
$$C(\vec{\nu}) = \frac{1}{2N} \sum_{n}^{N} ||\hat{\vec{y}}_n - \vec{y}_n||^2. \tag{5.3}$$

Especially in the case of the binary classification task, often the Binary Cross Entropy (BCE) is chosen

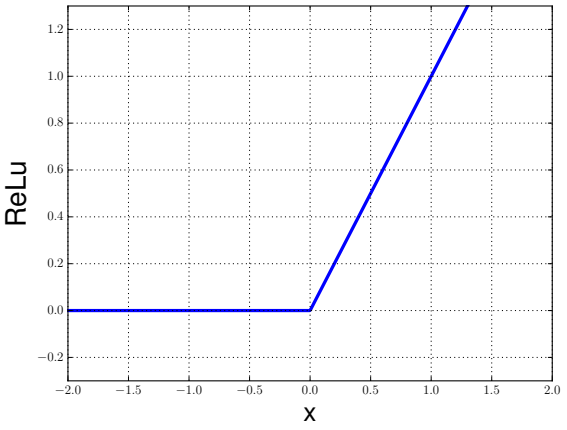
$$C(\vec{\nu}) = -\sum_{n}^{N} \left(\vec{y}_n - \ln\left(\hat{\vec{y}}_n\right) + (1 - \vec{y}_n) \cdot \ln\left(1 - \hat{\vec{y}}_n\right)) \right). \tag{5.4}$$

To optimize the network parameters $\vec{\nu}$, the loss has to be minimized, which is a high dimensional and often a non-convex problem. The basic concept of a gradient descend optimization process is shown in the following.

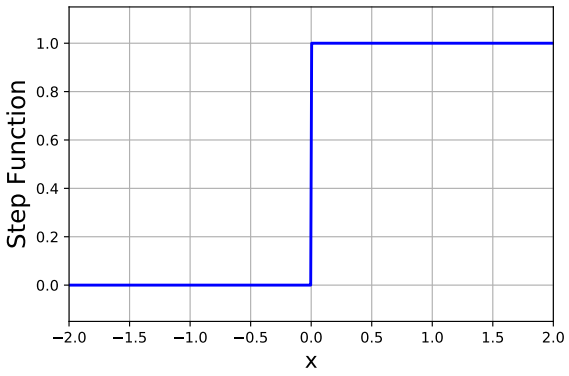
Assume a simplified NN with two hyperparameters $\vec{\nu} = (\nu_1, \nu_2)^\intercal$ and loss function C. To find the direction $\Delta\vec{\nu} = (\Delta\nu_1, \Delta\nu_2)^\intercal$ in which the hyperparameters have to be adjusted,



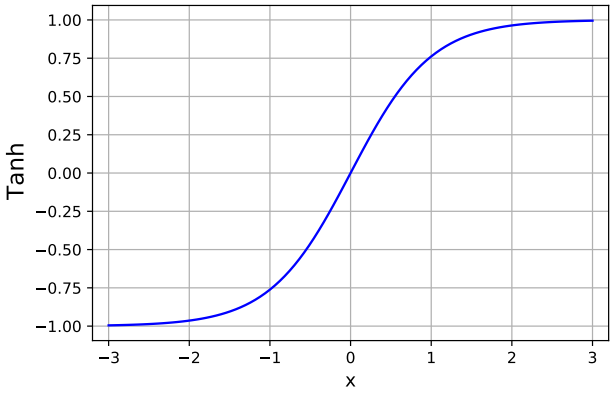
(a) sigmoid function $y = \frac{1}{1+e^{-x}}$



(b) ReLu function $y = \max(0, x)$



(c) step function $y = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$



(d) hyperbolic tangent $y = \tanh(x)$

Figure 5.2.: Illustrations of different activation functions: simoid, Rectified Linear Unit (ReLu), step function, and hyperbolic tangent (tanh).

one has to look at changes in C:

$$\Delta C(\nu_1, \nu_2) = \frac{\partial C}{\partial \nu_1}\Delta\nu_1 + \frac{\partial C}{\partial \nu_2}\Delta\nu_2 \tag{5.5}$$

$$= \nabla C \cdot \Delta\vec{\nu}. \tag{5.6}$$

with the derivative $\nabla C(\nu_1, \nu_2) = \left(\frac{\partial C}{\partial \nu_1}, \frac{\partial C}{\partial \nu_2}\right)^\intercal$. Knowing the relationship between ΔC and $\Delta\vec{\nu}$ the change in $\vec{\nu}$ can be chosen. Using the learning rate $\eta > 0$, a common choice is

$$\Delta\vec{\nu} = -\eta\nabla C. \tag{5.7}$$

Combining Equation (5.7) and Equation (5.6) shows that the loss is decreasing if $\nabla C \neq 0$ and $\eta \neq 0$

$$\Delta C = -\eta||\nabla C||^2 \leq 0. \tag{5.8}$$

The learning rate is a hyperparameter, which should be considered carefully. If η is too large, it is possible, that the minimum can not be determined accurately enough. Learning rates which are too small lead to exploding computing times.

In practice, it is not trivial to compute all gradients for a Neural Network and to find appropriate adjustments for all parameters $\vec{\nu}$. The state of the art algorithm to compute these gradients is the so-called backpropagation algorithm [39].

To optimize the parameters $\vec{\nu}$ step wise, the loss function could be evaluated on a single training datum. However, one can expect the data to have a certain degree of variability, which would lead to a very noisy minimization. To tackle this, data are often bundled into minibatches. Now one can use the average of the gradients for a minibatch. However, it must be noted that too big minibatches suppress noise, which can be a hindrance, as it can lead to the algorithm getting stuck in a local minimum.

When all the training data has been used once for the network training, a training epoch has been completed. Neural Networks are usually trained for multiple epochs until the training is stopped.

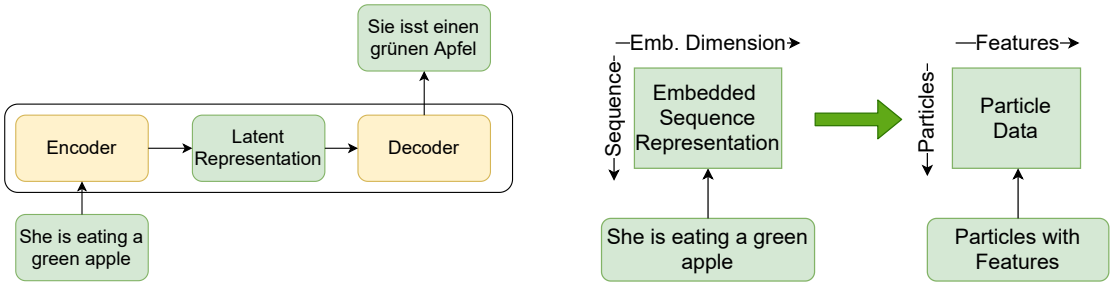
5.1.2. Overtraining and Early Stopping

Deep Neural Networks with multiple hidden layers have enormous capacities to extract complex information from the used training dataset. Therefore it is possible, that the network starts to learn properties during the training, which hold only for the training dataset and do not exist in the validation dataset or in general. This has the consequence, that the model is specialized on the training set and deteriorates its performance on the test and validation sets. Such a overspecification is called overtraining.

To detect overtraining, the model has to be tested periodically after each training epoch for example. A comparison between the performance of the model on the training set with the one on the validation set allows one to detect overtraining. As appropriate quantity, the loss function can be evaluated based on both sets. In general, the training loss is always expected to be slightly lower than the validation loss. Nevertheless, the validation loss should decrease if the training loss decreases. A typical sign of overtraining is, when the validation loss starts to increase, while the training loss continues to decrease. At this point, the model starts to learn structures in the training data, which are specific for the training set and do not hold for the validation set.

A simple method to avoid overtraining is early stopping. To do so, the loss is evaluated after each epoch on the training set and on the validation set as well. Early stopping counts the number of epochs for which the validation loss did not undercut its minimum. If these counts exceed a certain threshold, the training is stopped. This threshold is called early-stopping-epochs and is another hyperparameter for the training. This counter mechanism is necessary to allow the model to overcome local minima in the loss landscape. If the validation loss does not improve for multiple epochs, one can be sure, that this is not due to a local minimum, but to overtraining (or no learning progress at all). Common thresholds for early stoppings are epochs between 10 and 15.

Depending on the size of the datasets, it is possible that the evaluations of the loss dataset are very noisy, especially at the beginning of the training. To address this, the mean of the



(a) Structure of the Transformer Model. An input sequence is passed to the encoder part, which produces a latent representation of the sequence. This representation is used by the decoder part to translate it into another language.

(b) Input comparison for self-attention. NLP tasks translate each word into a quantitative embedding dimension (left). Therefore, the input has a sequence dimension and an embedding dimension. This is analog to a matrix with a dimension for particles and one for their features (right).

Figure 5.3.

last 15 epochs available can be computed, instead of considering the fluctuating validation loss. This allows to avoid a training stop due to noise and ensures an effective early stopping mechanism. This mechanism is used for all trainings presented in Chapter 6 to avoid incorrect early stopping.

5.2. Self-Attention

The concept of Self-Attention originates from the field of natural language processing (NLP). Self-attention is used here, because it allows translating a word by taking the remaining sequence into account, instead of translating it straightforward. To do so, context information is used.

One of the most significant papers in the field of attention is "Attention Is All You Need" [40]. It presents a sequence translation model, the so-called Transformer, which uses a new method to apply attention (see Section 5.2.1). Furthermore, the authors also show that their mechanism is superior in quality and more efficient in terms of its computing time compared to benchmark models. The model is built on an encoder-decoder structure, as shown in Figure 5.3a. With the use of self-attention, the encoder translates the input in a latent representation containing as much extracted information as possible. This is then used by the decoder to produce a final output.

For the Continuum Suppression, the encoder mechanism is applied on particle candidates to produce a latent representation of them. Due to the self-attention mechanism, this representation includes context information between particle candidates. Producing such a representation has two purposes: on the on hand, the self-attention mechanism is a promising technique to extract more information from the input and therefore, to increase the performance of the classifier. On the other hand, the context information in the latent representation is the foundation, which allows for permutation invariance under the particle order (see Section 5.3).

The following sections explain building bricks of the encoder layer of the Transformer model in a bottom-up structure. First, the basic mechanism of self-attention is explained. This is extended to the concept of multi-head self-attention and finally, its usage in the encoder layer is explained. The encoder layer is used for the Three Streamer model described.

5.2.1. Self-Attention Mechanism

By using the normal attention mechanism, attention is payed on specific parts of the input sequence to find a fitting output, e.g. the model's dictionary pays attention to the input sequence to find the best translation in its dictionary for the considered word in the input.

In contrast, by using self-attention, a model focuses on different parts of the input to find context information for each input element. Therefore, the input gives attention to itself, which explains the naming of self-attention. This mechanism computes so-called attention scores, by setting each input element into context with each other input. Such an example is shown in Figure 5.4.

The authors of [40] use the scaled dot-product to compute attention scores. Here it holds for the input $X \in \mathbb{R}^{\text{sequence} \times \text{embedding}}$, its shape will be referred to (sequence, embedding) in the following. The sequence dimension is the size of the input to which self-attention should be applied. For particle candidates, this corresponds to a list of candidates. The embedding dimension describes a quantitative representation of each word in the sequence in a higher-dimensional space. Further information about embedding can be taken from [41]. For the Continuum Suppression, this representation directly corresponds to the features of each particle candidate. Therefore, the input must be of shape (particles, features) to make use of self-attention in the Continuum Suppression. The adapted input shape is illustrated in Figure 5.3b.

For self-attention, X is multiplied by three different weight matrices W^Q, W^K, and W^V, these are implemented as three linear layers. The results are called query Q, key K, and value V. Using the scaled dot-product, the attention scores are computed by

$$Z = \text{softmax}(\frac{Q \times K^T}{\sqrt{d_k}})V \tag{5.9}$$

where d_k is the size of the feature dimension of the key. The product of $Q \times K^T$ produces a so-called attention map of shape (particles, particles), which includes the attention weights. These weights indicate the attention given from the second particle dimension to the first one. Note that in general, the weight matrix does not have to be symmetrical. In addition, padded particles are masked out from the attention map. The resulting matrix elements are scaled by $\sqrt{d_k}$ and transformed with the softmax function $f_i(\vec{x}) = \frac{e^{x_i}}{\sum_i e^{x_i}}$ to norm its values row wise. Finally, the resulting matrix is multiplied by V. The output Z is a matrix with attention scores, which are row-wise normed to one. Its scores point out regions of interest, and it is of the same shape as the input (particles, features). The scaled dot-product is illustrated in Figure 5.5a.

The concept of self-attention is a simple, but very effective way to learn about relationships between input elements, rather than learning about the input elements themselves. This mechanism is the basis for the multi-head attention mechanism, described in the next chapter.

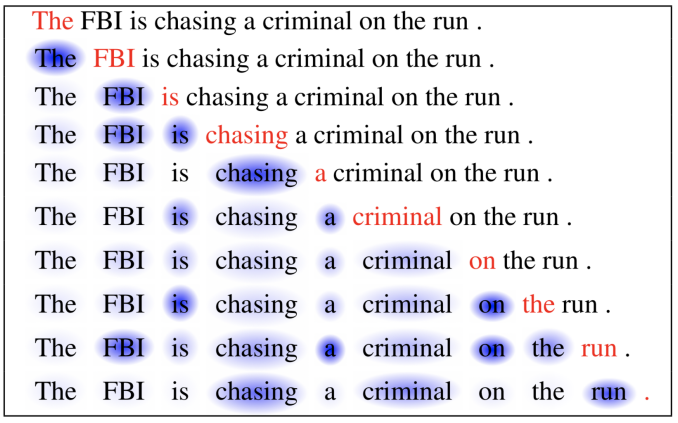


Figure 5.4.: In an NLP example, self-attention is used to point out attention weights for the sequence "The FBI is chasing a criminal on the run". The betrayed word to translate is marked in red. The attention weights given to other words in the sequence are marked in blue tones, whereas dark tones correspond to high weights and light tones to low weights. To translate the words, only the sequence before is taken into account. Taken from [42]

5.2.2. Multi-Head Self-Attention

The concept of multi-head attention is to apply the attention mechanism multiple times in parallel on the input X. This can be done for self-attention or attention in general, however this section only considers self-attention.

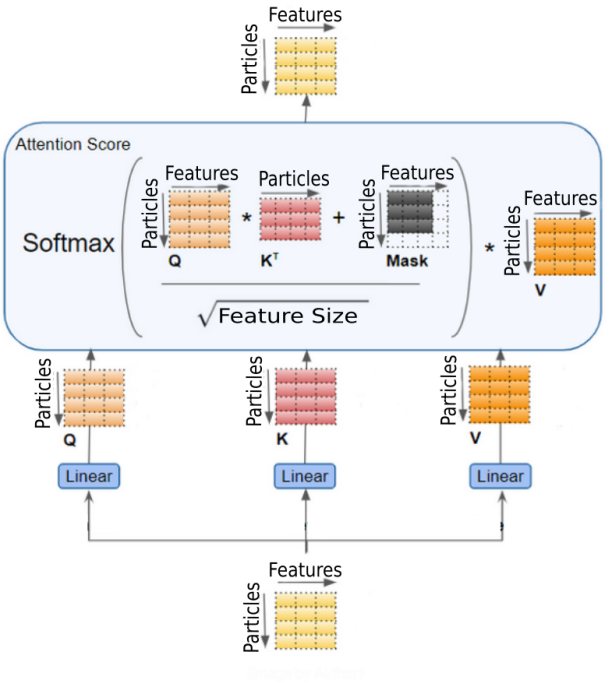
According to Section 5.2.1, an attention score can be computed for a given input X of dimension (particles, features). This is the procedure of one attention head. Multi-head attention uses N heads on the input X. In order to do so, the feature dimension of the input is sliced in N parts, resulting in N matrices of size (Pparticles, features/N). Each slice is linearly transformed by weight matrices W_i^Q, W_i^K, W_i^V, where i indicates the attention head, and used to calculate Z_i, as explained in Section 5.2.1. The results Z_i are concatenated together to the original shape of the input. After that, it is weighted with the trainable matrix W^0

$$\text{Multihead Attention}(Q, K, V) = \text{Concat}(Z_1, Z_2, ..., Z_N)W^0. \tag{5.10}$$

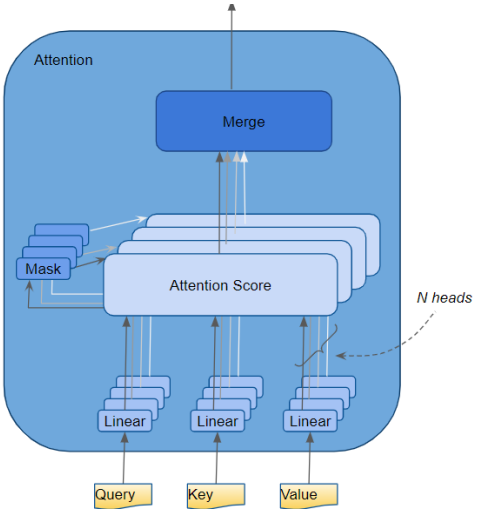
The concept of multi-head attention is also shown in Figure 5.5b. Its advantage is that each attention head has to attend to the information in a subspace of the feature dimension only. This simplifies the work of the individual heads and ensures in general that information of each feature dimension of X is present.

5.2.3. Transformer Encoder Layer

The Transformer is a sequence transduction model proposed by the authors of [40]. This model has an encoder-decoder structure, whereby the encoder is used in this work to give self-attention to particle candidates. An overview of the structure of the encoder layer is shown in Figure 5.6a.



(a) Self-attention mechanism. The input sequence is transformed linearly to produce query, key, and value. The attention score is computed by the scaled Dot-product. Additionally, a mask is used to exclude padded values. Adapted from [43]



(b) Illustration of multi-head attention. Query, key, and value are proceeded by a linear layer each. Their output is used to compute the attention score (scaled dot-product), where specific parts can be masked out. This mechanism can be applied by N heads in parallel. Their outputs are finally concat together. Taken from [44]

Figure 5.5.: Mechanisms for self-attention (a) and miltihead self-attention (b).

As input, the encoder layer takes a matrix of shape (particles, features). The main parts of the layer are the multi-head self-attention layer (see Section 5.2.2) and the feedforward layer. Both of them are applied with a residual skipping connection and a layer normalization which are explained in the next paragraphs.

Residual skipping connections [45] redirect the input of a layer and add it to its output (see Figure 5.6a). If a Neural Network is too deep, it can happen, that the first few layers can extract useful information from their input, but the last few layers cannot extract any more information. If backpropagation is applied then to adjust the weights, these last layers cause vanishing gradients. Thus, the backpropagation algorithm does not provide any weight adjustment and the training does not work anymore. A residual skipping connection circumvents this by allowing the next layer to access the input of the previous layer. This has the consequence, that if a layer produces vanishing gradients, backpropagation can skip these layers and compute useful gradients through the whole network. Therefore, residual skipping connections are especially important to facilitate the training of deeper network structures [45].

The encoder also contains a layer normalization [46] after the attention layer and another one after the feedforward layer. By applying this, the layer inputs are normalized along the layer dimension. This means, that a minibatch consisting of M events $(x_1, x_2, ..., x_M)$ each with K features $x_m = (x_{m,1}, x_{m,2}, ..., x_{m,K})$ is normalized by

$$x_{m,k} = \frac{x_{m,k} - \mu_m}{\sigma_m + \epsilon}. \tag{5.11}$$

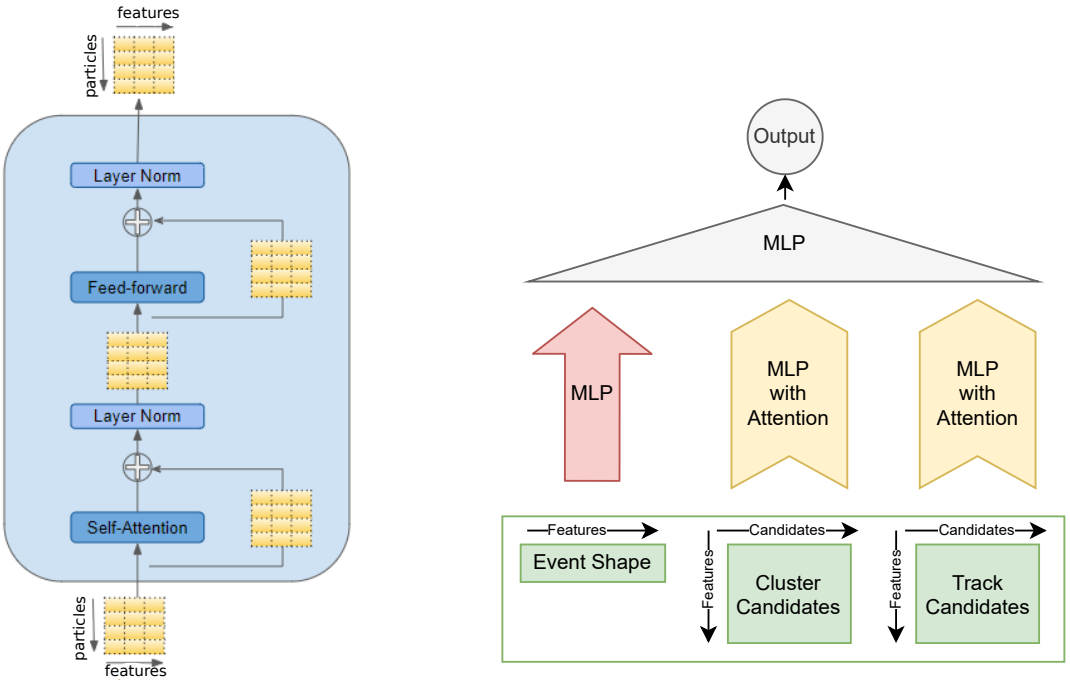
Here μ_m is the mean over all features k in event m, σ_m the corresponding standard deviation, and ϵ a small number which is added to avoid zero divisions. This normalization aims, like batch normalization [47], to reduce the shift in the variable distributions of the training data compared to the test data (covariate shift). Layer normalization also leads to an acceleration of the training process [46].

The output of the encoder layer is of the same size as its input, which allows stacking the layers easily. Since its input is of two dimensions, the encoder layer can be used as input for particle candidates, as explained in Section 5.3.

5.3. Three Streamer

The Three Streamer model is designed for a Deep Continuum Suppression application and is the basis for the Deep Ensemble (see Section 5.5), which is trained in the scope of this work. The Three Streamer is named after its three separate input data streams: event shape data, cluster candidates, and track candidates. The model is shown in Figure 5.6b.

The event shape data consist of 30 variables available which describe the whole event. Therefore, these are per event variables and the input for each event is of dimension (1). These variables are used as input for a standard MLP. ECL Clusters, i.e. photon candidates, are organized in a two-dimensional matrix of shape (Particle, Feature). In Figure 5.7a a distribution of gamma candidates per event is shown. The input shape for gamma candidates is set to 25. If there are fewer candidates in an event, the remaining entries



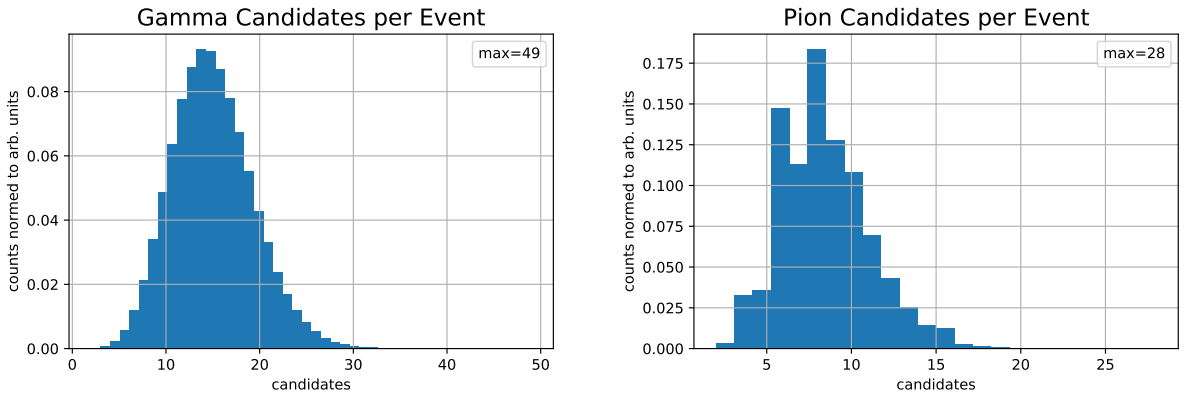
(a) Structure of an Encoder layer. The layer consists of a (multi-head) self-attention module and a feedforward layer, each is applied with residual skipping and followed by a layer norm. Adapted from [44]

(b) Structure of a Three Streamer model. The model has three input streams: an MLP for event shape data and two MLPs with an attention mechanism on particle candidates.

Figure 5.6.

are zero padded. Since there are ten features per gamma candidate, the input shape is finally (35, 10). This holds analogously for the pion candidates with final input shape (25, 15) and corresponding Figure 5.7b. Both the gamma and the pion candidates are used as input for an MLP with an attention approach. In detail, the Transformer Encoder layer (see Section 5.2.3) is implemented to use self-attention on the input data. Since the output of the encoder with shape (particles, features) should be passed to an MLP, one dimension has to be reduced. To do so an average pooling layer with a kernel size of the particle dimension is used to shape the output into (features).

One should note that the pooling layer provides the operation which gives permutation invariance under the particle order. In general, applying a pooling operation to achieve this invariance would not be effective since pooling reduces the information available. That is the reason why multi-head self-attention is used before pooling. It is important, that self-attention allows learning connections between the input elements and should therefore reduce the information loss of a pooling operation.



(a) Number of gamma candidates per event.

(b) Number of pion candiates per event.

Figure 5.7.: Number of gamma and pion candidates per event. The underlying dataset is described in Section 6.1.

Table 5.1.: Three Streamer architecture. The upper part lists the number of neurons per hidden layer, the number of hidden layers, and the number of output neurons for each MLP used in the three streams and the Head-MLP. The lower part shows the number of layers and neurons for the encoder layers (see Section 5.2.3). Additionally, the number of attention heads used for multi-head attention (see Section 5.2.2) within the layers is shown.

	Event	Gamma	Pion	Head MLP
MLP Hidden Neurons	200	500	500	500
MLP Hidden Layers	3	5	5	4
MLP Output Neurons	20	20	20	1
Enc. Layers		5	5	
Enc. Neurons		2000	2000	
Attn Heads		10	15	

Each of the three network streams produces an output of shape (20). A so-called Head MLP uses these to produce a final output prediction between one and zero, corresponding to whether an event is continuum or not. To do so a sigmoid activation function (see Figure 5.2a) is used for the last neuron of the Head-MLP, for all other Neurons of the Three Streamer the ReLu function (see Figure 5.2b) is used. Hyperparameters of the model, like the number of layers and neurons, are determined empirically and are shown in Table 5.1. Overall there are 4 520 686 trainable parameters.

5.4. Probability Calibration Methods

A probability calibration of a model aims to map its output values to probabilities. Since it is a common choice for classification tasks to restrict the output of a classifier to values between zero and one, probability calibration is supposed to map an output to its corresponding signal probability. To evaluate the degree of calibration, a so-called reliability plot is

evaluated. Such a plot shows the fraction of signal events plotted over the binned classifier output. Therefore, an optimal calibration would correspond to a linear function with slope one. Additionally, the Expected Calibration Error (ECE) [48] is a metric to quantify the merit of calibration. With a classifier output binned in M bins with content B_m, the bin accuracy $\mathrm{acc}(B_m)$, the bin confidence $\mathrm{conf}(B_m)$ and the ECE can be computed as:

$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i) \tag{5.12}$$

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} p_i \tag{5.13}$$

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|. \tag{5.14}$$

Where $\hat{y}$ describes the predicted class label, y the true class label, p_i the signal fraction of sample i, $|B_m|$ the number of entries in bin m, and n the total number of samples.

To optimize the calibration of a model, probability calibration methods can be used. These are applied after the training of a classifier and can be seen as a function which transforms the output of the classifier. In the scope of this work, Platt calibration [49] and isotonic calibration [50] are applied to compare the degree of calibration of a model with Platt and isotonic calibration, and a classifier without applied calibration method. The two mechanisms are explained in the following.

Platt Calibration

assumes predictions $\hat{y} \in [-1, 1]$ and transforms them into the range $[0, 1]$. Since it is common use to predict values $\hat{y} \in [0, 1]$, this transformation is not necessary in most cases. To calibrate the values $\hat{y}$ to probabilities $P(y = 1|\hat{y})$, the linear function $A \cdot \hat{y} + B$ is transformed by a sigmoid

$$\hat{y}_{\mathrm{cal}} = \frac{1}{1 + e^{A \cdot \hat{y} + B}}, \tag{5.15}$$

with fit parameters A and B. To determine these, the authors of [49] suggest to transform the target probabilities to $t_+ = \frac{N_+ + 1}{N_+ + 2}$ and $t_- = \frac{1}{N_- + 2}$ for positive labels N_+ and negative ones N_-. For $N_\pm \to \infty$, this corresponds to $t_+ = 1$ and $t_- = 0$. By using the $\mathrm{BCE}(\hat{y}_{\mathrm{cal}}, t)$, a maximum likelihood fit can be performed to adjust A and B.

Isotonic Calibration

sorts the model output according to its prediction values $\hat{y}$. Afterward, the data is iteratively combined in groups with the constraint, that these groups build a function with a monotonous ascending signal fraction. To do so, the Pool-Adjacent-Violators Algorithm (PAVA) is used. Its exact definition can be taken from [50]. The prediction values of the data is transformed to the mean prediction of the group they belong to. An isotonic fit for the Three Streamer presented in Section 6.5 is shown in Figure 6.2.

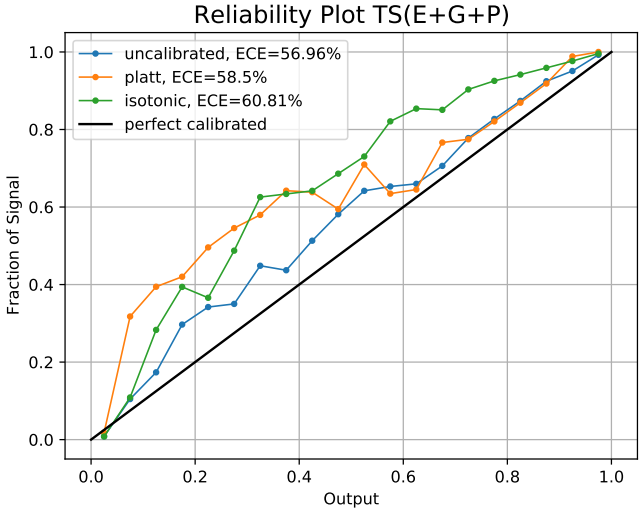


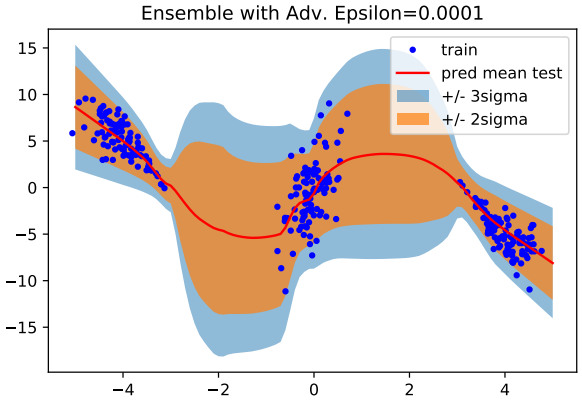
Figure 5.8.: Reliability Plot for a Three Streamer model. The output values of the model are binned to plot their signal fraction over the model output. The black line shows a perfect calibrated model. The blue data show the calibration of the model without a calibration method. The same model with isotonic and Platt calibration is shown as green and orange data respectively. In addition, the the corresponding ECE value for each model is given.

5.5. Deep Ensembles

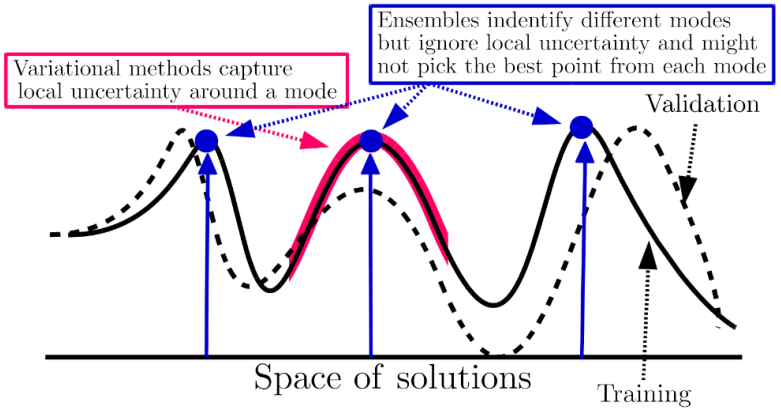
This work uses Deep Ensembles [4] to create predictive uncertainties for continuum classification tasks. This section explains the concept of Deep Ensembles and will give a comparison with Bayesian Neural Networks afterward.

A Deep Ensemble is a group of deep learning models. Such approaches are especially known from Decision Trees [51]. The individual models are either trained on subsamples of the dataset (e.g. bagging) or depend on previous models (e.g. boosting). In contrast, each model of a Deep Ensemble is trained individually with different initialization parameters and on the full dataset. This leads to an ensemble of uncorrelated models trained to achieve a high accuracy on their own. This is significant since the prediction of high-quality uncertainties demands that the models converge to different points in the solution space while ensuring a high accuracy at the same time. If this is given, the ensemble output is a composition of each model output for a given input.

For regression tasks it is suitable to use an ensemble of M models, predicting the mean μ_m and the standard deviation σ_m of the target distribution. The ensemble is then handled like a gaussian mixture $\sum_m^M \frac{\mathcal{N}(\mu_m,\sigma_m^2)}{M}$. Therefore, it holds for the ensemble output μ and



(a) Ensemble of NNs trained with the negative log-likelihood (NLL) and adversarial training. The training data (blue dots) is implemented with heteroscedastic uncertainties and varying densities of data points. The ensembles output mean is shown as a red line, the 2σ and 3σ uncertainty bands are shown in orange and blue.



(b) Sketch of Solution space. Possible solutions for the training data are shown as a solid line, the ones for validation data as a dotted line. Convergence modes are shown as blue dots, local uncertainties are marked as red areas. Taken from [52]

Figure 5.9.

its uncertainty σ^2

$$\mu = \sum_m^M \frac{\mu_m}{M} \tag{5.16}$$

$$\sigma^2 = \sum_m^M \frac{\sigma_m^2 + \mu_m^2}{M} - \mu. \tag{5.17}$$

To address classification tasks, an ensemble of M models can be trained, each with one discrimination output. As uncertainty, the upper and lower confidence level of an ensembles prediction can be determined by evaluating the corresponding quantiles of the prediction distribution.

Before applying a Deep Ensemble on physical data, their performance and the quality of their uncertainties were tested. This is shown in Figure 5.9a. The toy dataset counts 300 points (blue dots) and has heteroscedastic noise, which means, that the magnitude of the noise in y direction is dependent on x. For the data holds

$$y(x) = 7 \cdot \sin(x) + 3 \cdot \left|\cos\left(\frac{x}{2}\right)\right| \cdot \mathcal{N}(0, 1). \tag{5.18}$$

Additionally, there is a variety of the data density

$$x = \mathcal{N}(-4, 0.4) + \mathcal{N}(0, 0.3) + \mathcal{N}(4, 0.4), \tag{5.19}$$

especially the regions without data are needed to observe the reaction of the ensemble in regions without data.

The ensemble consists of 100 MLPs. Each of them has two hidden layers with 400 neurons to predict mean μ and variance σ^2 of the toy dataset. Each network is trained individually on the full dataset. To generate smooth uncertainty bands, small deviations $\epsilon = 0.0001$ were added to the input values $x + \epsilon$ (adversarial training, see [53]).

The result in Figure 5.9a shows the training data and an evaluation of the ensemble. The 2σ and 3σ uncertainty bands fit suitable in all regions with training data. As soon as the ensemble has to predict values in regions without training examples, wider uncertainty bands are produced. This behavior is expected in regions without training data and shows that Deep Ensembles are excellent candidates for uncertainty predictions.

A widely used method to produce predictive uncertainties is to use Bayesian Neural Networks. The next section explains the different principles of Bayesian Networks and Deep Ensembles.

5.5.1. Comparison with Bayesian Neural Networks

The current state of the art to produce predictive uncertainties is to use Bayesian Neural Networks (BNNs). Due to the popularity of BNNs, this section should give a short overview of them and compare the underlying principles with Deep Ensembles afterward. This comparison is based on [52].

The underlying concept of BNNs is to replace the constant network weights $\omega_{j,i}^k$ by weight distributions. For an evaluation step, each network weight is sampled from its distribution.

Uncertainties are produced by evaluating the network multiple times for each input. During the training of BNNs, the weight distributions have to be learned. This is done by Variational Inference, a simplification of a maximum a posteriori approach, where the weight distributions are adjusted to fit the training data. Detailed information can be taken from [54].

The main difference between Deep Ensembles and BNNs is the way they cover possible solutions in the solution space. This is shown in Figure 5.9b. A standard NN would cover one optimum in the space of solutions, such convergent points are called modes. Nonetheless, the training with variational methods leads to one mode in the solution space, too. Due to its weight distributions, a BNN does not only cover a mode but the local area around that (red area), which is interpreted as uncertainty. The approach of ensembles is different: since there usually are different local optima, each model is expected to converge to a different mode (blue dots). Instead of evaluating local uncertainties, the uncertainties of an ensemble rise from the global distribution of modes. Overall, this leads to a coverage of multiple, possible solutions which results in a high diversity of modes. Since a high diversity is only desirable if there is an appropriate trade-off between diversity and accuracy, this has to be investigated as well.

To evaluate the quality of Deep Ensembles regarding to high diversities, the authors of [52] present a prediction diversity vs. test accuracy plot. The diversity is measured as a fraction d_{diff} of different prediction values of the considered model with the baseline optimum (see below). Since the prediction diversity depends on the accuracy of the model, the diversity is normalized by $(1 - \text{accuracy})$.

The example shown in Figure 5.10 is created by using a ResNet20v1 [55] trained on the CIFAR-10 dataset [56] including 10 classes. To evaluate the performance of a single-mode model, one is trained as a baseline optimum (green star). To access the local area around the baseline optimum, checkpoints of the training trajectory are saved to sample new models from these checkpoints as subspace from the baseline model (colored dots). To do so, different sampling methods are used: random subspace (blue), dropout subspace (yellow), simple gaussian (violet), and rank 4 gaussian (pink) sampling (see [52] for detailed explanations).

Additionally to the baseline model and its subsampled models, multiple independent models were trained, converging to different modes in the solution space which corresponds to the principle of a Deep Ensemble. These models are marked as red stars. The result in Figure 5.10 shows, that models subsampled from the baseline model, i. e. solutions in the local area around the baseline model, seem to have a clear dependency in the diversity-accuracy plane. All independent optima achieve a better diversity-accuracy trade-off than the sub sampled models. This means that independently trained models can predict with a much higher diversity by achieving at least the same test accuracy than models in the local area of a single mode. Since it is crucial for suitable uncertainty predictions to cover a broad range of predictions by ensuring high accuracy, this experiment proves that Deep Ensembles are excellent candidates to predict uncertainties. This gives the motivation to test a Deep Continuum Suppression with predictive uncertainties with Deep Ensembles.

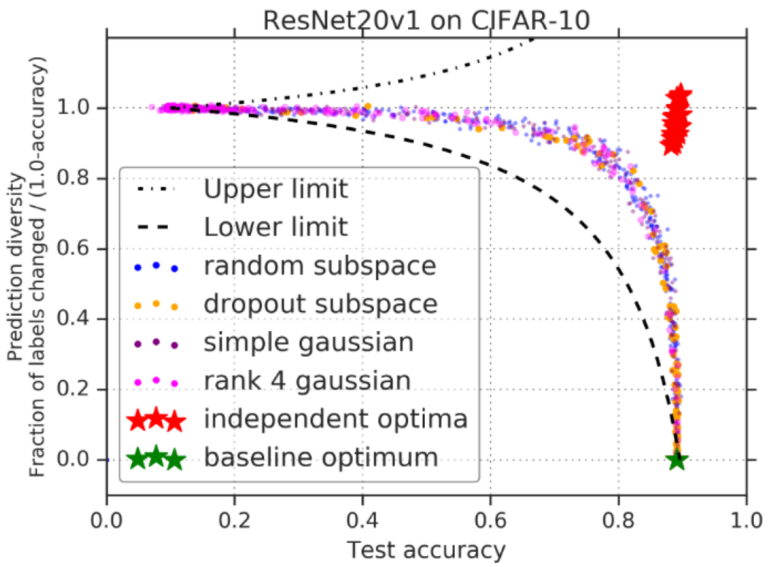


Figure 5.10.: Prediction diversity vs. test accuracy plot of ResNet20v1 on the CIFAR-10 dataset. Shown are different subsampling methods (colored dots) from the training trajectory of a baseline optimum (green star). Corresponding upper and lower limits are shown as dashed lines. Independent optima are marked as red stars. Taken from [52]

5.6. Distance Correlation

To decorrelate a classifier output from vertex variables like Δz (see Section 4.3), an appropriate metric is needed to measure this correlation. This work uses Distance Correlation (DisCo) [57] to do so this and to decorrelate the classifier from Δz. This chapter explains distance correlation and the way it can be used for decorrelation.

DisCo is a way to measure the correlation between two variables. In contrast to the well-known Pearson correlation [58], which only addresses linear correlation, the concept of DisCo allows to be sensitive on nonlinear correlations too. This ability is achieved by using all pairwise distances. Assume two-variable samples of size N (X_i, Y_i) with $i = 1, 2, .., N$. Then $a_{i,j}$ and $b_{i,j}$ are the distances within each sample

$$a_{i,j} = ||X_i - X_j|| \tag{5.20}$$

$$b_{i,j} = ||Y_i - Y_j||. \tag{5.21}$$

Furthermore $Aa_{i,j}$ and $B_{i,j}$ are defined as

$$A_{i,j} = a_{i,j} - \bar{a}_{.j} - \bar{a}_{i.} + \bar{a}_{..} \tag{5.22}$$

$$B_{i,j} = b_{i,j} - \bar{b}_{.j} - \bar{b}_{i.} + \bar{b}_{..} \tag{5.23}$$

with the mean of all i for the j-th index $\bar{a}_{.j}$, vice versa for $\bar{a}_{i.}$, and the mean for all i, j indices $\bar{a}_{..}$ (and analogue for $B_{i,j}$). Now the distance covariance can be defined as

$$\mathrm{dCov}^2(X, Y) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{i,j} B_{i,j}. \tag{5.24}$$

Distance correlation is defined as analog to the Pearson correlation. With the motivated distance covariance above, it holds

$$\mathrm{dCorr}^2(X,Y) = \frac{\mathrm{dCov}^2(X,Y)}{\mathrm{dCov}(X,X)\cdot\mathrm{dCov}(Y,Y)}. \tag{5.25}$$

Distance correlation behaves like Pearson correlation: values near one indicate a strong correlation, while values near zero indicate no correlation. Further information can be taken from [57].

The substantial advantage of addressing nonlinear correlations too makes the distance correlation an attractive method for decorrelation tasks. The authors of [5] use this advantage and present a method, based on DisCo, to decorrelate a classifier output from a certain variable. In order to do so, the $\mathrm{dCov}^2(X,Y)$ term is added as a regularisation term to the loss function

$$\mathcal{L}' = \mathcal{L} + \lambda \cdot \mathrm{dCorr}^2(\Delta z, \hat{y}) \tag{5.26}$$

where λ controls the strength of the decorrelation term, Δz is the variable to decorrelate from the classifier output $\hat{y}$. By appending the DisCo term during training, the minimization algorithm will automatically minimize the decorrelation between the two variables, if the parameter λ is well chosen. It is important to note, that λ is the only hyperparameter introduced by decorrelation with DisCo. Therefore one can conclude, that the decorrelation with DisCo enables an effective decorrelation mechanism with easy implementation.

6. Applied Continuum Suppression

After introducing a promising mechanism for the Continuum Suppression in the previous chapter, this chapter shows an applied suppression for these methods.

First, the underlying dataset for the Continuum Suppression is explained. Next, the preprocessing of this dataset is pointed out. To compare the results of Continuum Suppression models, Receiver Operating Characteristic curves are explained. Next, an MLP from [3] is reproduced to have a benchmark model. A preliminary model for a Deep Ensemble, a single Three Streamer Model is trained and evaluated. In addition, a probability calibration test is done. After that, an ensemble of Three Streamer models is trained to add uncertainties to a Continuum Suppression. In the end, a Deep Ensemble using DisCo to decorrelate a classifier output from Δz is trained and evaluated.

6.1. Dataset

In 2017, the work of [3] introduces a Deep Continuum Suppression for the Belle II experiment. In order to allow for a good comparison between this work and the results of [3], the dataset used in this work is chosen to be as similar as possible to the one of its predecessor. The dataset in [3] is built on the Monte Carlo Campaign 7 phase 3. However, since the quality of the Monte Carlo generation is continuously improved to better model the detector data, this work uses data from the Monte Carlo Campaign 13a (MC13a) which may differ from campaign 7 (MC7).

The dataset consists of a signal and a background set, which were separately generated in the scope of MC13a. As explained in Section 4.1, the background sample consists of $u\bar{u}, d\bar{d}, s\bar{s}, c\bar{c}$, and $\tau^+\tau^-$ events. Analogous to [3], the process $B^0 \to K^0_S(\to \pi^+\pi^-)\pi^0(\to \gamma\gamma)$ (incl. c.c.) is chosen as signal. Therefore the signal sample only contains $B\bar{B}$ events with at least one meson decaying to $K^0_S\pi^0$. Using BASF2, such events are reconstructed from the background and the signal datasets, to obtain signal candidates.

For the reconstruction, only tight track and cluster candidates are considered. This implies that in the corresponding detector regions only high energy cuts were made, leading to higher purity of correctly reconstructed candidates. Furthermore, cuts during the reconstruction require for $\pi^\pm$ candidates a χ^2-Probability over 0.001 for the track fit. Additionally all reconstructed π masses are restricted to $0.115 \leq M_\pi \leq 0.152\,\text{GeV}$. Analogously, the mass of the reconstructed K^0_S is expected to lie between 0.48 and 0.516 GeV. Finally the

reconstructed B^0 is allowed to have an energy difference to the theoretical B^0 energy of $-0.3 < \Delta E < 0.3\,\text{GeV}$. Its beam-energy constrained mass $M_{bc} = \sqrt{\frac{E_{beam}^2}{c^4} - \frac{p_B^2}{c^2}}$ is restricted to $5.2 < M_{bc} < 0.53\,\text{GeV}$. An overview of all applied cuts is given in Table 6.1.

Besides the mentioned cuts, only clusters with at least one CDC hit and neutral particles with a momentum of not less than 0.05 GeV are considered. Both, clusters and tracks, are allowed to have a maximal momentum in the center-of-mass frame of 3.2 GeV.

Table 6.1.: Cuts applied during the reconstruction.

Particle	Cut
B^0	$5.2 < M_{bc} < 0.53\,\text{GeV}$
	$-0.3 < \Delta E < 0.3\,\text{GeV}$
K_S^0	$0.48 \leq M_{K_S^0} \leq 0.516\,\text{GeV}$
π^0	$0.115 \leq M_\pi \leq 0.152\,\text{GeV}$
$\pi^\pm$	$\chi^2_{\text{ProbFit}} > 0.001$
	$0.115 \leq M_\pi \leq 0.152\,\text{GeV}$

After reconstruction, the dataset retains 384 716 events with a signal-to-background ratio of 0.636. This is less than a tenth compared to the data used in [3], where 4 914 670 events with a signal-to-background ratio of 1.32 were available. Furthermore, the dataset used in this work is highly imbalanced. To counter this, weights are used to sample minibatches with an equal amount of signal and background during the training. Each plotted result and each metric presented in this work, is produced with weighted events if not mentioned otherwise.

After reconstruction, the dataset is split into three parts: a training set with 81 % of the data available, a validation set with 9 %, and a test set with 10 %. Each dataset has the same signal-to-background fraction. The training set is used for the training of the classifier. During this, the model is evaluated on the validation set. Since the model has not seen the validation data, the evaluation shows a training-independent model performance and can be used for early stopping. After training, the test set is used to determine the performance of the model on another separate set.

6.2. Preprocessing

During preprocessing, the dataset is prepared for better training of the model. This includes data cleaning, like deleting events produced with Not-a-Number (Nan) values which could originate from failed vertex fits for example. Moreover, outliers in the dataset could lead to high neuron outputs in the model and therefore lead to more difficult training. To address this, the variable distributions of all datasets are capped to the $5 \cdot 10^{-5}$-th quantile of the training data. Additionally, selected variables are Yeo-Johnson [59] transformed. This aims to change the variable distribution to a gaussian shape since it is easier for the model to

learn from them. The Yeo-Johnson transformation performs a power transformation for each distribution in the data. To do so, the function $\psi(\lambda, x)$ is defined as

$$\psi(\lambda, x) = \begin{cases} \left((x+1)^{\lambda} - 1\right)/\lambda & \text{if } \lambda \neq 0, x \geq 0 \\ \log(x+1) & \text{if } \lambda = 0, x \geq 0 \\ -\left[(-x+1)^{2-\lambda} - 1\right)\right]/(2-\lambda) & \text{if } \lambda \neq 2, x < 0 \\ -\log(-x+1) & \text{if } \lambda = 2, x < 0 \end{cases} \tag{6.1}$$

with parameter λ and the variable to transform x. With a maximum likelihood fit on the training set, the best matching λ is determined. Next, the variable transformation is applied in each dataset. Plots of all transformed and not transformed variables are shown in Appendix A.

6.3. Figure of Merit

To compare different models with each other a figure of merit has to be chosen. In this case, the Receiver Operating Characteristic curve (ROC) is a suitable choice. A ROC curve shows the signal efficiency of a classifier over its the background rejection. To compute these, the variables True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN) are needed. Their concept is explained in Figure 6.1. Now the True Positive Rate (TPR) and the False Positive Rate (FPR) can be computed

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.2}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{6.3}$$

The signal efficiency directly corresponds to the TPR, whereas the background rejection is defined as $1 - \text{FPR}$. As an example for different ROC curves Figure 6.3 can be considered. Additionally, the Area Under Curve (AUC) is specified to quantify the quality of a ROC curve.

6.4. Reproduction of Previous Work

To have a fair comparison of the models from this work with the previous work from [3], the MLP (see Section 4.3) from [3] is reproduced as accurately as possible and evaluated on the actual dataset from MC13a. To do so, the clusters and tracks in each event were ranked according to their momenta in the center-of-mass frame. For clusters, the first ten candidates of the B^0 and the first ten of the ROE are kept. For tracks of B^0 daughters, the first five positively charged, and the first five negatively charged track candidates are kept. The same holds for tracks of the ROEs. Figure 4.3 illustrates the particles selection. If there are not enough candidates to fill out all slots, the remaining slots are zero-padded.

For the preprocessing, all variable distributions are binned in 100 bins with an equal amount of data (equal frequency binning). In [3] is mentioned, that input variables can be

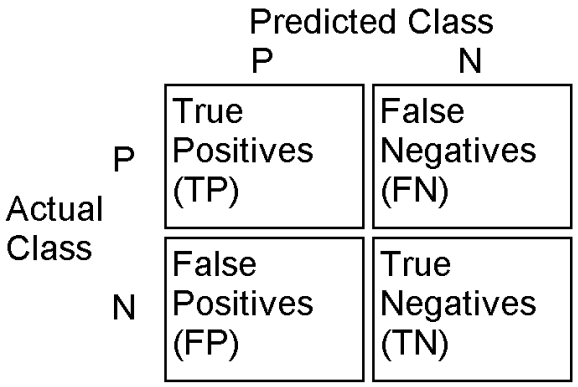


Figure 6.1.: Confusion matrix for binary classification tasks with Positive and Negative classes (i.e. signal and continuum). True Positives (TP) and True Negatives (TN) are correct predicted Positives and Negatives respectively. On the other hand, False Positives (FP) and False Negatives (FN) are incorrectly classified Positives and Negatives. Taken from [60]

normalized during the preprocessing. Since there is no information about the normalization itself and which features to normalize, it was decided not to normalize. However, since this work does not use a normalization either, a possible effect would show up on both models.

The architecture of the MLP has 5 hidden layers with 50 neurons each. As activation function, the hyperbolic tangent (see Figure 5.2d) is used, the activation function of the last neuron is a sigmoid function (see Figure 5.2a). The training is performed with minibatches of size 500, with class weights used to ensure balanced training. To minimize the loss, the ADAM optimizer [61] is used with a learning rate of 0.001. Finally, the training is stopped if there was no validation loss improvement in the last ten epochs (early stopping). The ROC curve of the MLP is shown in Figure 6.4.

6.5. Three Streamer Model

As a preliminary stage for a Deep Ensemble, this section addresses the Three Streamer model which is explained in Section 5.3. The training of a Three Streamer, its performance for a Continuum Suppression, and its probability calibration are shown in this section.

6.5.1. Training

For the training of the Three Streamer model, the dataset introduced in Section 6.1 is used and preprocessed as described in Section 6.2. To calculate the loss, the Binary Cross Entropy (see Equation (5.4)) is chosen. The Three Streamer model has, as compared to other machine learning models, many hyperparameters to optimize. Since one of the benefits of ADAM is the handling of high-dimensional problems, this minimization algorithm is a suitable choice to optimize the loss of the Three Streamer model [61]. A learning rate of 10^{-4} is chosen. During the training, minibatches of size 500 are drawn from the training sample, which consists of 311 653 events. Since the available dataset is imbalanced, class weights are used to pick minibatches with equal amounts of signal and background from

the training batch. During the training, the model is evaluated after each epoch on the validation dataset. If the validation loss does not decrease for 12 epochs, the training is stopped.

6.5.2. Separation Power

After the training, the discrimination power of the Three Streamer model is evaluated on the test dataset. To do so, a ROC curve is plotted, and, in addition, the corresponding AUC score is computed (see Section 6.3). Furthermore, an MLP reproduced from [3] (see Section 6.4) is evaluated to compare the outcome of both model architectures.

The resulting ROC curves and their ROC AUC values are shown in Figure 6.3. The MLP achieves a ROC AUC score of 0.9848. The Three Streamer model reaches an AUC score of 0.9911 and therefore exceeds the MLP. This shows that the Three Streamer model is more powerful than the MLP. The better performance could be due to the implemented self-attention mechanism that probably extracts more information from the underlying data, or to the bigger network structure of the Three Streamer in general. Overall, these results demonstrate that the Three Streamer is a suitable candidate for the Continuum Suppression.

6.5.3. Probability Calibration

It is important for machine learning models, that their output is interpretable. For Deep Ensembles (see Section 5.5) in particular, it is important that the ensemble members are probability calibrated. Non-calibrated members would have distortions in their output values and make them incomparable with each other. In contrast, calibrated ensemble members lead to comparable and interpretable outputs. These allow for well-calibrated uncertainties of a Deep Ensemble. To ensure these, the model output of a Three Streamer is compared with the output of the same model, with additional probability calibration.

The probability calibrations are performed after the training, based on the validation dataset. This ensures an independent model evaluation on the test dataset. The calibration methods used to do so are Platt scaling and isotonic regression (see Section 5.4). The isotonic regression for the Three Streamer is shown in Figure 6.2. To be able to judge the quality of the calibration, a reliability plot is made and shown in Figure 5.8. Additionally, the ECE (see Section 5.4) quantifies the level of calibration.

The results in Figure 5.8 show that the isotonic calibration achieves an ECE of 60.81 and the Platt calibration a score of 58.5. The model without any calibration method reaches ECE=56.96. The reliability plot shows that the model without calibration and the one with isotonic calibration deviate in a concave shape from the ideal calibration. The Platt calibration has clear deviations from a perfectly calibrated model in the region between 0 and ≈ 0.6. For outputs above 0.6, the calibration performance is comparable to the model without the calibration method.

The calibration results in Figure 5.8 show clearly that the model without any applied calibration method outperforms the one with Platt or isotonic calibrations. Other Three Streamer models trained in the scope for this thesis show similar results. This highly suggests, that trained Three Streamer models are by default probability calibrated, and

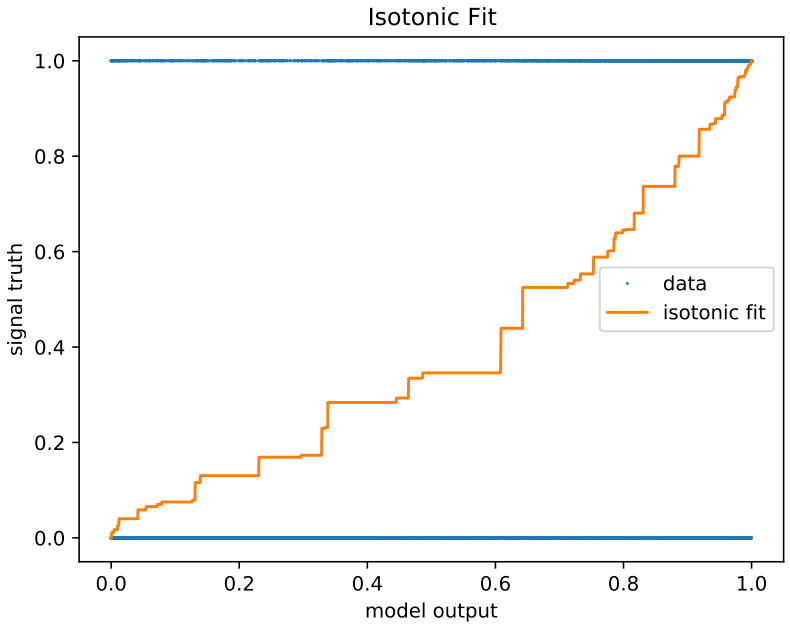


Figure 6.2.: Isotonic fit for the Three Streamer model. The data is sorted according to their predictions and plotted as signal or continuum (blue dots). Through application of PAVA, the data is grouped with monotonously ascending signal fractions.

none of the tested methods increase the degree of calibration. Therefore, Three Streamer models are suitable candidates for uncertainty predictions with an ensemble approach. The classification grades are nearly equal to each other, as the ROC AUC values in Figure 6.3 show.

Since the Three Streamer uses a sigmoid activation function in the output layer, it is reasonable to assume that such models are implicitly Platt calibrated. Moreover, it is possible that such models can perform a more effective Platt calibration. In general, these models have more weights in the output layer and in the previous ones to fine-tune the calibration. This could lead to better results than isotonic calibration or Platt scaling using only two parameters. Nevertheless, to find a clear reason why Platt and isotonic calibration do not improve the model calibration, further studies are needed. However, the aim of the calibration tests was to ensure a high degree of probability calibration and to compare methods for improvement. This analysis showed, that the default calibration of the Three Streamer is superior, no calibration method tested is able to increase the degree of probability calibration. Therefore, the Three Streamer can be directly used in a Deep Ensemble approach.

6.6. Ensemble of Three Streamers

In order to add predictive uncertainties to the Continuum Suppression, a Deep Ensemble is created. A motivation, an explanation on how these work and a comparison with variational

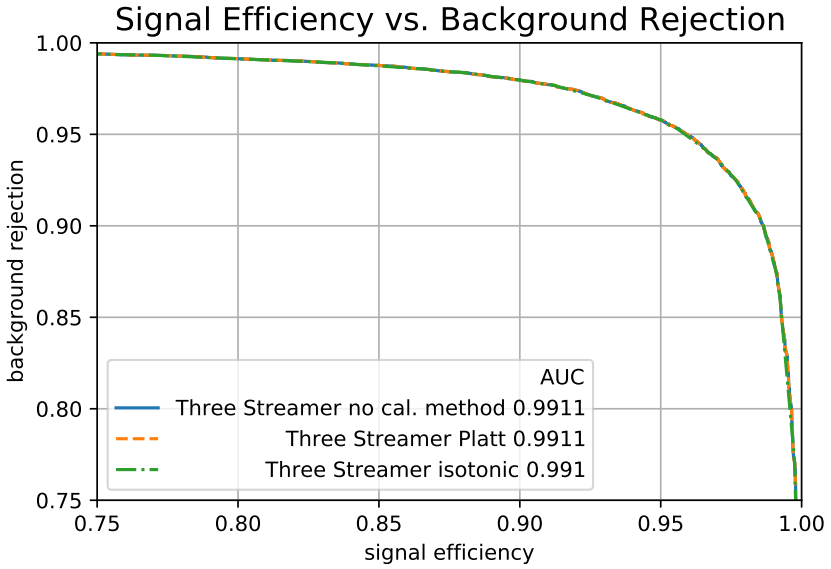


Figure 6.3.: Background rejection over signal efficiency for different Three Streamer models. The Three Streamer without any applied calibration method is plotted in blue. The same model with isotonic and Platt calibration is plotted in a green an orange respectively. In addition the ROC AUC values are given. The models perform identically, except of a slightly smaller AUC value of the isotonic one.

methods are given in Section 5.5. The ensemble is composed of 80 Three Streamer models, which are explained in Section 5.3.

6.6.1. Separation Power

To evaluate the predictive performance of the Deep Ensemble, a ROC curve, and corresponding AUC score are determined. The result is compared to a single Three Streamer model and a reproduced MLP of [3] (see Section 6.4). These results are shown in Figure 6.4.

The separation power of the Deep Ensemble lies with a ROC AUC score of 0.9912 slightly above the one of the Three Streamer with 0.9911. The MLP achieves a ROC AUC score of 0.9848 and does therefore, not perform as well as the ensemble. Furthermore, the results show that the separation power of the ensemble increases slightly through an ensemble effect.

6.6.2. Uncertainties

Using a Deep Ensemble allows for a Deep Continuum Suppression with predictive uncertainties. In order to provide more detailed insight into the produced uncertainties, intermediate steps are shown and discussed. After that, final results for predictive uncertainties are presented.

To predict uncertainties for an event, each member of the ensemble is evaluated to determine the 95 % confidence levels (CLs) of the resulting distribution, as explained in Section 5.5.

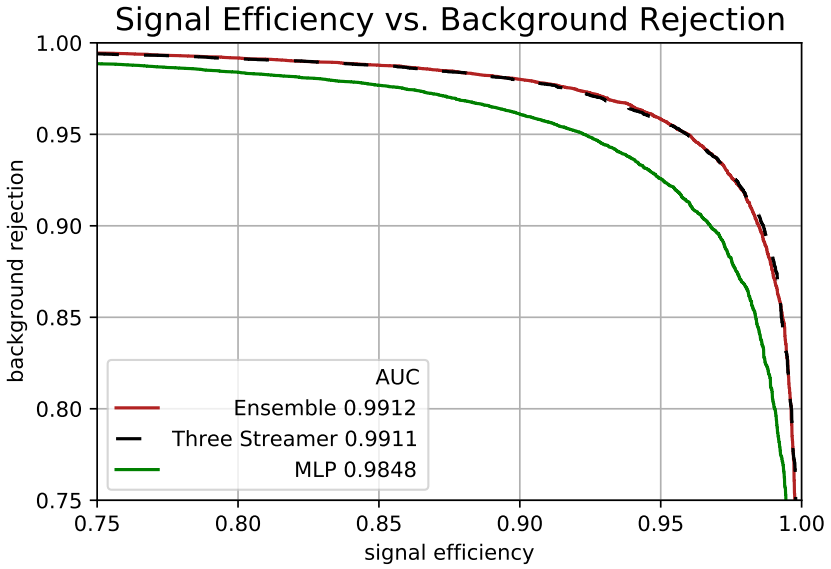
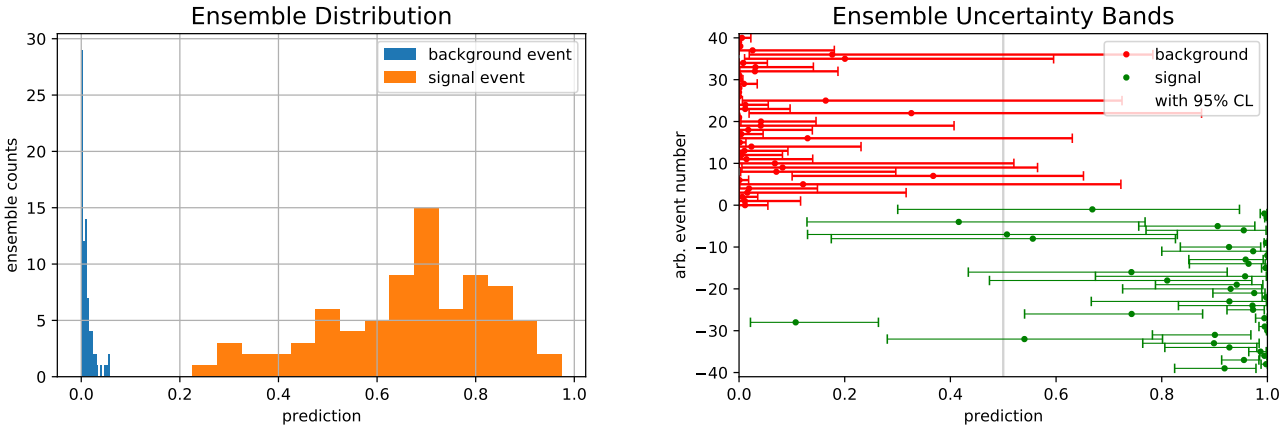


Figure 6.4.: Background rejection over signal efficiency for different models. The performance of an MLP (see Section 6.4) is plotted in green. The models designed in the scope of this work are the Three Streamer model (black) and the Ensemble of Three Streamers (dark red).

Such an ensemble distribution is shown for one signal and background event in Figure 6.5a. Regarding the background event, the predictions of the ensemble members are close together, which implies, that the modes in the solution space provide similar solutions. As a result, the ensemble is confident about its prediction. The shape of the distribution is similar to an exponential one. On the other side, the distribution of predictions for the signal event is more widely dispersed. In particular, there are ensemble members above and below the class limit of 0.5. This shows that the diversity of modes found by the ensemble, is much higher compared to the background event. This diversity leads to wider uncertainty limits. Therefore, the ensemble is not confident about its prediction.

In Figure 6.5b predictions and corresponding uncertainty limits are shown for 40 random signal and background events each. The plot shows that predictions near zero or one tend to have tighter uncertainties. The examples also suggest that predictions outside these regions have wider uncertainty bands. Moreover, the distributions in Figure 6.5a and the data plotted in Figure 6.5b prove, that the Deep Ensemble is able to predict asymmetric uncertainties. This is especially beneficial for predictions near zero (or one) as in this case the model can give wider uncertainties towards one (zero), depending on the variety of modes the ensemble can reach.

To investigate the ability of the Deep Ensemble to predict proper uncertainties for the Continuum Suppression, uncertainty vs. prediction plots are made. These scatter plots show the upper and lower uncertainty limits for all events, dependent on their predicted value. In Figure 6.6 an uncertainty vs. prediction plot with only true signal data is shown (Figure 6.6a), Figure 6.6b shows analogously shows a plot with background data, and



(a) Ensemble evaluation on one signal event and one background event.

(b) Ensemble evaluation on 40 signal events and 40 background events.

Figure 6.5.: Plots for Deep Ensemble uncertainties.

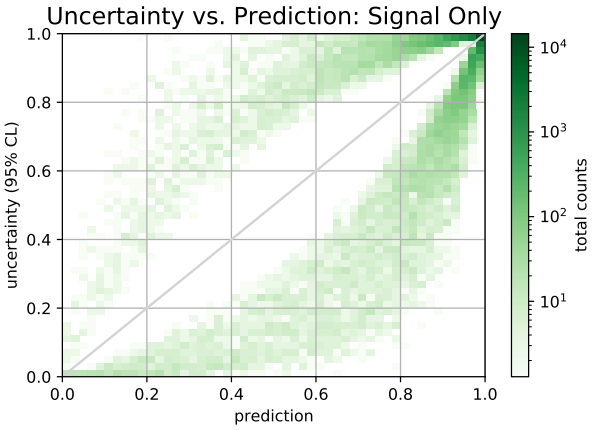
Figure 6.6c finally combines both, signal and background data. All plots are made with class weights.

Figure 6.6a shows, that the major part of the data is located in the region above 0.5 because most signal events are classified correctly. For predictions below 0.5, especially the distribution of the upper uncertainty limits can only be recognized weakly due to low statistics. However, an asymmetric distribution for the upper and lower uncertainty limits can be expected. In regions near zero and one, these limits are close together, whereas the spread widens towards predictions of 0.5. Moreover, one can recognize a minimum gap between the upper and lower limits. This one is small for uncertainties near zero and one and is very pronounced in between.

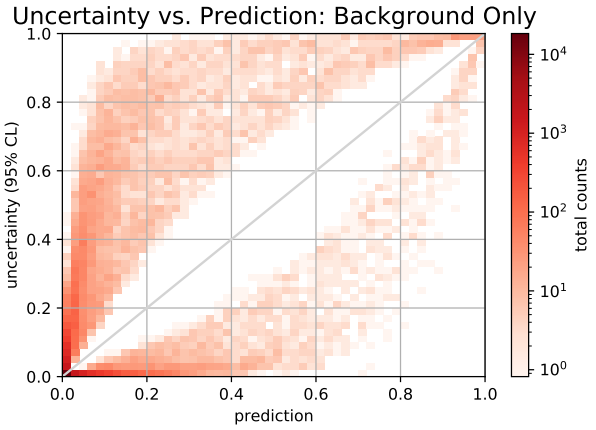
In general, all observations made for Figure 6.6a can be found in Figure 6.6b. Additionally, the upper uncertainties for the background only plot cover a wider range, compared to the lower limits of the signal plot. Figure 6.6 is a superposition of the described signal and background plots.

The general shape of the uncertainty distributions is as expected. This means, that the ensemble is very confident near predictions close to zero and one. Towards predictions of 0.5, both, the upper and lower, uncertainty limits get wider, implying, that the ensemble is uncertain about its predictions.

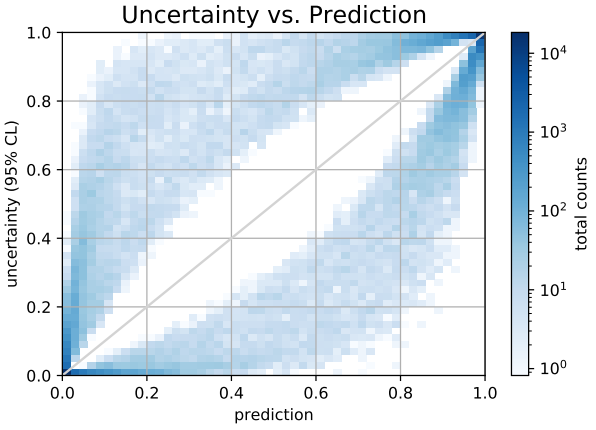
Especially mentionable is the observed minimum gap between the limits. It indicates that tight uncertainty bands can only be found near zero and one. A broad gap outside these regions shows, that the ensemble has a minimum uncertainty standard here. If this would not be the case, the ensemble could be able to make a signal prediction of 0.5 for example, with very tight uncertainties. But a prediction of 0.5 signal probability indicates that the ensemble cannot assign the event to a class. Tight uncertainties show, that the ensemble is confident about its prediction of 0.5 signal probability. This would mean, that all ensemble members give predictions near 0.5 and therefore, that they either do not have the capacity



(a) Signal only



(b) Background only



(c) Signal and background

Figure 6.6.: Uncertainty vs. prediction plots for a Deep Ensemble. The x-axis shows the predicted value (i.e. the mean) of the ensemble. On the y-axis the absolute uncertainty limits of the ensemble shown. For this, the upper (above grey line) and lower (under grey line) uncertainty limits for given predictions are shown as scatter plots. Figure 6.6c shows the whole test dataset, Figure 6.6a only true signal events, and Figure 6.6b only true background events. Since the dataset is imbalanced, all plots are produced with class weights.

to classify the event or that the event is so exotic, that it does not fit in any structures learned by the ensemble members. However, since no exotic events are used in this work, and a Three Streamer with enough capacity is showed in Section 6.5, such predictions would be contradictory and a minimum gap between the uncertainty limits highlights the reliability of the ensemble.

Another noticeable point are the uncertainty ranges for background events. Here the upper uncertainty limits have notably large ranges. This effect does not occur in the lower limits. Such an effect shows, that there are predictions of ensemble members that differ widely from the predicted ensemble mean for events classified as background. In this example, the variety of different modes reached by the ensemble can be observed particularly well. Nevertheless, these wide uncertainty ranges can be traced back to the fact that continuum events mimic the chosen signal process. Therefore, some ensemble members cannot distinguish these events, which leads to higher uncertainties.

In conclusion, the results in Figure 6.6 show: The ensemble is confident about its predictions near zero and one and insecure in regions in between; a minimum gap between the uncertainty limits underline the credibility of the uncertainty ranges; pronounced upper uncertainty limits for background events show their similarity to signal events. Therefore, it is shown, that Deep Ensembles are excellent candidates to predict interpretable and reliable uncertainties for Continuum Suppression.

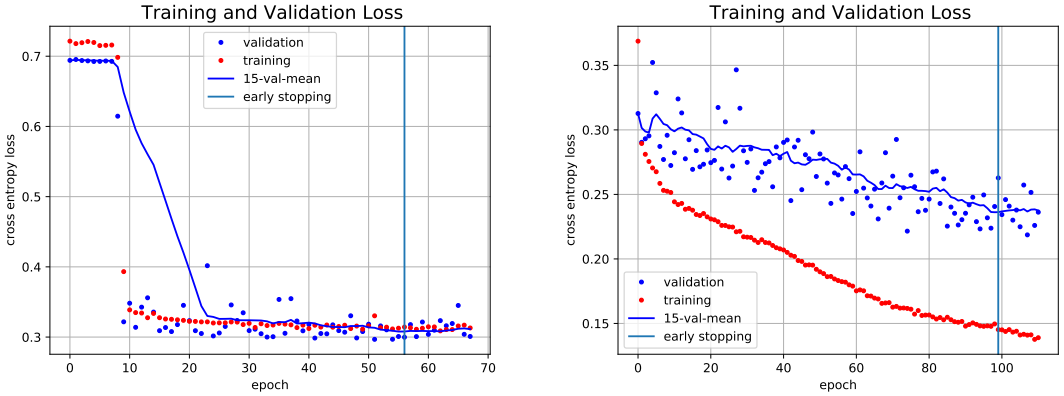
6.7. Ensemble Decorrelation with Distance Correlation

It is well known, that the use of vertex information in a Continuum Suppression results in a correlation between the classifier output with vertex features, especially with Δz. To address this, a decorrelation mechanism using DisCo is explained in Section 5.6. In the following section this method is applied to decorrelate a Deep Ensemble from Δz.

6.7.1. Training

To ensure comparable results, the ensemble hyperparameters are chosen equivalent to the one in the previous sections. Therefore, the ensemble consists of 80 Three Streamers with architectures as explained in Section 5.3. The training is performed as described in Section 6.5.1. In addition, the DisCo term is added to the cost function.

During the training of the individual models, it turned out that the DisCo term has an unexpectedly strong influence on the training. The authors of [5] use values up to 600 for λ. Figure 6.7a shows the loss of a Three Streamer model with $\lambda = 150$. The figure shows, that the model does not learn anything at all in the first few epochs. Then, the loss abruptly falls and then continues to decrease only minimally. Such a scenario can also be observed for other values of λ. In addition, λ should not be chosen too small, as otherwise the decorrelation effect fails. To address this, the models are trained for ten epochs with $\lambda = 0$. From the tenth epoch, λ is step-wise increased by three in each epoch. This allows the model to first converge to a minimum and then slowly reduce the correlation. This approach is motivated by [62]. A result of this training procedure is shown in Figure 6.7b. It can be seen, that the training is more stable although λ reaches values up to 300.



(a) λ is set to 150 from the beginning.

(b) λ is set to zero. From the tenth epoch, λ is increased by three in each epoch.

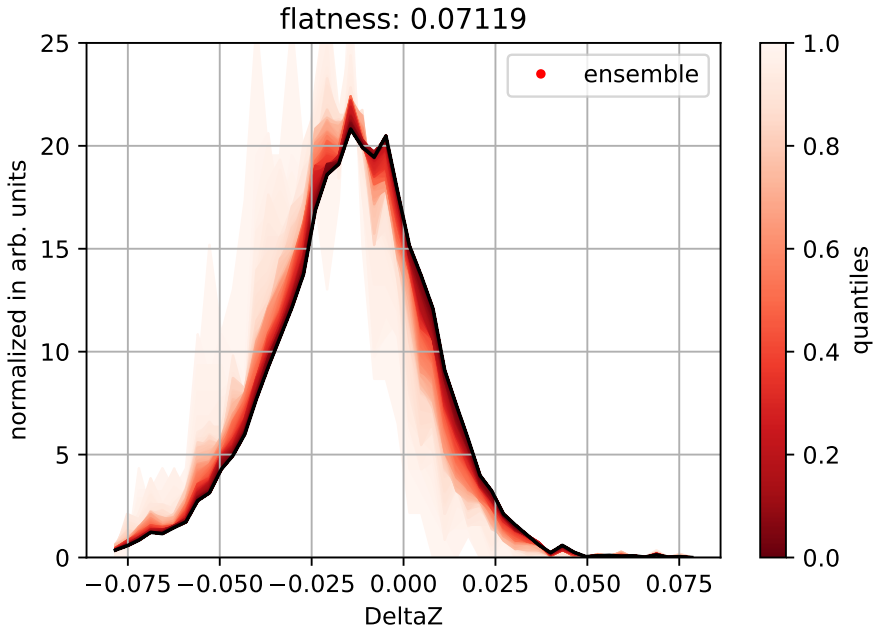
Figure 6.7.: Training and validation losses for the training of Three Streamer models using different implementations for λ. The training loss is shown in red and the validation loss in blue. To address fluctuations, the mean of the last 15 validation losses is determined and used for early stopping. The epoch from which no more improvement takes place is shown in light blue.

6.7.2. Decorrelation Power

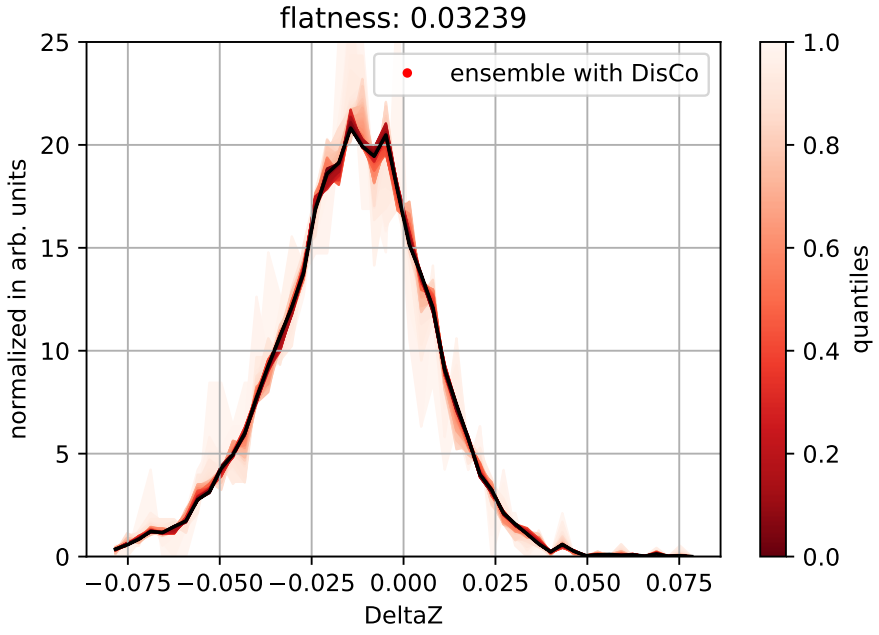
In order to evaluate the effectiveness of the decorrelation with DisCo applied on an ensemble of Three Streamers, the resulting bias on the Δz distribution is taken into account and compared with the ensemble from Section 6.6.

To access the degree of correlation with Δz, the Classifier Output Dependent (COD) distribution of Δz is plotted for both ensembles which are shown in Figure 6.8. This method was already used in [3] to measure the classifier correlation with Δz. Since the Δz distribution of background events is not significant, these COD distributions only take signal events into account. The black lines show the normalized Δz distribution without an applied classifier cut. Additionally, 100 classifier cuts are plotted in different red tones. These cuts are based on the quantiles of the classifier output. Low quantile cuts are shown as dark red while high quantile cuts fade to light red. For a cut on a given quantile of the classifier output the resulting Δz distribution is plotted in the corresponding red tone. Since only signal events are taken into account, 40 % of all signal events would be cut away in the example. The number of signal events used in this thesis is low in general, which explains the non-smooth behavior of the Δz distributions in the COD curves.

In addition, a flatness score [3] is computed to quantify the deviations of the quantile distributions compared to the uncut Δz distribution. Therefore, the Δz distribution is equal frequency binned which results in a flat distribution. Then 100 quantile cuts are applied to this distribution which should result in a flat distribution in an optimal case. To determine the flatness of these quantile cuts the squared differences are summed up and normed. This concept was already used in [31], for its implementation [63] was used. The flatness score is plotted on top of the COD diagrams in Figure 6.8.



(a) Ensemble without distance correlation.



(b) Ensemble with distance correlation. λ is set to zero and increased by 3 per epoch, starting at epoch ten.

Figure 6.8.: Comparison of COD curves of an ensemble (Figure 6.8a) and an ensemble trained with DisCo (Figure 6.8b).

Figure 6.8a shows the correlation between the ensemble from Section 6.6 with Δz. It can be seen, that there are significant deviations for high quantile cuts. Low quantile cuts already lead to a visible bias in the Δz distribution. The higher the quantile cuts, the more significant the bias. The decorrelation results in Figure 6.8b show, that the application of DisCo reduces the red deviations clearly. The bias for low quantile cuts seems to be suppressed completely. For quantile cuts around the 50 % quantile, deviations on the peak of the distribution are present. Biases for low quantile cuts are still visible but are reduced compared to the ensemble without a decorrelation mechanism. The bias reduction is also notable considering the flatness scores. The decorrelation using DisCo reduces the flatness score by over 50 %.

The comparison between the original ensemble and the decorrelated one using Disco clearly shows the significant reduction of the bias in Δz. Therefore one can conclude, that the decorrelation with DisCo is an effective and handy method to decorrelate a classifier output from Δz. It would be a motivation for future work to use a DisCo-decorrelated model in a CPV study to test the decorrelation effect in such studies.

The comparison of the decorrelation results presented in this thesis with the ones from previous work is not possible. Figure 6.9 shows a comparison between the COD curve of reproduced MLP and a COD curve for the corresponding MLP (DNN) from [3]. The figure shows clearly, that the bias is significantly higher for an MLP based on the actual dataset of MC13a compared to the one based on MC7. One can therefore not assume, that the results of this work are comparable to one on other datasets.

6.7.3. Separation Power

The performance of the ensemble trained with DisCo is compared with the normal one and the reproduced MLP (see Section 6.4). Corresponding ROC curves are plotted in Figure 6.10.

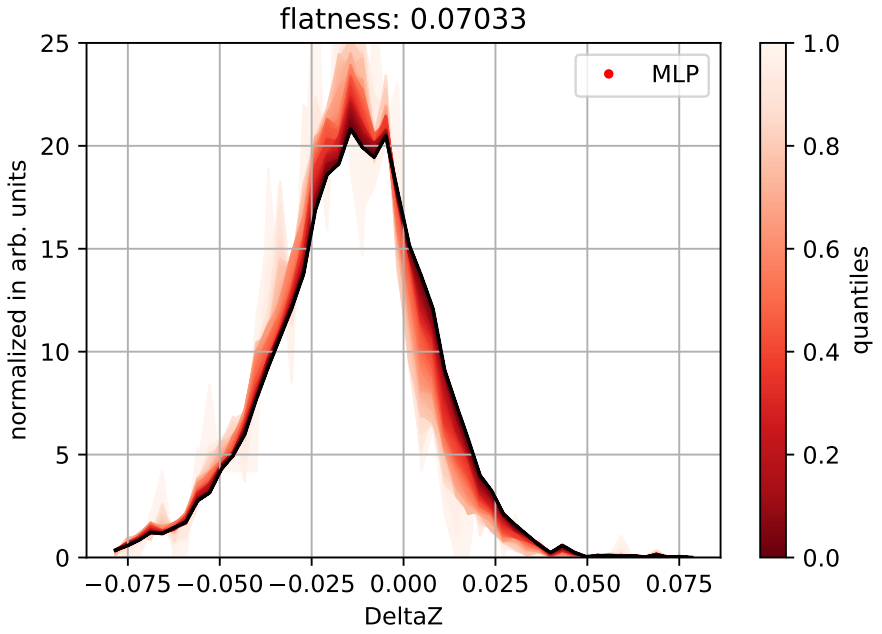
The results show, that an ensemble without DisCo outperforms both other models, the MLP and the normal ensemble. The ensemble trained with DisCo achieves a ROC AUC score of 0.9819 and therefore cannot compete with the MLP. Obviously, the decorrelation term hinders the performance of the model and leads to a trade-off between the degree of decorrelation and performance.

6.7.4. Uncertainties

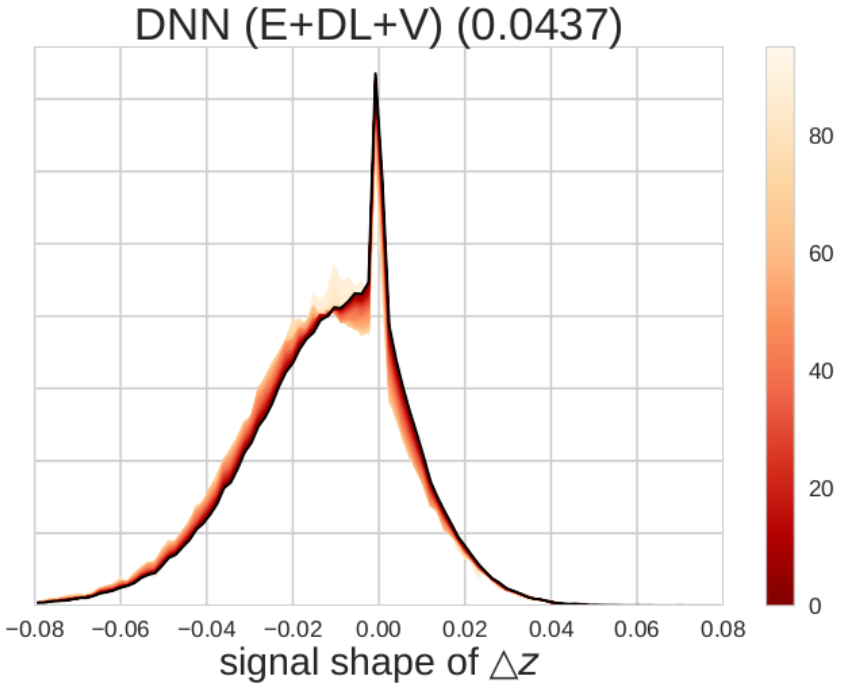
The Uncertainty vs. Prediction plots for the ensemble trained with DisCo are given in Figure 6.11. The overall behavior of the uncertainties is similar to the ones shown in Figure 6.6a. Nevertheless, several differences can be found in the details of the distributions.

At first, the minimum gaps seem to be smaller compared to the ones in Figure 6.6c. This can be traced back to the fact, that the DisCo term complicates the training and it seems to lead to a lower variety in the modes the ensemble members converge to. Therefore, the predictions of the ensemble members differ less from each other which finally leads to tighter uncertainty limits.

In the signal only plot, the lower uncertainty limits tend more to values near zero. But more statistics is needed in regions with low prediction for a reliable statement. In the



(a) COD curve of an MLP based on MC13a.



(b) COD curve of an MLP based on MC7. The peak is due to failed vertex fits. Taken from [3]

Figure 6.9.: Comparison between the correlation of an MLP (DNN) based on different datasets. Figure 6.9a shows the corresponding COD curve based on MC13a, Figure 6.9b is the corresponding one based on MC7.

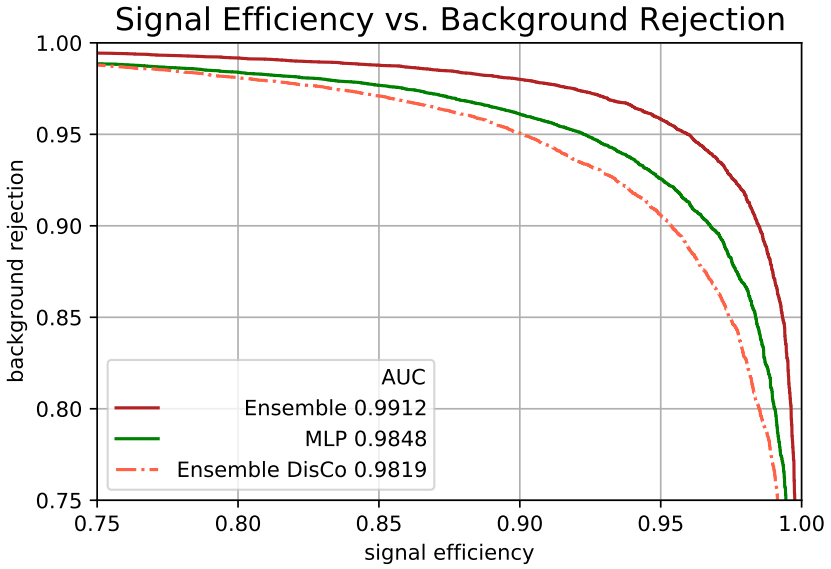
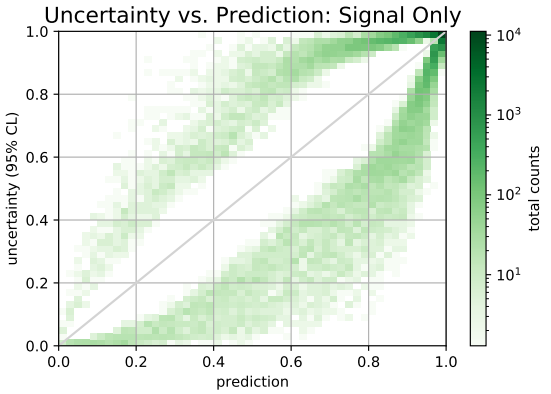


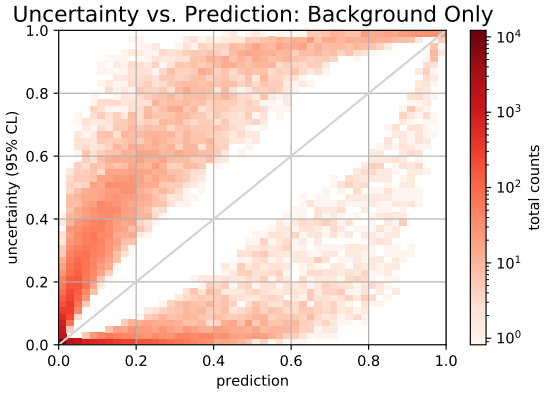
Figure 6.10.: Background rejection over signal efficiency for different models. The performance of an MLP (see Section 6.4) is plotted in green. The models designed in the scope of this work are the ensemble (dark red) and the ensemble trained with DisCo (light red).

background only plot, the tendency of the upper uncertainty limits seems less pronounced compared to the ones in Figure 6.6b.

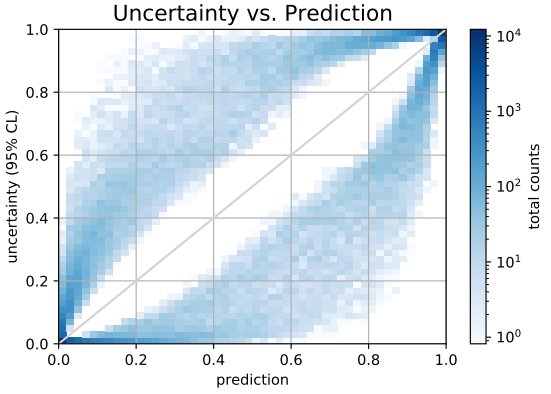
In general, the uncertainties of the ensemble trained with DisCo displays the expected behavior. Nevertheless, to understand the details of their behavior, further studies are needed.



(a) Signal only



(b) Background only



(c) Signal and background

Figure 6.11.: Uncertainty vs. prediction plots for a Deep Ensemble. The x-axis shows the predicted value (i.e. the mean) of the ensemble. On the y-axis the absolute uncertainty limits of the ensemble shown. For this, the upper (above grey line) and lower (under grey line) uncertainty limits for given predictions are shown as scatter plots. Since the dataset is imbalanced, all plots are produced with class weights.

7. Summary and Outlook

The goal of this master's thesis was to extend the Deep Continuum Suppression by three mechanisms: an invariant particle input, predictive uncertainties, and a decorrelation from an analysis variable, which was performed with Δz. To do so, a Three Streamer model using a self-attention-based input mechanism allowing for invariance under the particle order was designed. Afterward, predictive uncertainties were added to the DCS by using this Three Streamer model in a Deep Ensemble approach to predict well-calibrated and interpretable uncertainties. To decorrelate this ensemble from Δz, Distance Correlation is used to capture non-linear patterns and is appended as a regularisation term in the model's cost function.

All mechanisms were implemented and evaluated in an applied Continuum Suppression based on MC13a data. To compare their performance an MLP from previous work was reproduced a as benchmark model [3].

This work showed, that DisCo can be easily applied to a model by decreasing the correlation with Δz by over 50 %, as measured by the flatness score. However, since the decorrelated ensemble cannot compete with the performance of either an MLP or of a normal ensemble, a trade-off between the degree of decorrelation and the discrimination power of the ensemble was observed. In contrast, the Three Streamer model, as well as its ensemble, could outperform both the decorrelated ensemble and the MLP. The ensemble of Three Streamers performs slightly better than the single Three Streamer model. In conclusion, the performance of the DCS could be increased notably.

The uncertainties of the Deep Ensemble are evaluated by uncertainty vs. prediction plots. These plot show, that ensembles are able to predict asymmetric, well-calibrated and therefore interpretable uncertainties. Additionally, a minimum gap was observed for the upper and lower uncertainty limits. The upper uncertainty limits of background events had a clear tendency towards a signal probability of 100 %. This effect was less pronounced in the decorrelated ensemble.

Despite the fact that the performance of the DCS could be improved, there is still potential for further improvements. The self-attention-based input mechanism is restricted to particles with the same features. This could be optimized by finding a way to combine track and cluster variables in a self-attention-based input to extract information between tracks and clusters too. To optimize the application of Deep Ensembles, a study to analyse the dependency between the size of the ensemble and the behaviour of their uncertainties could

be made. Additionally, a comparison between uncertainties in a Continuum Suppression of an Deep Ensemble and a Bayesian Network would be thinkable. The next step for the decorrelation with DisCo would be to test the decorrelation in a CPV study for its suitability.

Bibliography

[1] **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.

[2] **Belle Collaboration**, K. Abe *et al.*, "Observation of Mixing-induced CP Violation in the Neutral B Meson System," *Phys. Rev. D* **66** (Aug, 2002) 032007. https://link.aps.org/doi/10.1103/PhysRevD.66.032007.

[3] D. Weyland, "Continuum Suppression with Deep Learning Techniques for the Belle II Experiment," Master's thesis, Karlsruhe Institute of Technology (KIT), 2017.

[4] B. Lakshminarayanan *et al.*, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," arXiv:1612.01474 [stat.ML].

[5] G. Kasieczka and D. Shih, "Robust Jet Classifiers through Distance Correlation," *Phys. Rev. Lett.* **125** no. 12, (2020) 122001, arXiv:2001.05310 [hep-ph].

[6] 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," *Physics Letters B* **716** no. 1, (2012) 1–29. https://www.sciencedirect.com/science/article/pii/S037026931200857X.

[7] MissMJ and Cush, "Elementary Particles Included in the Standard Model," 2019. https://de.wikipedia.org/wiki/Standardmodell.

[8] V. C. Rubin and W. K. Ford, Jr., "Rotation of the Andromeda Nebula from a Spectroscopic Survey of Emission Regions," *Astrophys. J.* **159** (1970) 379–403.

[9] B. Pontecorvo, "Mesonium and Anti-mesonium," *Sov. Phys. JETP* **6** (1957) 429.

[10] E. Kearns *et al.*, "Detecting Massive Neutrinos," *Scientific American* **281** no. 2, (1999) 64–71. http://www.jstor.org/stable/26058368.

[11] J. H. Christenson *et al.*, "Evidence for the 2π Decay of the K_2^0 Meson," *Phys. Rev. Lett.* **13** (Jul, 1964) 138–140. https://link.aps.org/doi/10.1103/PhysRevLett.13.138.

[12] M. Kobayashi and T. Maskawa, "CP-Violation in the Renormalizable Theory of Weak Interaction," *Progress of Theoretical Physics* **49** no. 2, (02, 1973) 652–657, https://academic.oup.com/ptp/article-pdf/49/2/652/5257692/49-2-652.pdf. https://doi.org/10.1143/PTP.49.652.

[13] **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.

[14] B. Wang, "Searches for New Physics at the Belle II Experiment," arXiv:1511.00373 [hep-ex].

[15] **Belle-II**, W. Altmannshofer *et al.*, "The Belle II Physics Book," *PTEP* **2019** no. 12, (2019) 123C01, arXiv:1808.10567 [hep-ex]. [Erratum: PTEP 2020, 029201 (2020)].

[16] **SuperKEKB**, K. Akai *et al.*, "SuperKEKB Collider," *Nucl. Instrum. Meth. A* **907** (2018) 188–199, arXiv:1809.01958 [physics.acc-ph].

[17] N. Toge, "KEK B-factory Design Report," Tech. Rep. KEK-Report-95-7, KEK, Tsukuba, 1995. https://cds.cern.ch/record/475260.

[18] **Particle Data Group**, P. Zyla *et al.*, "Review of Particle Physics," *Progress of Theoretical and Experimental Physics* **2020** no. 8, (08, 2020) , https://academic.oup.com/ptep/article-pdf/2020/8/083C01/34673722/ptaa104.pdf. https://doi.org/10.1093/ptep/ptaa104. 083C01.

[19] @belle2collab (Belle II Experiment), "Belle II at SuperKEKB Has Reached a New Luminosity World Record," 2021. https://twitter.com/belle2collab/status/1407985979953012743.

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

[21] R. Richter *et al.*, "Design and Technology of DEPFET Pixel Sensors for Linear Collider Applications," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **511** no. 1, (2003) 250–256. https://www.sciencedirect.com/science/article/pii/S0168900203018023. Proceedings of the 11th International Workshop on Vertex Detectors.

[22] A. Baur, "Trigger Studies for $e+e- \to \pi+\pi-\gamma$ and $e+e- \to \mu+\mu-$ and Implementation of a New DAQ Logging and Monitoring System at Belle II," Master's thesis, Karlsruhe Institute of Technology (KIT), 2020.

[23] Belle II software group, "Documentation - BASF2 Framework," 2021. https://software.belle2.org/.

[24] R. Brun and F. Rademakers, "ROOT: An Object Oriented Data Analysis Framework," *Nucl. Instrum. Meth. A* **389** (1997) 81–86.

[25] **BaBar, Belle**, A. J. Bevan *et al.*, "The Physics of the B Factories," *Eur. Phys. J. C* **74** (2014) 3026, arXiv:1406.6311 [hep-ex].

[26] D. Besson and T. Skwarnicki, "Upsilon Spectroscopy: Transitions in the Bottomonium System," *Annual Review of Nuclear and Particle Science* **43** no. 1, (1993) 333–378, https://doi.org/10.1146/annurev.ns.43.120193.002001. https://doi.org/10.1146/annurev.ns.43.120193.002001.

[27] M. Röhrken, "Time-Dependent CP Violation Measurements : Analyses of Neutral B Meson to Double-Charm Decays at the Japanese Belle Experiment," 2014. http://swbplus.bsz-bw.de/bsz39581362xcov.htmhttps://doi.org/10.1007/978-3-319-00726-7.

[28] E. Farhi, "A QCD Test for Jets," *Phys. Rev. Lett.* **39** (1977) 1587–1588.

[29] **CLEO**, D. M. Asner *et al.*, "Search for Exclusive Charmless Hadronic B Decays," *Phys. Rev. D* **53** (1996) 1039–1050, arXiv:hep-ex/9508004.

[30] G. C. Fox and S. Wolfram, "Observables for the Analysis of Event Shapes in e^+e^- Annihilation and Other Processes," *Phys. Rev. Lett.* **41** (Dec, 1978) 1581–1585. https://link.aps.org/doi/10.1103/PhysRevLett.41.1581.

[31] T. Keck, "FastBDT: A Speed-optimized and Cache-friendly Implementation of Stochastic Gradient-boosted Decision Trees for Multivariate Classification," *CoRR* **abs/1609.06119** (2016) , arXiv:1609.06119. http://arxiv.org/abs/1609.06119.

[32] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," arXiv:1406.2661 [stat.ML].

[33] A. Santoro *et al.*, "A Simple Neural Network Module for Relational Reasoning," *CoRR* **abs/1706.01427** (2017) , arXiv:1706.01427. http://arxiv.org/abs/1706.01427.

[34] F. Rosenblatt, "The perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.," *Psychological Review* **65** no. 6, (1958) 386–408. http://dx.doi.org/10.1037/h0042519.

[35] C. Patterson, "Managing a Real-time Massively-parallel Neural Architecture," 01, 2012.

[36] C. Dilmegani, "Dark side of Neural Networks Explained," 2021. https://research.aimultiple.com/how-neural-networks-work/.

[37] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com.

[38] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[39] D. Rumelhart *et al.*, "Learning Representations by Back-propagating Errors," *Nature* **323** (1986) 533–536.

[40] A. Vaswani *et al.*, "Attention Is All You Need," arXiv:1706.03762 [cs.CL].

[41] Y. Goldberg and O. Levy, "word2vec Explained: Deriving Mikolov et al.'s Negative-sampling Word-embedding Method," *CoRR* **abs/1402.3722** (2014) , arXiv:1402.3722. http://arxiv.org/abs/1402.3722.

[42] J. Cheng *et al.*, "Long Short-Term Memory-Networks for Machine Reading," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 551–561. Association for Computational Linguistics, Austin, Texas, Nov., 2016. https://aclanthology.org/D16-1053.

[43] J. Alammar, "The Illustrated Transformer," 2018. http://jalammar.github.io/illustrated-transformer/?utm_source=share&utm_medium=ios_app.

[44] K. Doshi, "Transformers Explained Visually (Part 1-4)," 2020. https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452.

[45] K. He *et al.*, "Deep Residual Learning for Image Recognition," arXiv:1512.03385 [cs.CV].

[46] J. L. Ba *et al.*, "Layer Normalization," arXiv:1607.06450 [stat.ML].

[47] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167 [cs.LG].

[48] C. Guo *et al.*, "On Calibration of Modern Neural Networks," *CoRR* **abs/1706.04599** (2017) , arXiv:1706.04599. http://arxiv.org/abs/1706.04599.

[49] J. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Adv. Large Margin Classif.* **10** (06, 2000) .

[50] J. De Leeuw *et al.*, "Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods," *Journal of Statistical Software* **32** (10, 2009) .

[51] J. Zhang, "Dive into Decision Trees and Forests: A Theoretical Demonstration," arXiv:2101.08656 [cs.LG].

[52] S. Fort *et al.*, "Deep Ensembles: A Loss Landscape Perspective," arXiv:1912.02757 [stat.ML].

[53] I. J. Goodfellow *et al.*, "Explaining and Harnessing Adversarial Examples," arXiv:1412.6572 [stat.ML].

[54] C. Blundell *et al.*, "Weight Uncertainty in Neural Networks," arXiv:1505.05424 [stat.ML].

[55] K. He *et al.*, "Deep Residual Learning for Image Recognition," arXiv:1512.03385 [cs.CV].

[56] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2016.

[57] G. J. Székely and M. L. Rizzo, "Brownian Distance Covariance," *The Annals of Applied Statistics* **3** no. 4, (2009) 1236 – 1265. https://doi.org/10.1214/09-AOAS312.

[58] K. Pearson, "Note on Regression and Inheritance in the Case of Two Parents," *Proceedings of the Royal Society of London* **58** (1895) 240–242. http://www.jstor.org/stable/115794.

[59] I.-K. Yeo and R. Johnson, "A new Family of Power Transformations to Improve Normality or Symmetry," *Biometrika* **87** (12, 2000) .

[60] K. Horak *et al.*, "Classification of SURF image features by selected machine learning algorithms," in *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 636–641. 2017.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds. 2015. http://arxiv.org/abs/1412.6980.

[62] O. Taubert *et al.*, "Loss Scheduling for Class-Imbalanced Image Segmentation Problems," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 426–431. 2020.

[63] T. Keck, "FastBDT," 2017. https://github.com/thomaskeck/FastBDT/blob/e67f71525612020acc78721031fca681d173c144/PyFastBDT/utility.py.

A. Continuum Suppression Features

A.1. Event Shape Variables

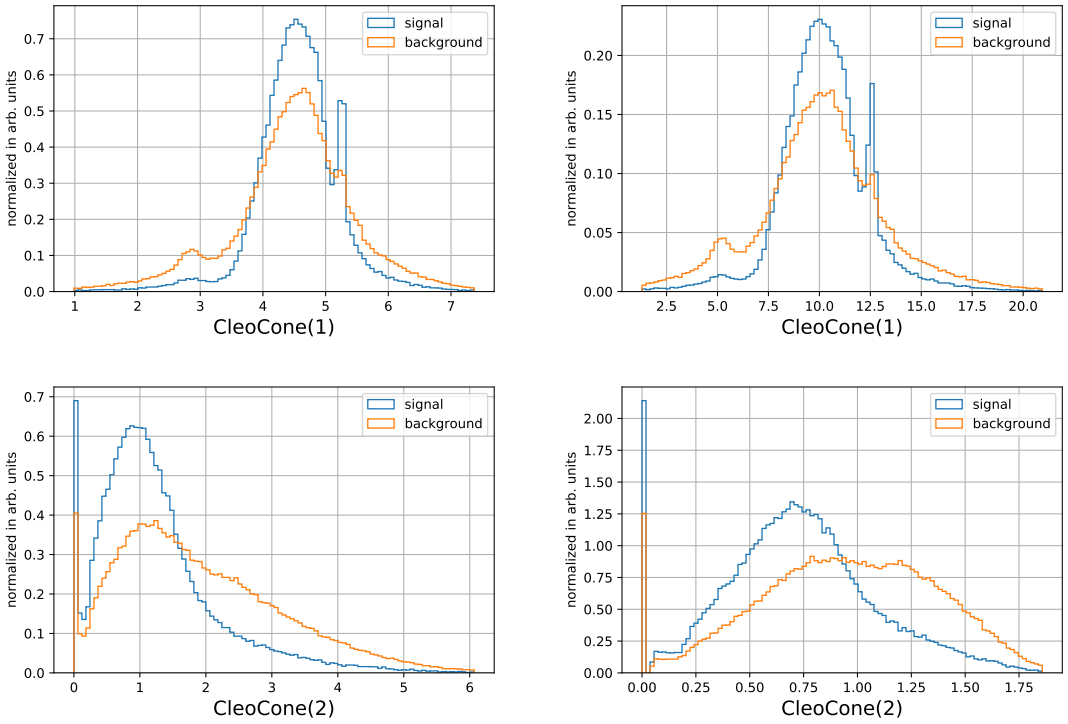


Figure A.1.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2).

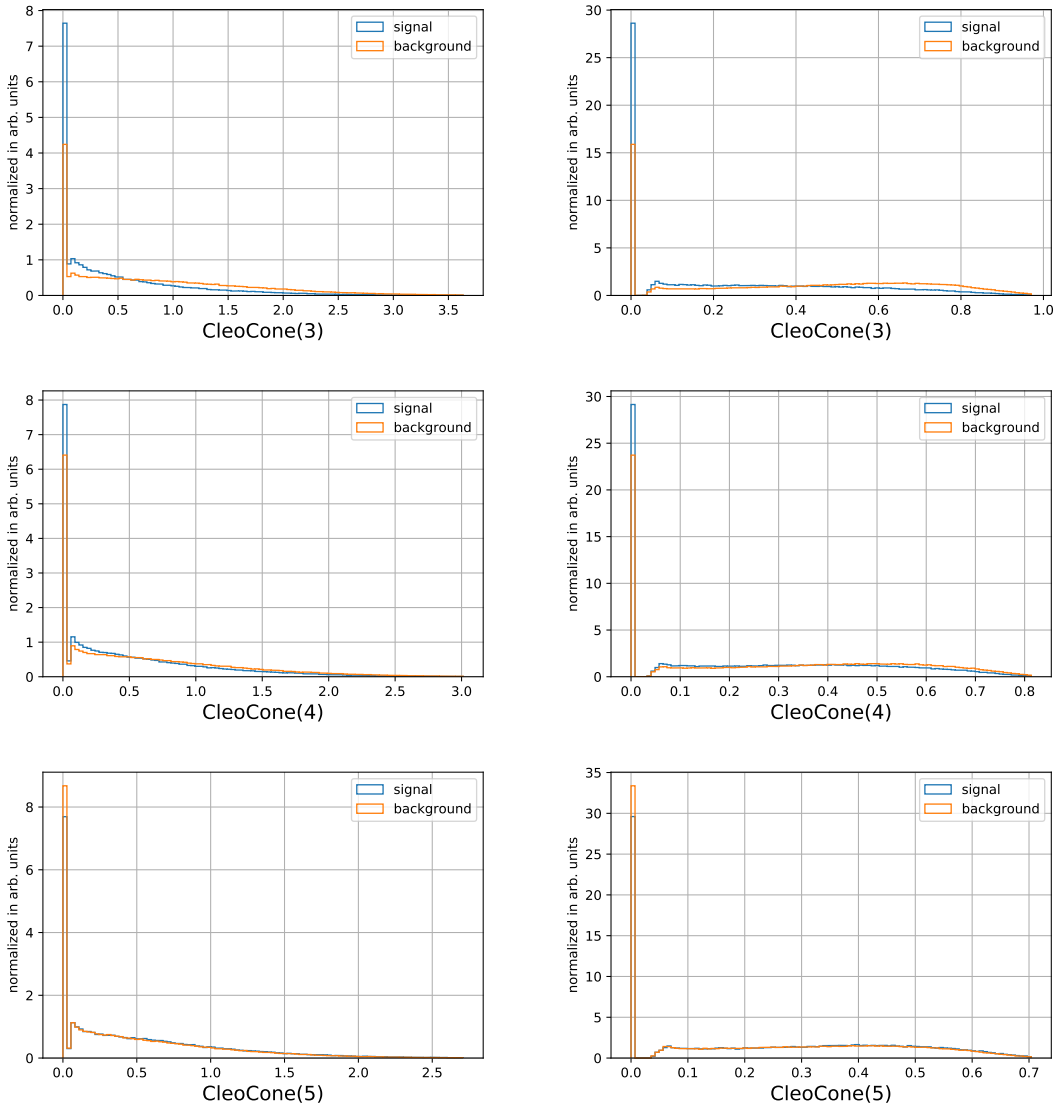


Figure A.2.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2).

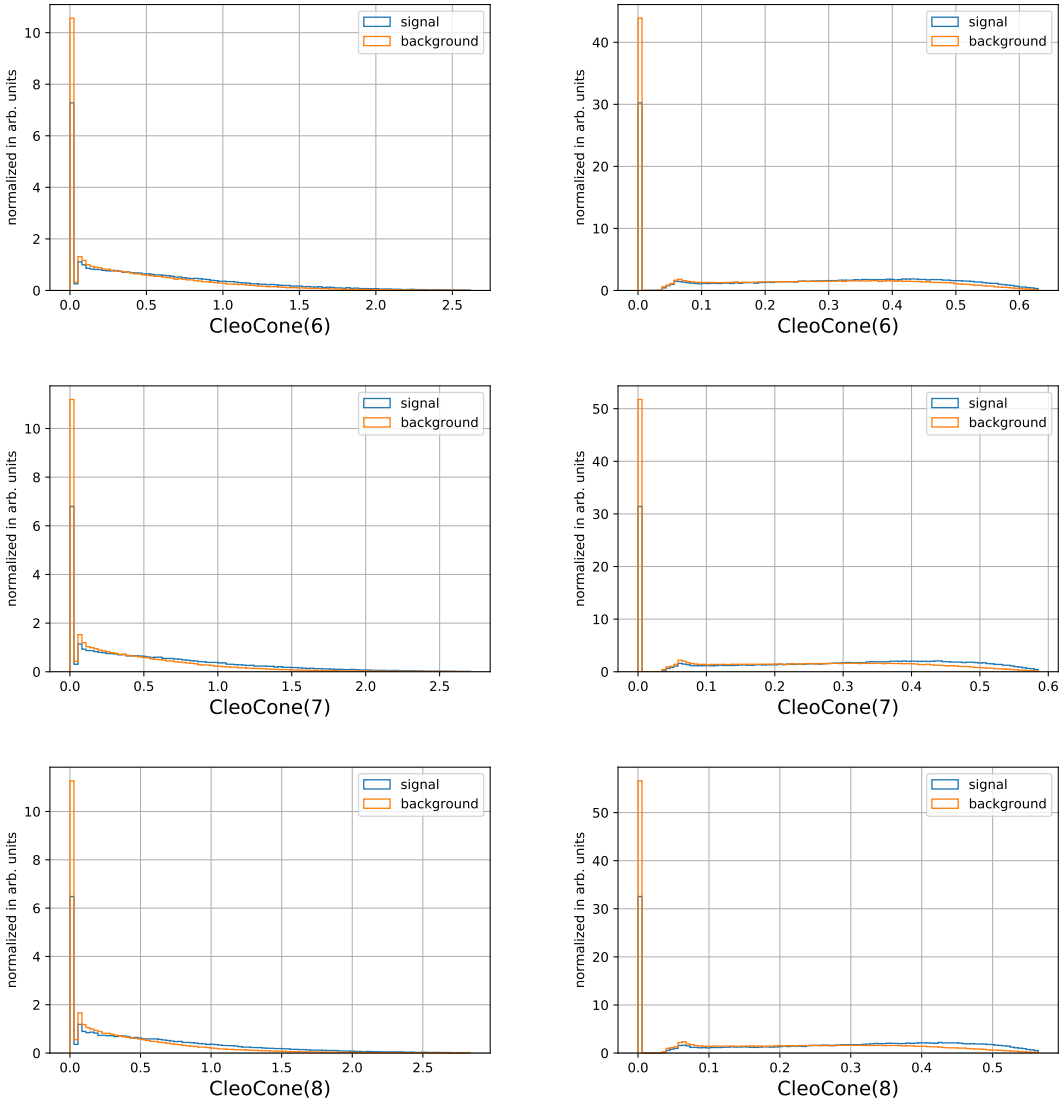


Figure A.3.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2).

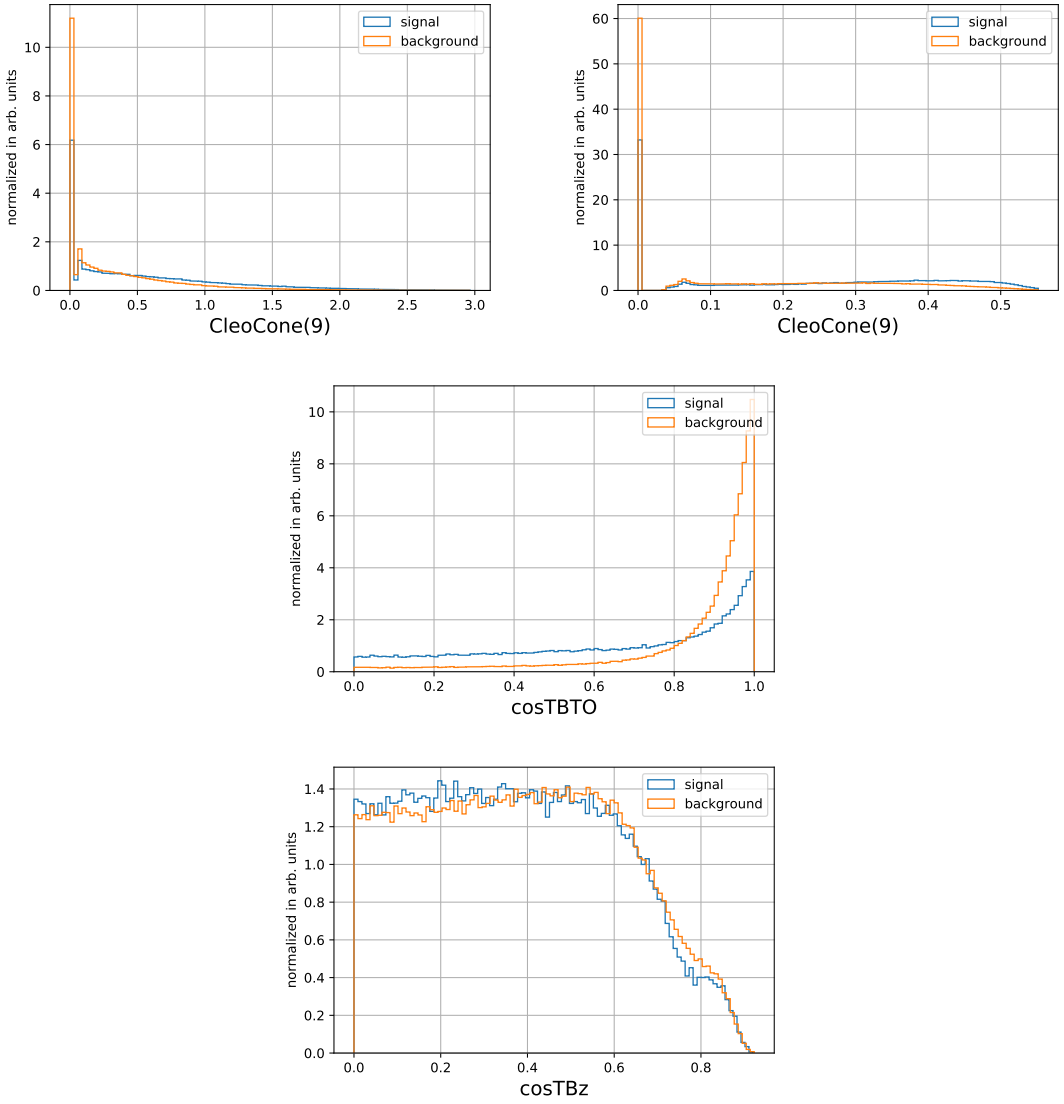


Figure A.4.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Centered features were not transformed. Not transformer features are shown in the middle.

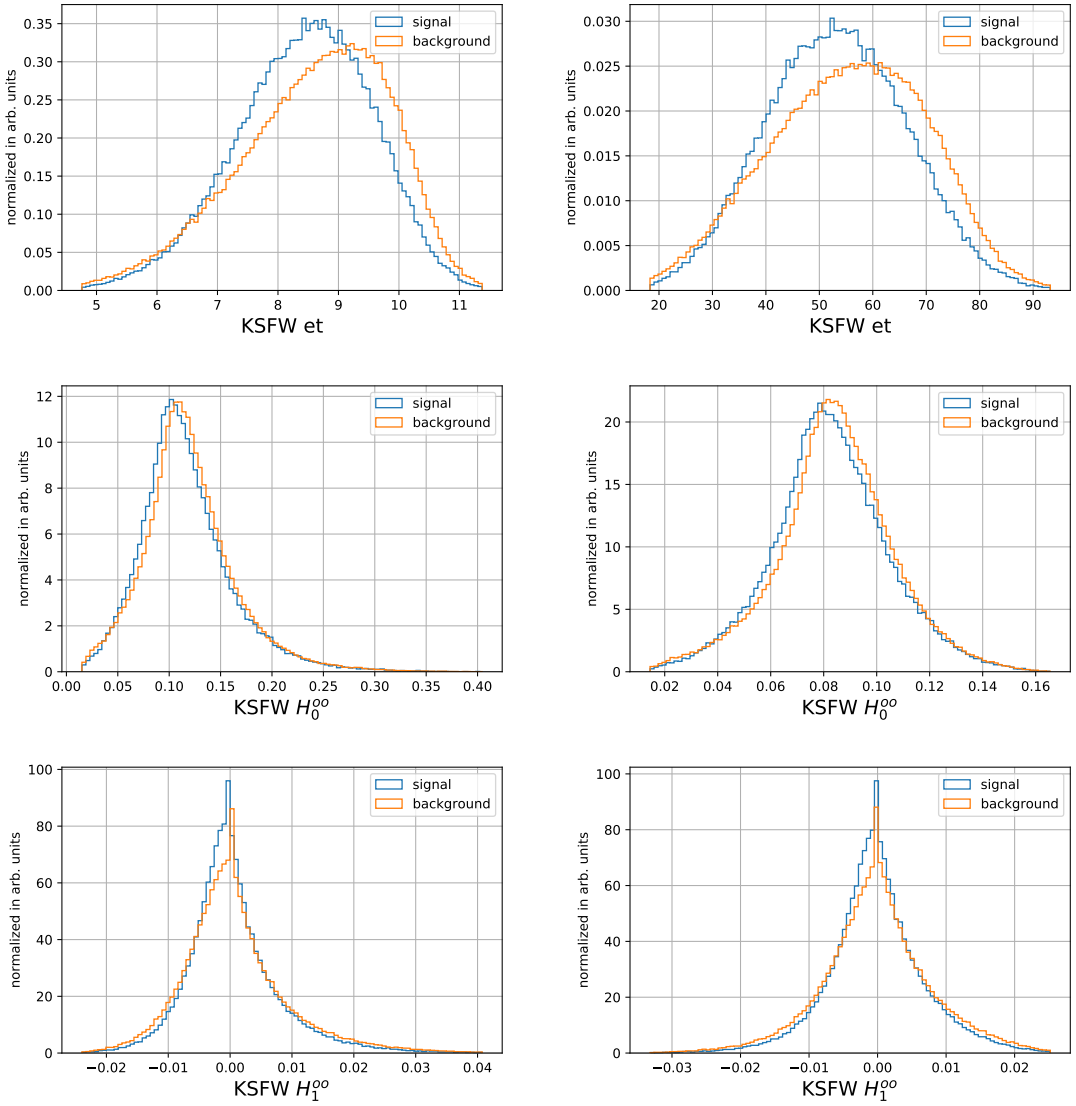


Figure A.5.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2).

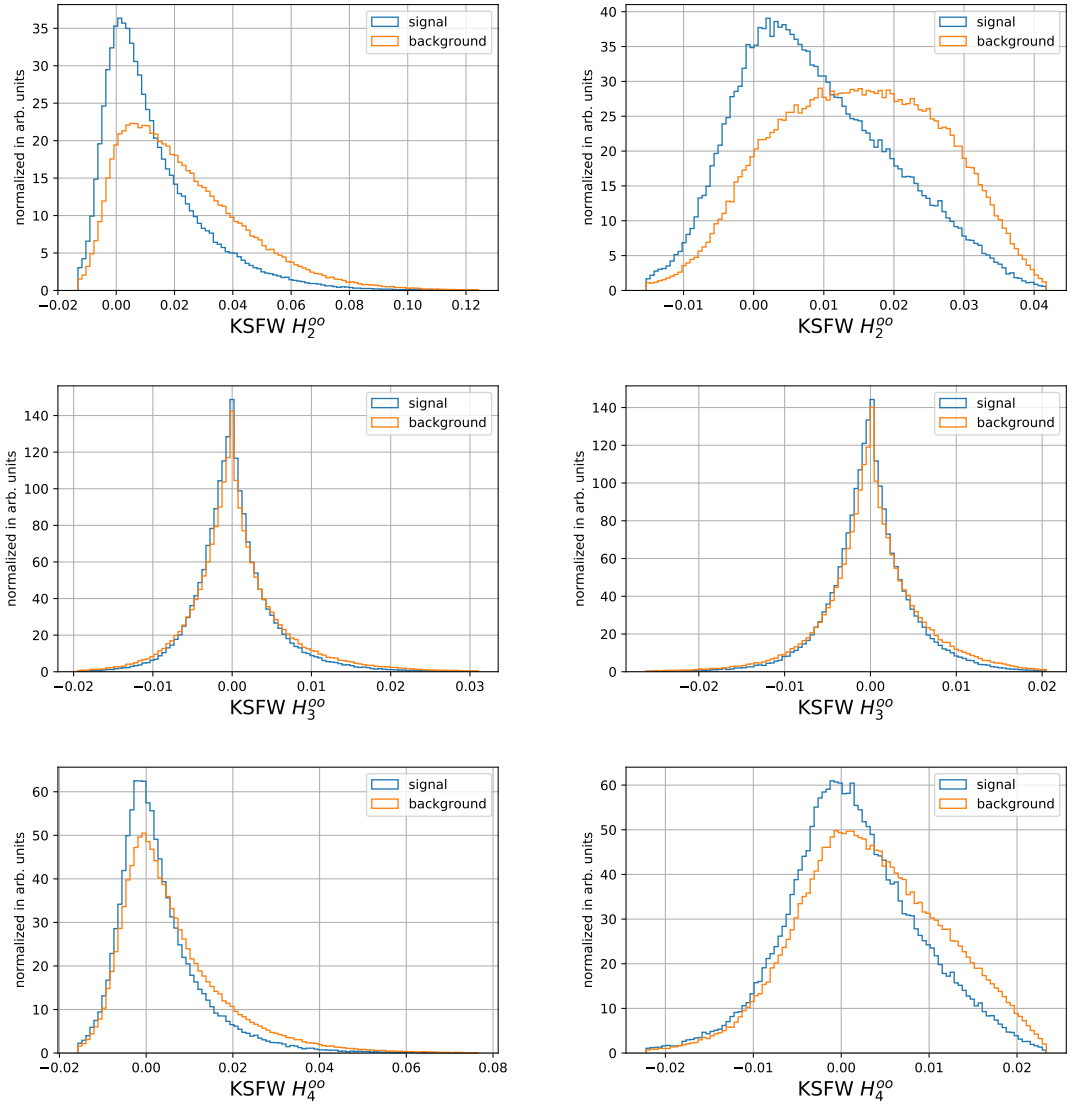


Figure A.6.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2).

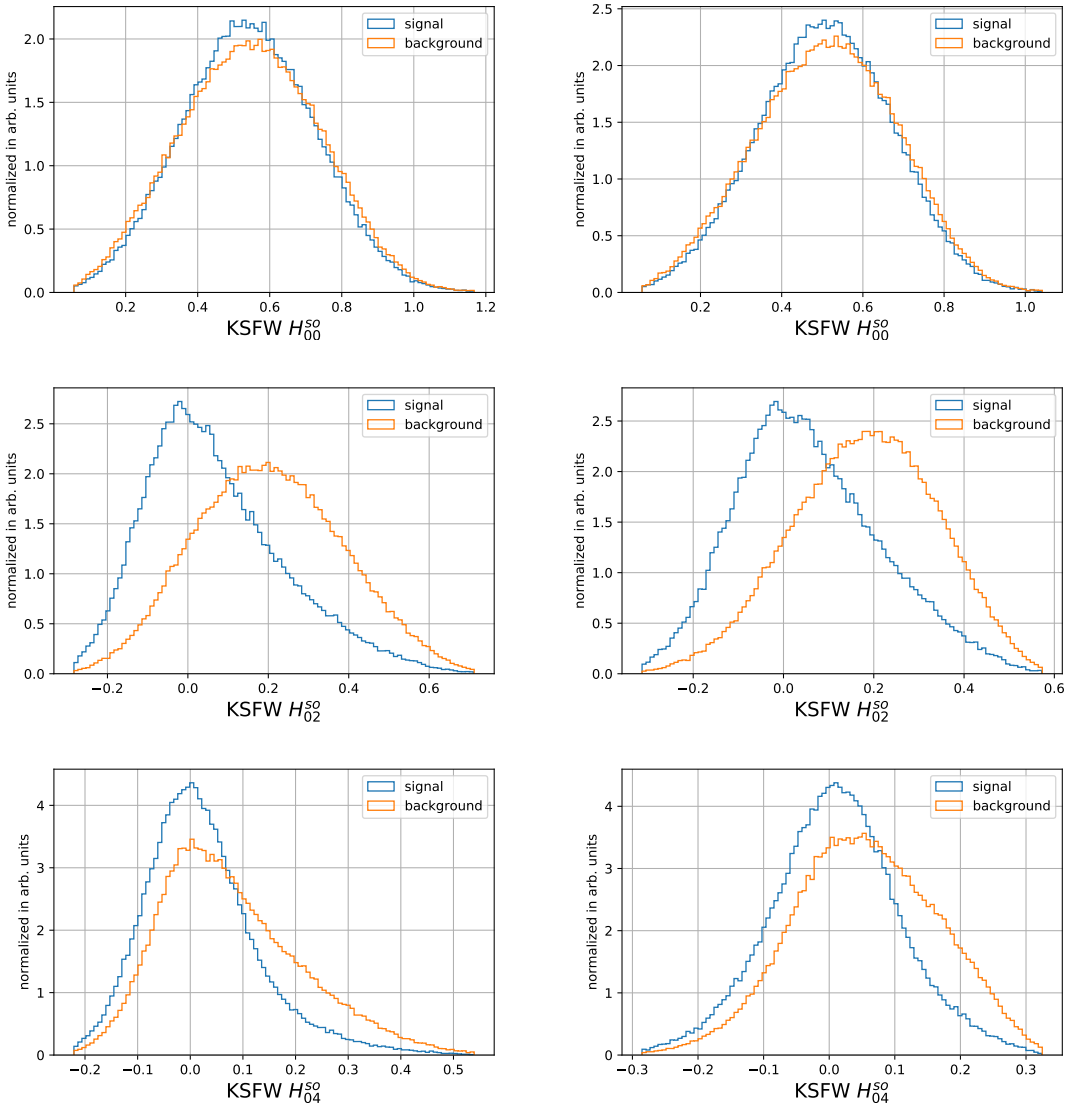


Figure A.7.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2).

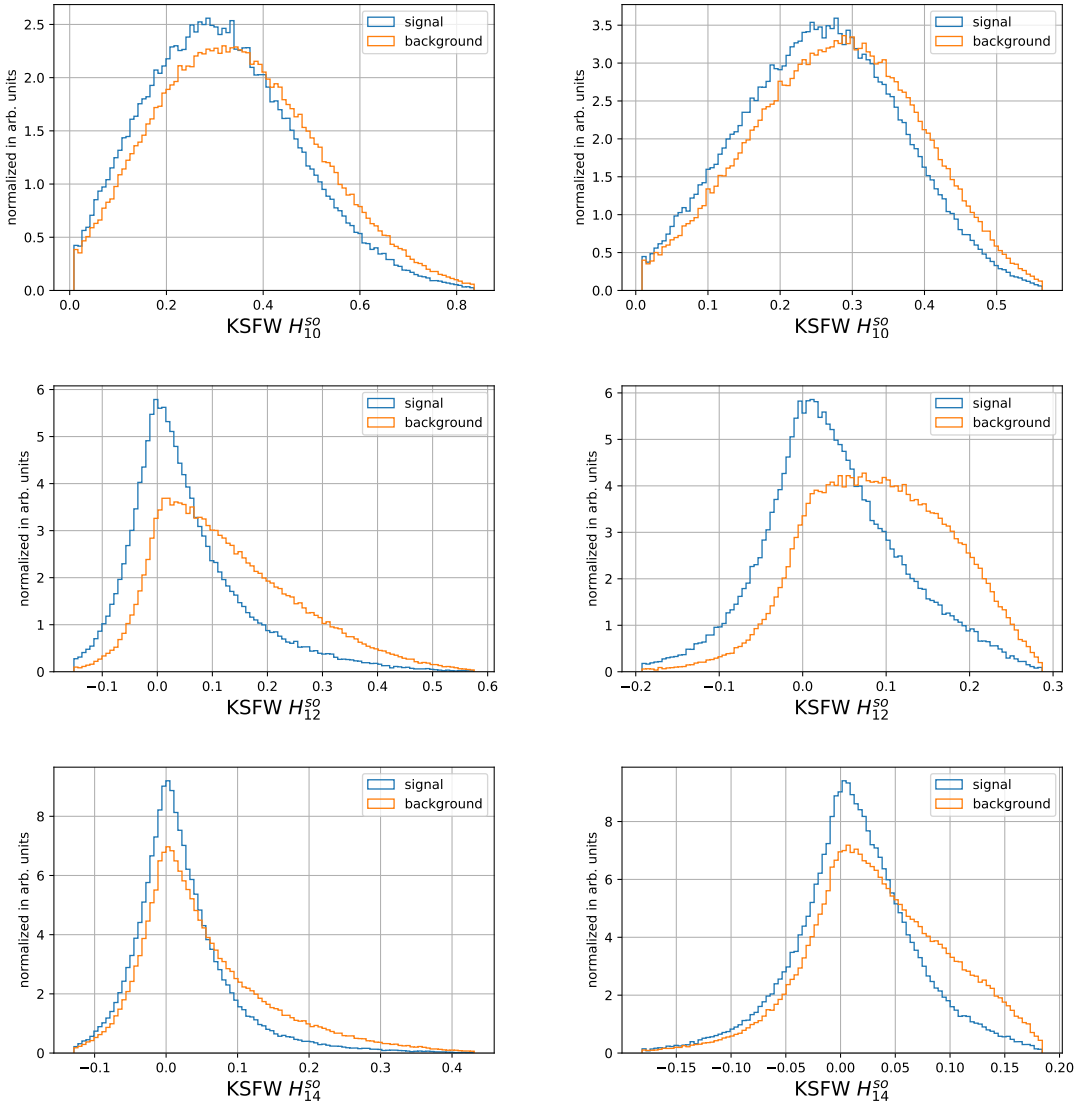


Figure A.8.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Not transformer features are shown in the middle.

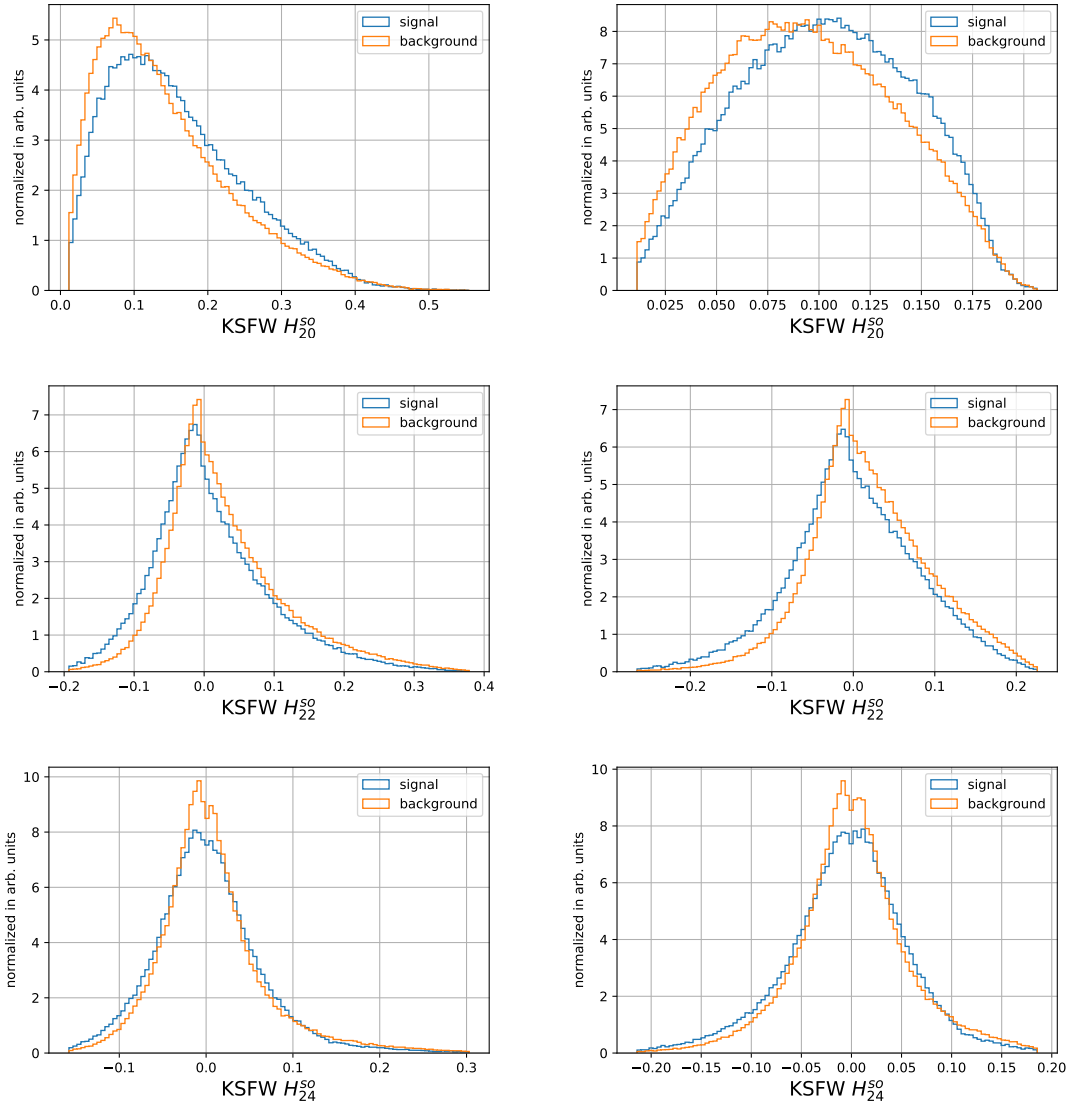


Figure A.9.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Not transformer features are shown in the middle.

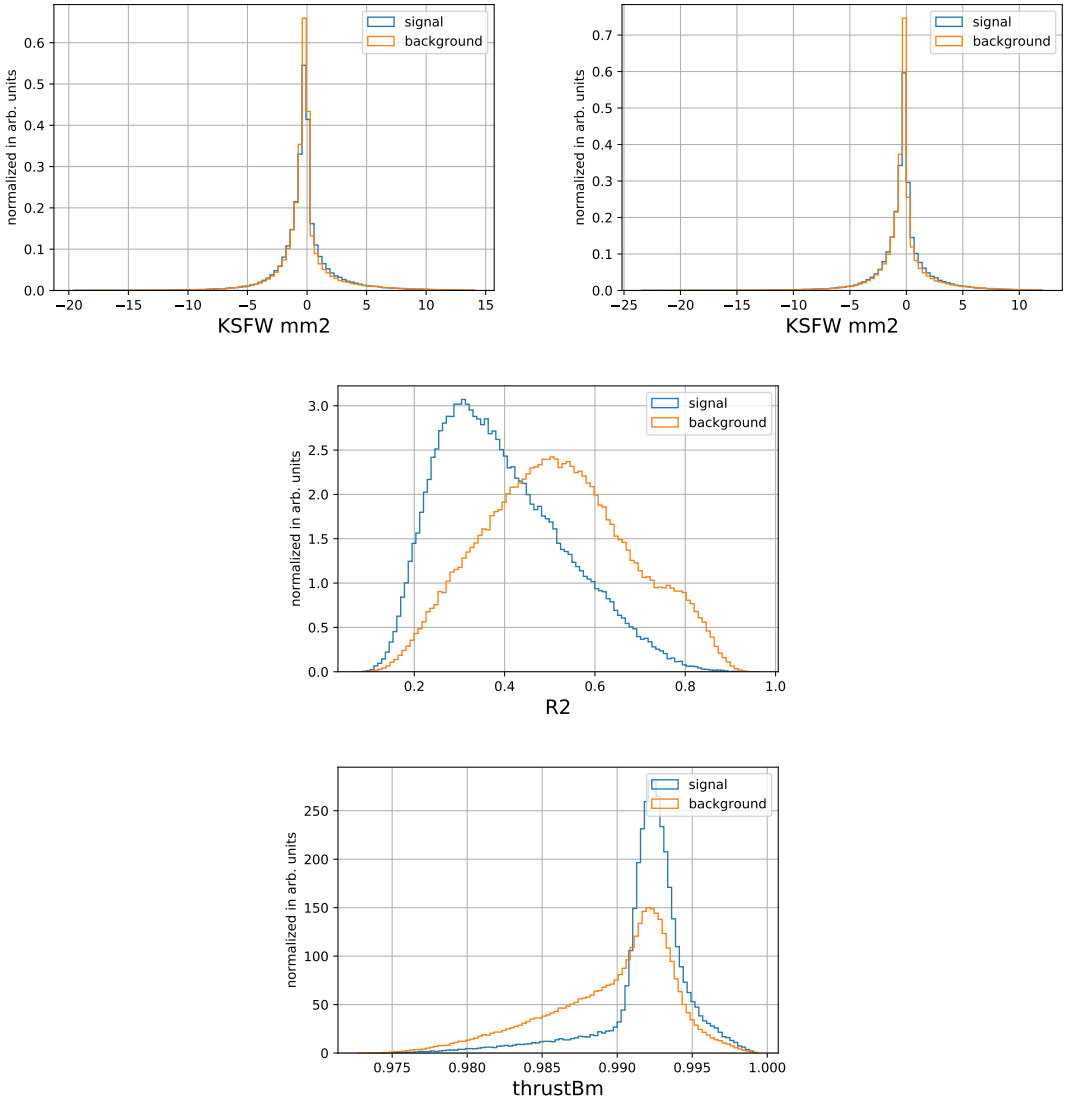


Figure A.10.: Normalized and trimmed distributions of event shape features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Centered features were not transformed. Not transformer features are shown in the middle.

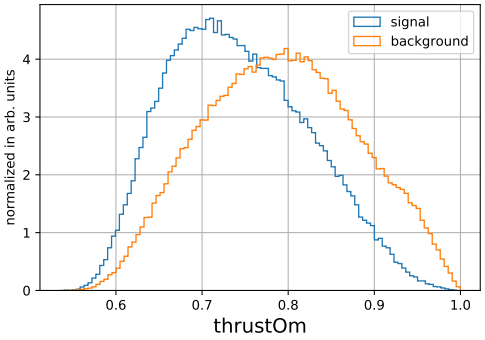


Figure A.11.: Normalized and trimmed distributions of cluster features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Not transformer features are shown in the middle.

A.2. Cluster Candidate Features

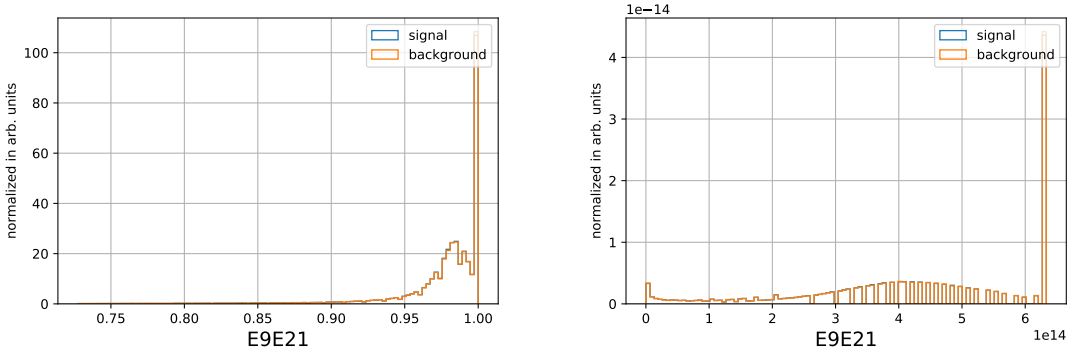


Figure A.12.: Normalized and trimmed distributions of cluster features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Not transformer features are shown in the middle.

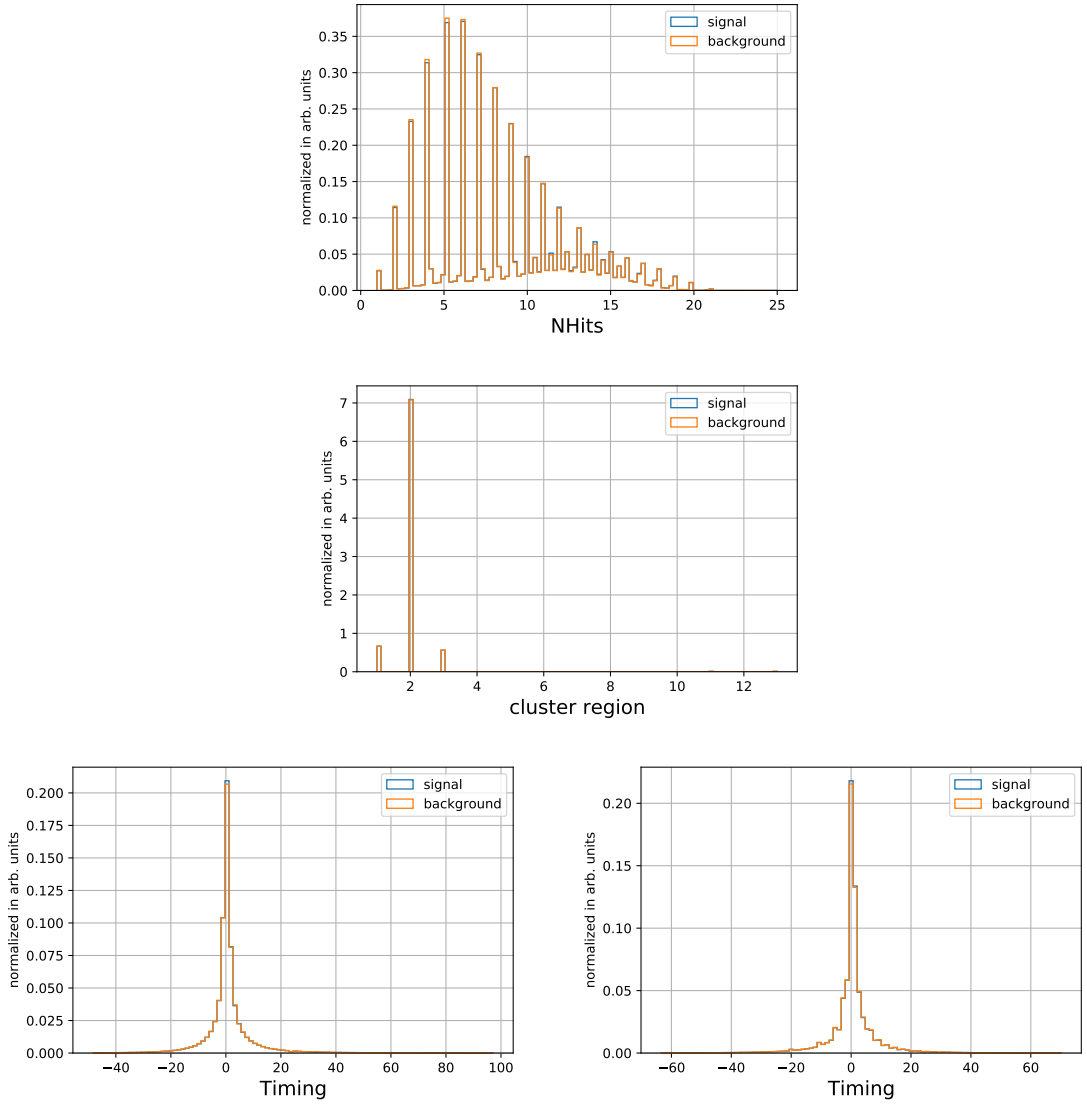


Figure A.13.: Normalized and trimmed distributions of cluster features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Centered features were not transformed. Not transformer features are shown in the middle.

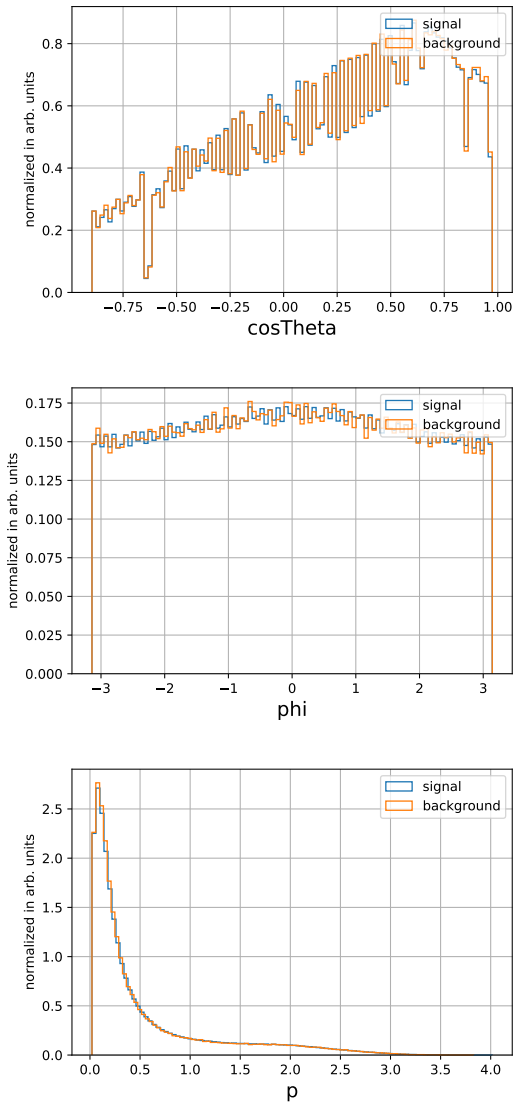


Figure A.14.: Normalized and trimmed distributions of cluster features used in the Continuum Suppression (see Section 4.2). The shown features were not transformed.

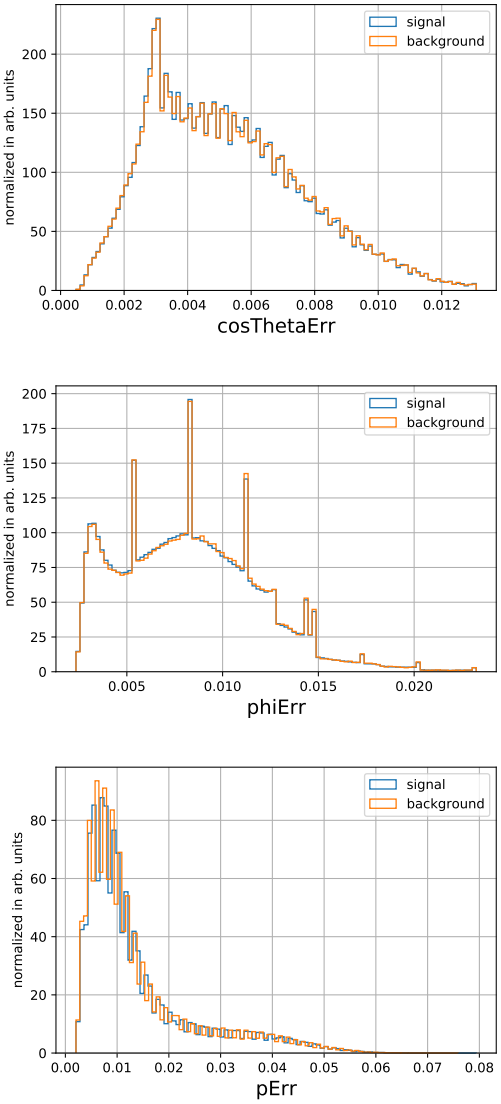


Figure A.15.: Normalized and trimmed distributions of cluster features used in the Continuum Suppression (see Section 4.2). The shown features were not transformed.

A.3. Track Candidate Features

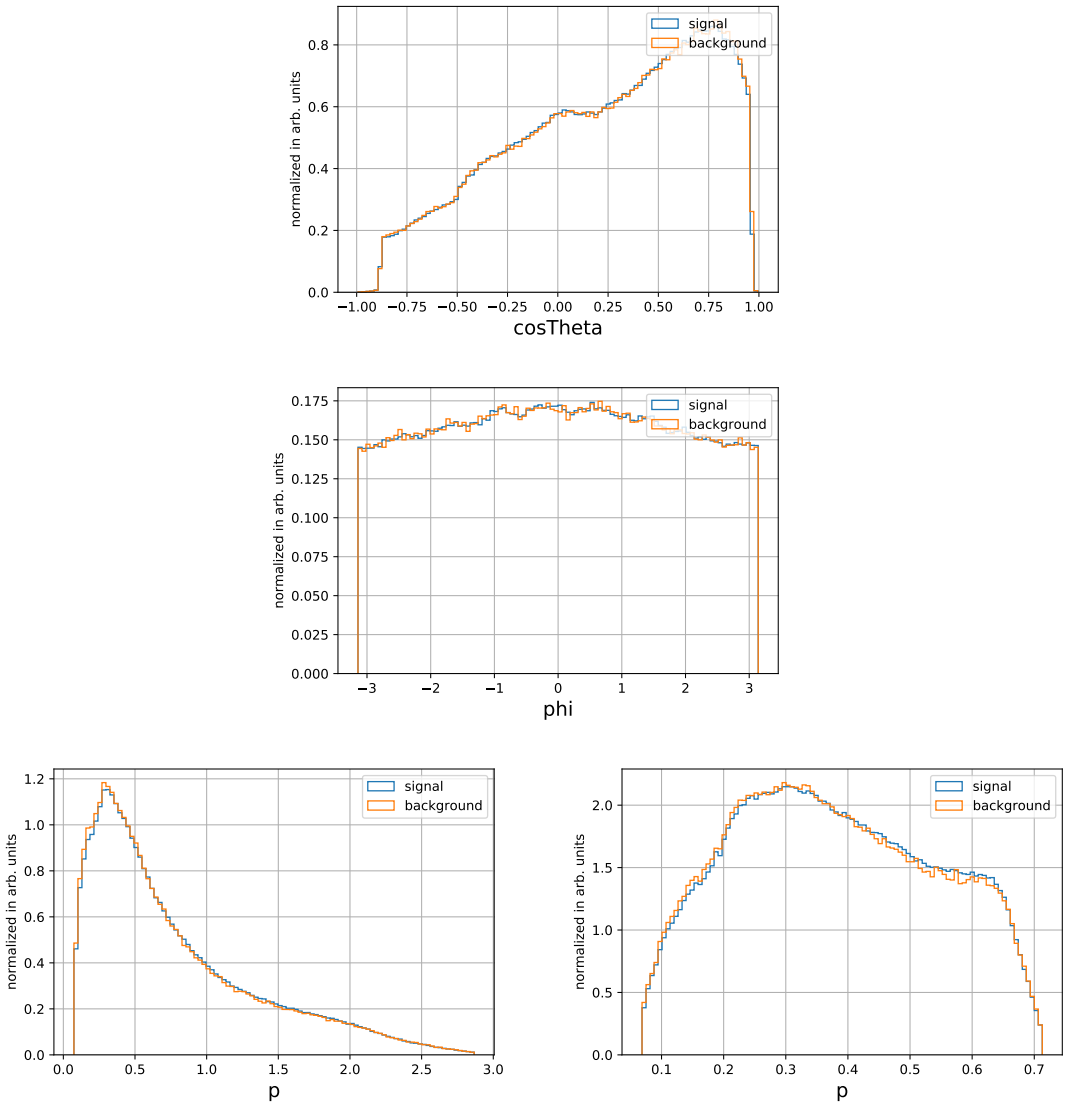


Figure A.16.: Normalized and trimmed distributions of track features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Centered features were not transformed.

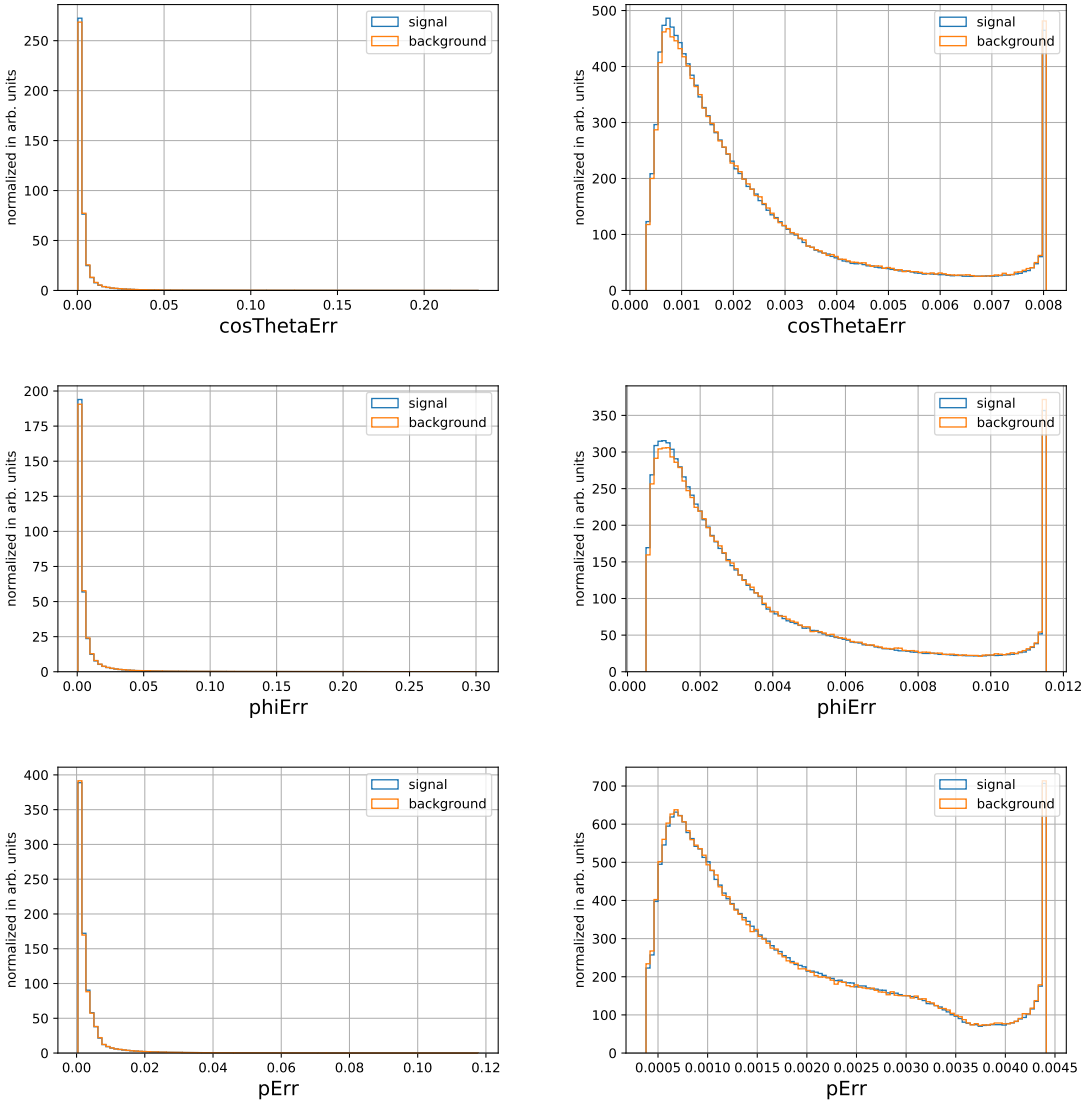


Figure A.17.: Normalized and trimmed distributions of track features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Not transformer features are shown in the middle.

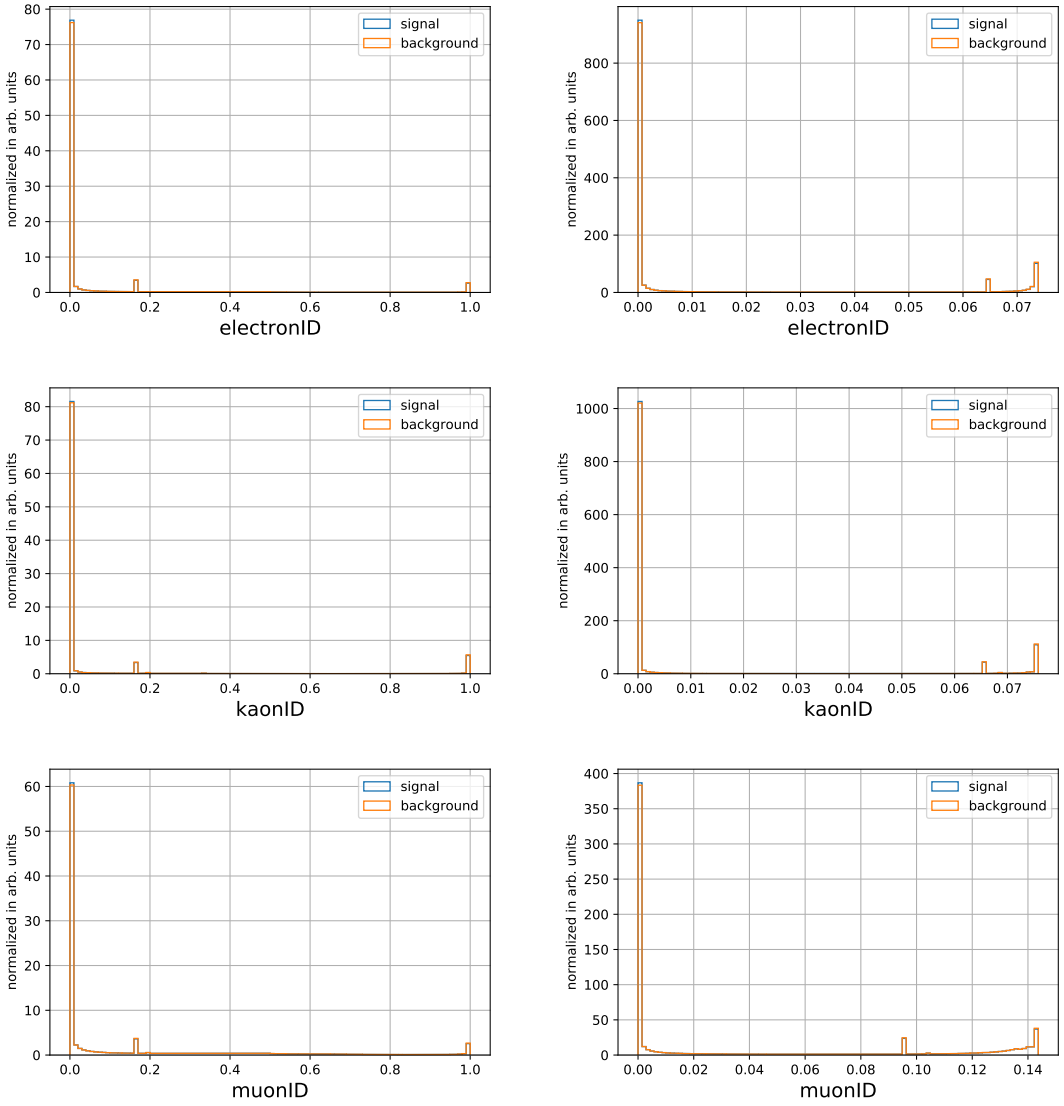


Figure A.18.: Normalized and trimmed distributions of track features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Not transformer features are shown in the middle.

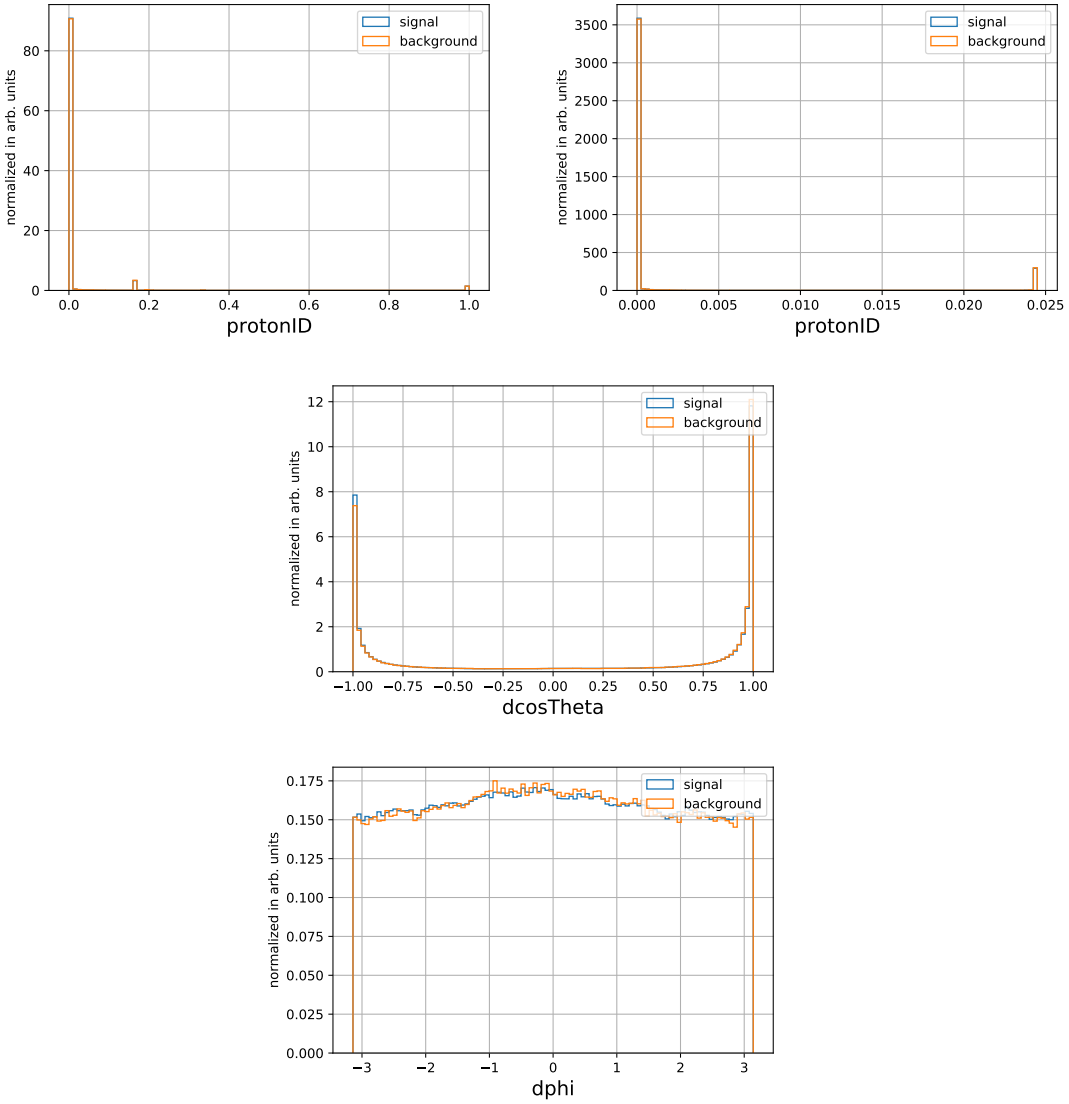


Figure A.19.: Normalized and trimmed distributions of track features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Centered features were not transformed.

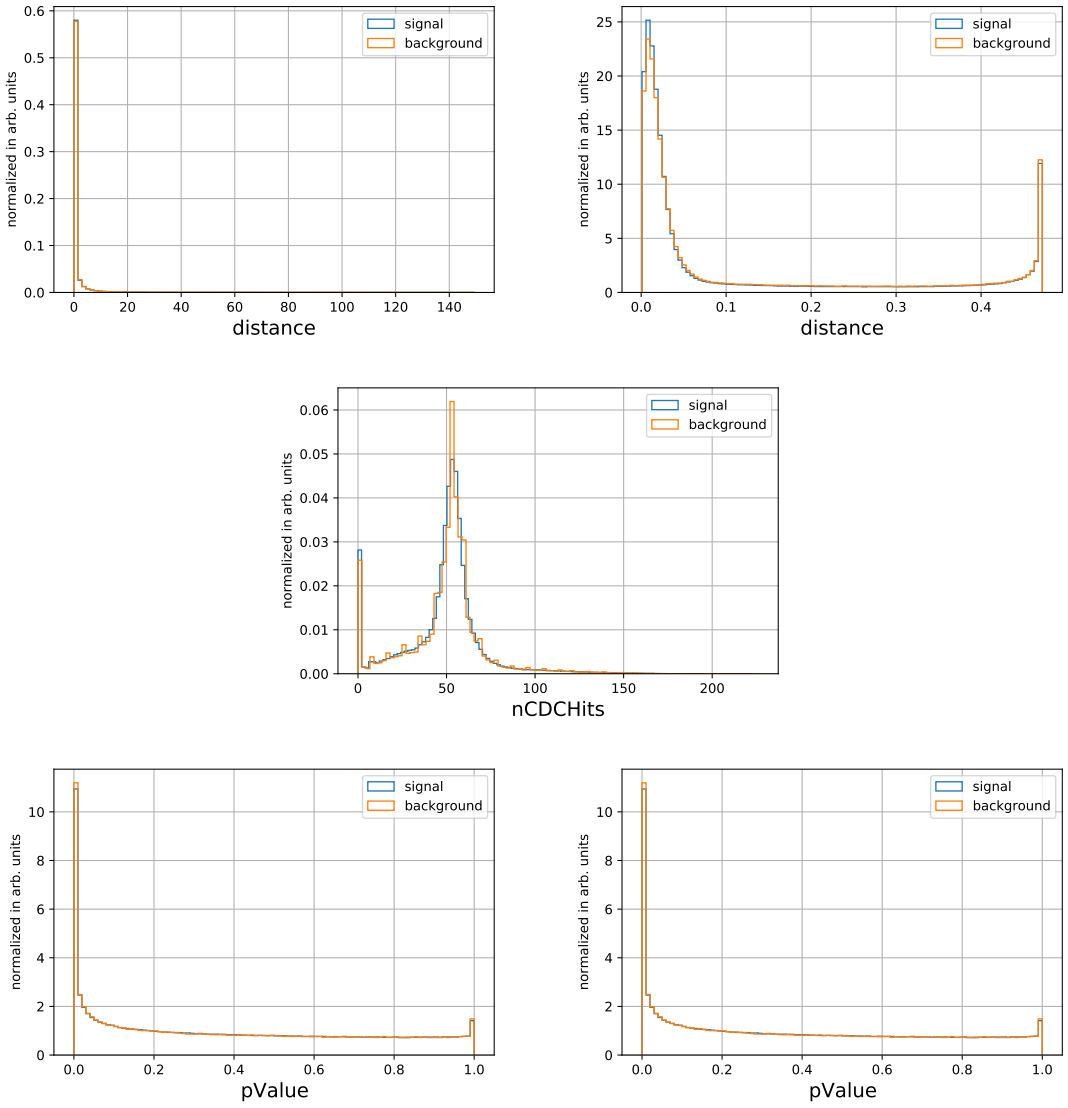


Figure A.20.: Normalized and trimmed distributions of track features used in the Continuum Suppression (see Section 4.2). The distribution of the features is shown before (left) and after (right) the Yeo-Johnson transformation (see Section 6.2). Centered features were not transformed.