Karlsruher Institut für Technologie

IEKP-Bachelor-KA/2014-13

A Study of Multivariate Techniques for Continuum Suppression

Matthias Büchner

Bachelor Thesis

Department of Physics
Karlsruhe Institute of Technology (KIT)

Referee: Priv. Doz. Dr. Thomas Kuhr
Co-Referee: Pablo Goldenzweig (PhD)

Institute for Experimental Nuclear Physics

September 2014

Karlsruher Institut für Technologie

IEKP-Bachelor-KA/2014-13

Eine Studie von Multivariaten Methoden zur Kontinuumsunterdrückung

Matthias Büchner

Bachelorarbeit

Fakultät für Physik
Karlsruher Institut für Technologie (KIT)

Referent: Priv. Doz. Dr. Thomas Kuhr
Korreferent: Pablo Goldenzweig (PhD)

Institut für Experimentelle Kernphysik

September 2014

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.

Unterschrift: ..... Ort, Datum: .....

Im Gedenken an Georg Büchner

Table of Contents

1 | Introduction

The B factories like the KEKB accelerator in Japan, which houses the *Belle* experiment, focus on the investigation of B mesons. They consist of a $\bar{b}$ quark and either a u, d, c or s quark or their respective antiparticles. B mesons are of special interest to the particle physics community as their mean lifetime of 1.6×10^{-12} s is relatively long as compared to other heavy mesons. Due to this longevity the chance of measuring for example time dependent CP violation in an experimental setup increases. Besides, B mesons are quite heavy and thus feature many rare decay products that are also studied at *Belle*.
At B factories, B mesons are produced in collisions of an electron e^- and its antiparticle e^+:

$$e^+e^- \rightarrow \Upsilon(4S) \rightarrow B\bar{B}, \tag{1.1}$$

where the $\Upsilon(4S)$ resonance denotes an excited state consisting of a $b\bar{b}$ quark pair.

While B mesons have been studied since the 1980s to learn about, e.g., CP violation, the general idea of symmetry breaking in weak interaction physics has already been around since the 1950s. Until then physics was thought to be invariant under discrete transformations, e.g., of charge ($C \rightarrow -C$), parity ($\vec{P} \rightarrow -\vec{P}$) or time ($t \rightarrow -t$).
The first experimental evidence for symmetry breaking in weak interactions emerged in the τ-θ-puzzle. In the 1940s two mesons named τ and θ were known. These particles decay into two different states with different parity. Back then the conservation of parity was assumed to hold and consequently the initial states should be of different parity and thus distinguishable . However, in 1956 Lee and Yang published their results on these decays which presented τ and θ to be exactly the same particle as lifetime and mass were obviously the same [LY56]. Explaining this result required the breaking of parity in weak interactions.
In 1964, Cronin and Fitch discovered CP violation in the kaon system (pair of u or d and s quark) [Chr+64]. An attempt to explain CP violation in the kaon system was put forward by Kobayashi and Maskawa in 1973 [KM73] continuing the studies of Cabbibo [Cab63] (known as CKM theory). In 2008, they received the Nobel Prize for theorizing symmetry breaking which predicts at least three generations of quarks.
In 2001, the *Belle* experiment was first to observe CP violation outside the kaon system and thereby confirmed the CKM theory of flavour mixing. Today, at KEKB the *Belle II* experiment is being prepared and built to perform further high precision measurements. These could help to understand the insufficiency of CP violation in explaining the baryon asymmetry in our universe that is linked to the imbalance of matter and anti-matter.

Continuum Suppression

This thesis focuses on the procedure of *continuum suppression*, that will be outlined briefly now. A more detailed explaination will be given in Chapter 2.
Simultaneously to the production of $e^+e^- \to \Upsilon(4S)$ in interest, the alternative production of light quark pairs ($e^+e^- \to q\bar{q}$ $(q \in \{u, d, s, c\})$) is observed. The latter process is often referred to as *continuum*. The cross section of $q\bar{q}$ production is approximately three times lager than that of $\Upsilon(4S)$ production. As a consequence, the separation of background events, which belong to the continuum, from signal events ($e^+e^- \to B\bar{B}$) is necessary to perform any analysis focusing on B mesons. Thus, a multivariate classification is applied to separate signal events from background events.

In this thesis a multivariate tool named NeuroBayes is examined. Among other tools it is used for continuum suppression at the *Belle* experiment. NeuroBayes is a sophisticated implementation of a neural network that can perform classification tasks. Although this thesis focuses on NeuroBayes, many results will be generalizable and can be adapted to other neural network packages.
NeuroBayes is steered by a large number of settings that will be presented and studied in this thesis. Especially, this thesis will give advice on setting up a NeuroBayes network for continuum suppression and save the user time working out the optimal settings. Furthermore, this thesis will compare different layouts of neural networks (staged vs. unstaged) and give advice on when to use each of them.
The description of the network output distribution, especially with analytic functions, proves to be difficult due to peaks and cliffs that often appear in it. Thus, this thesis will especially elaborate on network settings that generate smooth output distributions.

In the following chapter, the need for continuum suppression is motivated and characteristic variables used in this thesis are introduced. The third chapter describes the multivariate techniques applied to suppress continuum with neural networks and gives an overview of the NeuroBayes package. Chapter four contains the results of the study of various networks settings. Here NeuroBayes parameters as well as network layouts employed for training are discussed. Furthermore, the reader will be presented with a guide to setting up a NeuroBayes network to perform continuum suppression. At last, the outputs of different network layouts will be compared by fitting the individual distributions obtained.

2 Continuum Suppression in B Physics

This chapter will motivate the need for continuum suppression. First of all, the necessity of continuum suppression is discussed and followed by an introduction to the variables used in continuum suppression.
In many *Belle* group analyses neural networks have been utilized for continuum suppression. Their utilization is described at the end of this chapter.

2.1 Event Shapes at B Factories

At B factories which operate on the $\Upsilon(4S)$ resonance two processes compete. In an e^+e^- collision either the $\Upsilon(4S)$ can be produced which decays almost exclusively into B mesons, or a pair of quarks ($q\bar{q}$ $(q \in \{u, d, s, c\})$) can be produced. Only events of the first type are in the focus of the B factories. In fact, the cross section of $q\bar{q}$ events, which we refer to as background, is three times larger than the one of the process $e^+e^- \to \Upsilon(4S)$ under consideration. So the vast majority of events are continuum background. The quarks produced in the e^+e^--annihilation can decay into lighter hadrons and thus create a large background. The large amount of continuum background is illustrated in Figure 2.1.
At *Belle* a dedicated off-resonance data-taking slightly below the $\Upsilon(4S)$-peak at approximately 60 MeV center-of-mass energy is performed in order to analyize background more closely [Bev+14].

To separate between signal and background events the shape of every event in the detector has to be analysed.
Continuum background events originating from light quark pairs are generated in a back-to-back way in the center-of-mass frame. Consequently, their kinetic energy almost matches the accelerator energy. The hadrons produced in the related fragmentation possess only a small transverse momentum. This leads to a spatially confined, jet-like structure.
On the contrary, the $B\bar{B}$ events, which are referred to as signal, have a spherical shape. Due to the mass of the $B\bar{B}$ pair almost corresponding to the beam energy, the pseudoscalar B mesons are generated almost at rest in the center-of-momentum frame. Their decay products feature a spin 0 which results in an isotropic distribution. An illustration of the different event shapes can be found in figure 2.2.

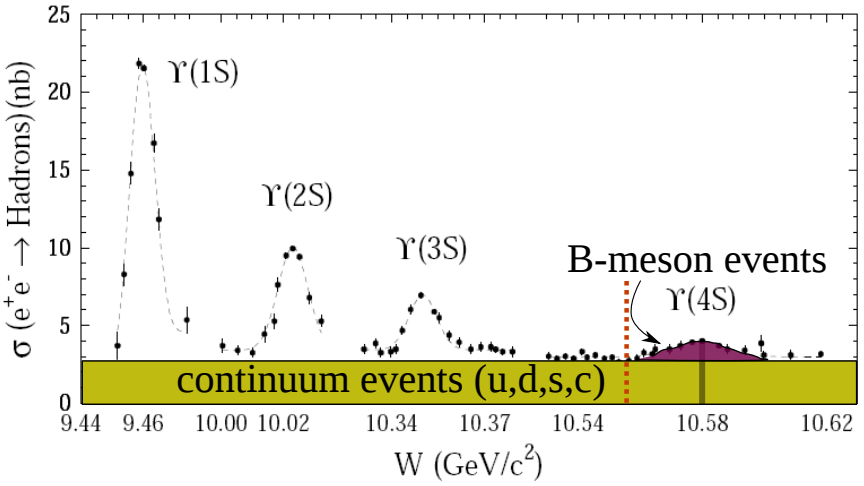


Figure 2.1: Illustration of e^+e^- to hadrons cross section in the region of $\Upsilon(1S)$ to $\Upsilon(4S)$. The continuum (light green) and $\Upsilon(4S)$ resonance (purple) are highlighted. The orange dotted line marks the production threshold for B mesons. Taken from [Neu11], based on [BS93].

2.2 Characteristic Variables

To describe the different types of events, discriminating variables measuring for example sphericity or kinematic distributions in an event are introduced. It can already be concluded that in $B\bar{B}$ events the angular distributions of the participating B mesons are uncorrelated with respect to the momentum direction while the decay products in continuum events have a noticeable correlation.
Information on the phase-space distribution of decay particles is obtained in a number of different ways. The continuum suppression process is therefore based on two stages. First of all, a kinematic selection is performed. Afterwards, a further background rejection is ensured by exploiting differences in the angular distributions of continuum and signal events. The variables and techniques presented are in reference to the ones used at *Belle*.

2.2.1 Thrust

A collection of N particle momenta $\vec{p}_i$ $(i = 1, ..., N)$ is obtained from the detector output. The thrust axis $\vec{T}$ is defined as the axis that maximizes the following expression.

$$T = \max(\vec{T}) = \max \frac{\sum_i \left|\vec{p}_i \cdot \vec{T}\right|}{\sum_i \left|\vec{p}_i\right|}. \tag{2.1}$$

The concept of thrust was established in the 1980s and originally introduced to quantify jet-energies in high energy physics.
However, thrust T is commonly not used on its own but rather the related variable $|\cos\theta_{\text{Thrust}}|$, where θ_{Thrust} denotes the angle between the thrust axis of the momenta of the B candidate

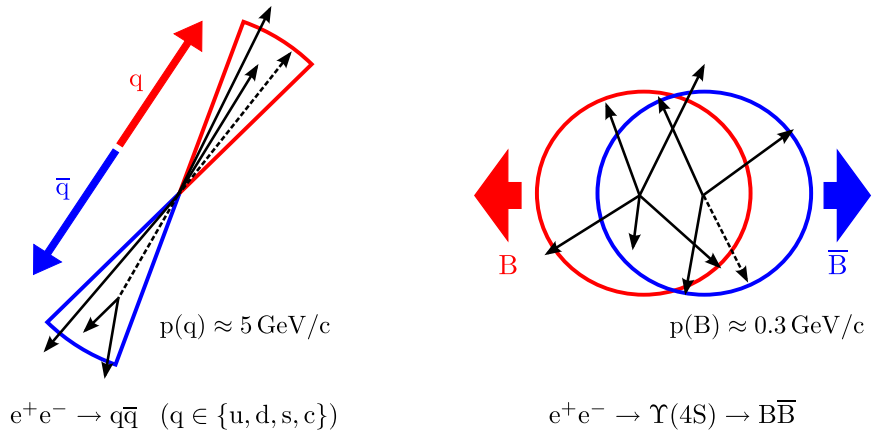


Figure 2.2: Illustration of event shapes. In continuum events (left) light quark pairs are produced back-to-back resulting in a jet-like event shape. $B\bar{B}$ events (right) exhibit a spherical shape due to their decay products being spin 0 particles. Taken from [RÏ2].

and the thrust axis of the remaining particles in the event (referred to as "rest-of-event", or ROE).
In $B\bar{B}$ events $|\cos\theta_{\text{Thrust}}|$ is expected to exhibit a uniform distribution as the B candidate is produced almost at rest in the $\Upsilon(4S)$ rest frame and thus their decay particles are isotropically distributed. In contrast to that, in $q\bar{q}$ events the particle momenta are aligned with the direction of the jets in the event and the ROE and B candidate are collimated and strongly directional. Consequently, the distribution of $|\cos\theta_{\text{Thrust}}|$ peaks at large values near 1.

2.2.2 Fox-Wolfram Moments

Fox-Wolfram moments were introduced in 1978 to yield another useful parameterization of phase-space momentum and energy flow. The k-th Fox-Wolfram moments are defined as

$$H_k = \sum_{ij} \frac{\left|\vec{p}_i\right| \left|\vec{p}_j P_k(cos\theta_{ij})\right|}{E_{vis}^2} \tag{2.2}$$

where P_k is the k-th Legendre polynomial, θ_{ij} denotes the opening angle between the i-th and j-th particle, $\vec{p}_i$ and $\vec{p}_j$ is the momentum of the i-th and j-th final state particle and E_{vis} denotes the sum of measured energy in an event. The normalized Fox-Wolfram moment R_k is defined as

$$R_k = \frac{H_k}{H_0}. \tag{2.3}$$

In continuum events with two jets the modified Fox-Wolfram moments R_k show values close to zero (one) if k is odd (even). Spherical $B\bar{B}$ events tend to show rather different values.
As a refinement of the Fox-Wolfram moments the *Belle* collaboration introduced another generation of moments, often referred to as Super-Fox-Wolfram moments and an even further development referred to as Kakuno-Super-Fox-Wolfram moments (KSFW). There are 17 different KSFW which are grouped into R_k^{so}, R_k^{oo} and R_k^{ss} [Bev+14]. The sum in equation

2.2 is either run over the final-state particles of the reconstructed B candidate denoted by superscript "s" or over the remaining final-state particles in the events denoted by superscript "o".
The moments R_k^{so} and R_k^{oo} are used to discriminate between continuum and background events, while R_k^{ss} is excluded due to correlations with other variables. The R_k^{so} are constructed in such a way that the missing momentum of an event is treated as an additional particle. Accordingly, the moment is composed of three parts: charged particle part, neutral particle part and missing particle part. Precise definitions of the different moments can be found at [Bev+14, p.114f].

2.2.3 B-Meson Flight Direction

In case a spin-1 $\Upsilon(4S)$ decays into two spin 0 B mesons the distribution of the polar angle θ_B between the reconstructed momentum of the B candidate in the $\Upsilon(4S)$ reference-frame and the beam axis follows $sin^2\theta_B = 1 - cos^2\theta_B$. In random combinations of tracks from continuum events the expected distribution of θ_B is uniform. Thus, the variable $|cos\theta_B|$ allows to discriminate between signal B events and B candidates from continuum background. Figure 2.3 features a typical distribution of $cos\theta_B$ used in the examination of this thesis.

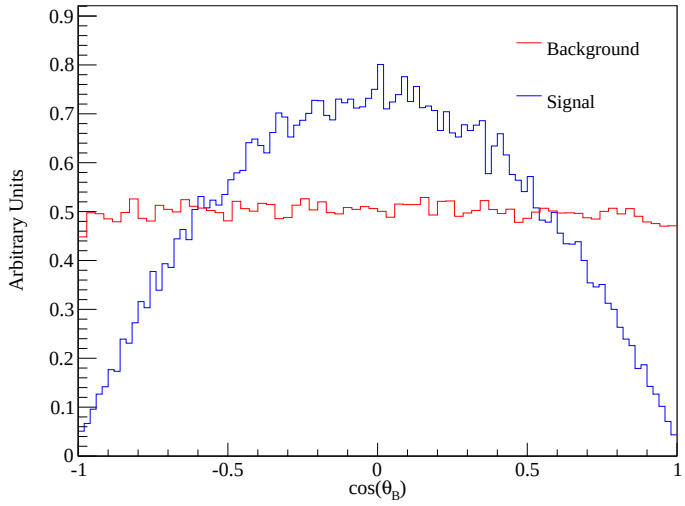


Figure 2.3: Example of distribution of $cos(\theta_B)$ used in this thesis with background (red) following an almost uniform distribution and signal (blue) following a $1 - cos^2(\theta_B)$ distribution.

2.2.4 CLEO Cones

CLEO cones have been introduced by the CLEO Collaboration in 1996 as an addtional set of discriminant variables [Asn+96]. The cones are distributed in steps of 10° and measure the momentum flow into concentric areas around the thrust axis of a reconstructed B candidate. Figure 2.4 illustrates the concept of CLEO cones.

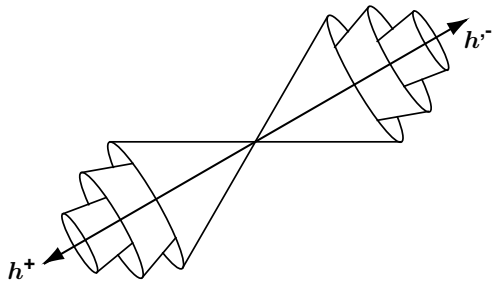


Figure 2.4: Concept of CLEO cones. h^+ and h'^- denote the momenta of two charged hadronic tracks in a $B^0 \to h^+h'^-$ event candidate. The momenta of the ROE particles in each cone are summed to construct the discriminant variable. Taken from [Bev+14, p.111].

2.3 Continuum Suppression with Neural Networks

Having introduced a set of discriminating variables, techniques to realize the continuum suppression have to be found. The most widely used approach for continuum suppression at B factories combines event shape variables such as KSFW or CLEO cones in one discriminating variable, often in a Fisher discriminant attaining values from 0 (for continuum events) to 1 (for $B\bar{B}$ events). This approach and a comparison to neural networks can be found at [Pri13, p.43].
However, some of the latest analyses made use of neural networks (NN) for classification [RÏ2, p.44]. One of the main advantages of neural networks is their capability to not only handle linear but also non-linear complex correlations between input variables. The mentioned Fisher discriminant can only take into account linear correlations among the input variables.

3 Multivariate Classification Techniques

In this chapter the fundamentals of the multivariate classification techniques used in this thesis are explained. Multivariate classification refers to determining the class a multidimensional tuple belongs to. For example, recognizing letters in handwriting is a classification task. From knowing certain parameters of a letter (symmetry, proportions, etc.) the letter can be classified.
The first part of the chapter gives an overview of *data preprocessing*, as this is an important prerequisite to any analysis and utilized in the NeuroBayes package. Secondly, the general concept of neural networks and training algorithms are explained.
Thirdly, the functioning of the NeuroBayes package as a special application of neural networks is explained in further detail.

3.1 Data Preprocessing

An important part of any multivariate analysis, for example in a neural network, is the preparation of the data set. In this regard, *data quality* is an important aspect. Data quality is defined by the qualification of a given data set for the intended use. Factors contributing to data quality include accuracy, completeness, consistency and interpretability. In any real-world application incomplete or inconsistent data sets occur frequently, for example due to missing input or discrepancies in category labels. Data Preprocessing tries to counteract the negative consequences entailed by the mentioned factors.
The most important components of data preprocessing are:

- **Data cleaning:** "Cleaning" data by filling in missing values or smoothing noisy data, removing outliers and resolving inconsistencies. As the quality of the input to any multivariate analysis technique directly corresponds to the output results, this step is of high importance. Many techniques are not robust if fed with missing data and thus will produce unreliable output. However, neural networks have proven to be quite robust in handling missing inputs.

- **Data reduction:** The aim of data reduction is to limit the data set to a (modified) subset representing the original data set. Data reduction is essential to reduce computing time or to make a data set useable at all. Important techniques in this area include dimensionality reduction and numerosity reduction.

Dimensionality reduction aims to transform a data set to a subset of lower dimension. An important technique in this context is *principial component analysis*. Numerosity reduction replaces the data with fewer representative values. Parametric methods assign parameters to a data set which are stored instead of the whole data set itself. Nonparametric methods store reduced representations of the data, e.g. histograms or clusters.

- **Data transformation:** In data transformation, the data are transformed to be more suitable for a data mining algorithm. Typical procedures include smoothing data to remove noise or normalization. Normalizing aims to give all attributes an equal weight. In the context of neural networks it is often favourable to scale the data to a small range, e.g. from 0.0 to 1.0. Normalized inputs will speed up the learning phase of a network significantly.

3.2 Neural Networks

The field of neural networks (NN) was initially launched by psychologists and neurobiologists in search for a computational analog of neurons. By today, this discipline is rather led by computer and data scientists.
A neural network is a set of connected input and output nodes. Each connection is asigned a weight, which is adjusted during the process of *network training*. Network training is performed using a set of random training data which are fed to the network and learned by it.

3.2.1 Multilayer Feed-Forward Neural Networks

Multilayer feed-forward neural networks consist of an input layer, a number of hidden layers, and an output layer. Within each layer an empirically determined number of *units* (also referred to as *nodes*) are located. The number of nodes of the input layer usually corresponds to the dimension of the n-tuples fed to the network. Sometimes a slightly higher number of nodes is used to address the so-called bias that will be explained later. Because the number of hidden layer nodes in a neural network can assume any integer value greater than zero, neural networks can theoretically approximate any function. However, the more complex a network the potentially higher becomes the chance of overtraining.
A model is said to be overfitted (or overtrained in case of NN) if it does not describe the underlying relations in the data, but rather random statistical fluctuations. Opposed to overtraining, a model not capable of describing a set of data due to lack of sophistication may be referred to as underfitted. More detailed explanations for the overtraining /-fitting phenomenon can be found at [Bis06, Chp.1].
The inputs to the network pass through the neurons of the network and are weighted in every network step, depending on the arbitrary number of hidden layers. In this context, the term feed-forward refers to the flow of information in the network topology, which propagates information in a forward direction from input to output layer. Eventually, the weighted outputs of the hidden layer are used as input to the output layer. The output layer returns a value or a set of values according to a nonlinear activation function determining the classifi-

cation. An example of a multilayer feed-forward NN is illustrated in Figure 3.1.
The nomenclature of neural networks denotes a network consisting of an input layer, one hidden layer and an output layer as a *two-layer* NN. The input layer is not counted in the number of layers as it only passes the input tuples to the hidden layers. If all nodes in succeeding layers are connected, a network is referred to as *fully connected*.

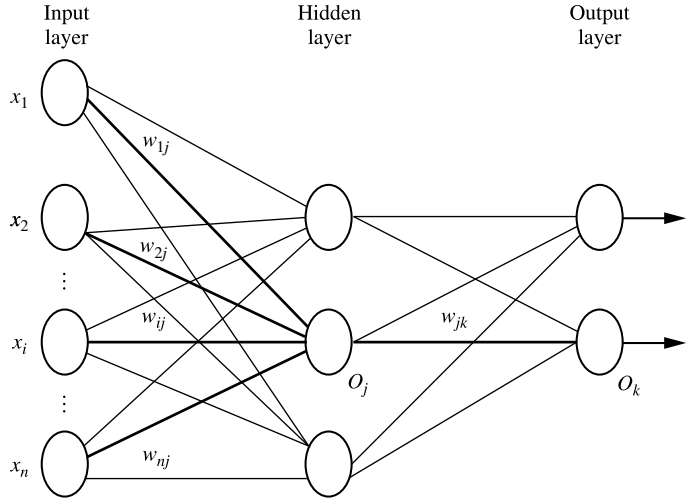


Figure 3.1: Illustration of a multilayer feed-forward neural network. Inputs are denoted by x_i ($i = 1, ..., n$), connection weights by w_{ij} and outputs by O_i. Taken from [HKP12, p.399]

.

3.2.2 Backpropagation

Backpropagation is a widely used regime to train a network using a set of training tuples. With backpropagation applied tuples are processed iteratively through the feed-forward network and the corresponding network output, its prediction of classification, is compared to the *known target* of each tuple. By the comparison of predicted value to target value the adjustment of the network weights is ensured in such a way that the prediction error is minimized. In this regard, *error* denotes the deviation of network prediction from the actual target value. The type of error or loss function used has to be determined empirically. Next, an algorithm for minimization of the network error has to be chosen. The minimization algorithm is then run until the connection weights converge. As the direction of weight modification is "backward" by comparing output to target, this type of training is referred to as *backpropagation*.
The training process consists of two repeated steps and is completed by a terminating condition. After initializing the network weights with small randomly chosen numbers, e.g. ranging from –1.0 to 1.0 the following steps are performed repeatedly for each tuple:

1. **Propagate training tuple:** After inputting the tuple to the first layer it is passed without change to the second layer. There, the weighted sum of all inputs to a node in the

hidden or output layer is computed depending on the weights along the individual connections. For a given unit j the net input is determined by

$$I_j = \sum_i w_{ij} \cdot O_i + \theta_j, \tag{3.1}$$

where the sum is taken over the i units in the previous layer, w_{ij} denotes the weights of the connections between node i to node j and O_i is the output of node i. A bias θ_j of the unit is added which acts as a threshold to control the activity of the node. The net input is then set into the activation function of the node. Activation functions are normally logistic or sigmoid functions. An example of a nonlinear activation function is

$$O_j = \frac{1}{1 + e^{-I_j}}, \tag{3.2}$$

which maps the input onto the range from 0 to 1. This procedure is performed for all units in hidden layers and finally for the output layer, which determines the network's prediction.

2. **Error Backpropagation:** To find the set of weights which lead to minimal network errors a gradient descent algorithm is applied. The error of a unit j is defined as

$$Err_j = O_j(1 - O_j)(T_j - O_j), \tag{3.3}$$

where O_j is the network output and T_j the target value of a given tuple. Equation 3.3 is derived by taking the derivative of Equation 3.2 and multiplying by the deviation of network output from target value $T_j - O_j$. The errors of units in hidden layers have to be calculated recursively depending on the errors of the following layers. In order to update the weights, a shift Δw_{ij} for each weight ij is determined by

$$\Delta w_{ij} = lErr_j O_i \tag{3.4}$$

and hence the updated weight is given by $w_{ij}' = w_{ij} + \Delta w_{ij}$. In equation 3.4 l denotes the learning rate of the network attaining values between 0.0 and 1.0. The learning rate is a helpful parameter if the backpropagation algorithm with its gradient descent method is stuck in a local minimum. On the one hand a too small learning rate can cause the training to be very slowly and thus inefficient, on the other hand a too large learning rate can involve oscillations between different solutions resulting in inadequate solutions.
Besides, the biases need to be updated during backpropagation. The updated bias θ_j' is given by

$$\theta_j' = \theta_j + lErr_j, \tag{3.5}$$

where l again denotes the learning rate.

Depending on the choice of the user, the weight and bias update can either be performed after every tuple (*case updating*) or a number of tuples has been fed to the network or after the whole set of tuples has been presented to the network (*epoch updating*). In the latter case weight and bias increments Δw_{ij}, $\Delta\theta_i$ are stored in a special variable and used to update the

weights and biases after the specified amount of data has been fed to the network.
To terminate the training a *terminating condition* has to be defined. For example, if the changes in the weights Δw_{ij} become smaller than a specified threshold the training is terminated. A second approach seeks to minimize the percentage of tuples misclassified in a previous training with the same data set.
Depending on the backpropagation algorithm, the number of iterations needed for the network to be terminated can range from few to many. As the computational efficiency in terms of time required to train a network with w weights and K tuples behaves like $\mathcal{O}(w \cdot K)$ [HKP12, p.404], techniques to speed up network training are highly desirable. Thus, some algorithms perform *network pruning* which removes links between units with least contribution to the network if this does not significantly decrease the classification efficiency. While this technique may help to lower computation time, it is even more useful to extract rules from the network and therefore increases the interpretability of the network output.

3.3 The NeuroBayes Package

The NeuroBayes neural network package is designed as a two-layer (input, hidden and output layer) feed-forward neural network that is trained using a backpropagation regime. The network is connected to an automatic preprocessing unit.
NeuroBayes essentially consists of two parts:

- **The NeuroBayes Teacher** which performs the network training based on a set of training data provided by the user. The Teacher also performs the user-requested preprocessing. During the training process the complex relations between the variables are learned while simultaneously evaluating the significance of each network weight and unit. Eventually, connections or units are removed from the network based on their statistical significance (pruning). This ensures that the network topology is kept as simple as possible.

- **The NeuroBayes Expertise** is the output of the NeuroBayes Teacher and effectively a trained neural network which can be applied to unseen data.

NeuroBayes provides an analysis file for each training containing a number of plots about the performance of the network as a whole and the contribution of individual variables to the network.
In the following paragraphs, the core concepts applied in the NeuroBayes package are presented. The parameteres used to steer NeuroBayes are introduced in Chapter 4.2. A detailed reference of NeuroBayes parameters can be found at [Phi12].

Preprocessing

The first step in analysing a multivariate data set in NeuroBayes is called preprocessing. It is automatically performed by the algorithm but determined by user-specific settings. The major goal of the NeuroBayes preprocessing is to find the optimal starting point for the training process, especially a set of initial weights. The preprocessing options available can

either be applied to all input variables (*global preprocessing*) or to individual variables.
The global and individual preprocessing can consist of two steps [Phi12, p.27]. First, input variables are equalized which means transforming an arbitrary input distribution to a flat distribution using a nonlinear transformation function. The possible second step is to transform the flattened distribution to a Gaussian with mean zero and $\sigma = 1$. Then the correlation of the individual variables to the target is calculated. Here certain variables with significance lower than a user specified threshold are eliminated [FK06, p.3].
The global preprocessing is encoded in a sequence of three integers `PRE` $= kij$. The parameter k set the required level of significance for a variable to be taken into the training. The parametter i is used to determine whether, the variables are de-correlated, normalised and rotated before training. With j the type of transformation applied to the variables (no transformation, flattening, Gaussian) can be specified. A detailed description of the interesting options of kij used in this thesis is given in Chapter 4.2.1.
For individual variable preprocessing the user can choose from a range of options for variable transformation. The chosen option is encoded in a preprocessing flag of the form `PreproFlag`=ij. The individual preprocessing flags used in this thesis are explained in Chapter 4.2.2.
A complete description of all available combinations for global and individual preprocessing flags is given in [Phi12, p.27f].

Bayesian Statistics in NeuroBayes

In Bayesian statistics *a priori* knowledge is incorporated. A conditional probability of event B to be observed on the condition that event A has already been observed is defined as $P(B|A)$. The *Bayes' Theorem* states that

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}. \tag{3.6}$$

If A is interpreted as theory and B as data, $P(data|theory)$ is the probability for the evidence $P(data)$ to be described by a theory. $P(theory)$ is called a Bayesian prior as it denominates the probability of a theory to hold in certain circumstances. Thus, Bayesian statistics can be very helpful to reject unphysical predictions.

Regularisation

Another important building block of NeuroBayes are its regularisation schemes. Regularisation seeks to minimize the risk of overtraining and to enhance the generalisability of the network. Therefore, different techniques - depending on user choice - constantly evaluate the statistical relevance of all network connections and entire nodes.
If the significance of a node or connection to the network as a whole falls below certain thresholds, it is removed from the network. This ensures that the minimal network topology is achieved and, consequently, the network is less sensitive to random fluctuations [FK06, p.3].
The different regularisation options the user can define in NeuroBayes are presented in Chapter 4.2.1.

4 Study of Continuum Suppression

This chapter is the core of this thesis and is of interest if a NeuroBayes network shall be set up. It contains many observations of irregularities and cures if possible.
Firstly, the setup of the study is described. The code package which is run for every continuum suppression is illustrated. Secondly, the Monte-Carlo samples used in this study are introduced. Thridly, the output of each training, called analysis, is to be outlined. Fourthly, a qualitative analysis of useful NeuroBayes parameterisations is performed. Special attention is paid to groups of variables and their importance to the network as a whole. Besides causes for irregularities are identified and explained. Lastly, a quantitative analysis of a number of network outputs is performed by fitting the different network outputs. Also, different neural network layouts will be compared.

4.1 Overall Setup of the Study

4.1.1 Structure and Functionality

During the analysis in this chapter, many neural networks will be trained and examined. Thus, it is necessary to set up an efficient envoirnment for the network training and performance evaluation.
On the one hand, networks will be trained using only the variables outlined in Chapter 2. This will be referred to as *unstaged* network training. On the other hand, networks can be trained using the output distributions of previously trained networks as input variables which will be referred to as *staged* network training.
The unstaged networks are trained using distributions generated by reconstructed Monte Carlo data of $B\bar{B}$ and continuum events. A set of signal distributions is trained against a set of background distributions each generated by Monte Carlo simulation. A more detailed description of the utilized training data is given in Chapter 4.1.2.
An example of a staged neural network training is illustrated in Figure 4.1. The network layout shown there can be organized into two stages:

1. **Pre-Stage Networks:** CLEO cones and Kakuno-Super-Fox-Wolfram (KSFW) moments are processed individually and a net is trained for each of the variables.

2. **Final Stage Network:** A final network is fed with the outputs of the individual networks from the previous stage (CLEO-cones, KSFW) and the distributions of $cos(\theta_B)$.

The network layout depicted in Figure 4.1 is according to [RÏ2] and has also been used in [Pri13]. In this study the performance using a staged network layout is compared to an unstaged network which utilizes all variables at once (see Chp. 4.3).
In this thesis all neural networks are realised using the NeuroBayes package introduced in Chapter 3.3. To steer NeuroBayes the source code written by Michael Prim during his PhD studies is used [Pri13]. On the one hand, the source code handles the initialization of the NeuroBayes Teacher with the chosen set of parameters, on the other hand it creates a number of outputs to evaluate the network performance after the training in NeuroBayes has finished. Furthermore, the source code allows to run a NeuroBayes Expertise.
The code structure is illustrated in Figure 4.2. At the heart of the programe is the `Continuum Suppression.cc` file. This file contains the list of networks to be trained. Besides, it is responsible for the output of a number of plots (purity vs. efficiency, purity vs. network output, network output, transformed network output.)

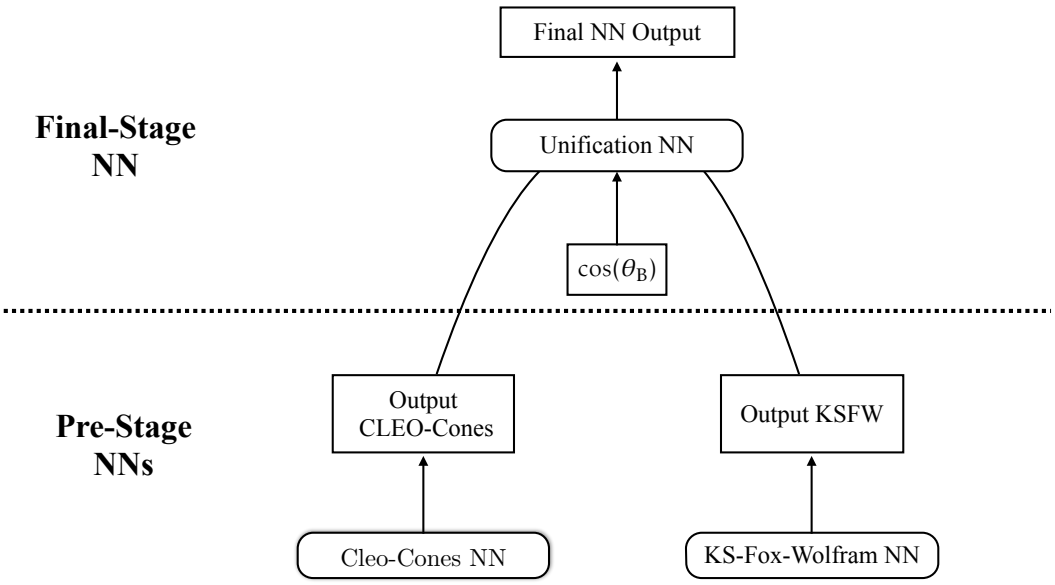


Figure 4.1: Flowchart of a typical neural network layout, used for example in [RÏ2]. The outputs of two individual NN for CLEO-cones and KSFW are combined with $cos(\theta_B)$ and processed in a unifying network. The *Final NN Output* is then used as an expertise to be run on experimental data.

The networks requested by the `ContinuumSuppression.cc` code consist of a list of variables incorporated in the network and the settings used to steer NeuroBayes. The `CSBase Method.cc` contains all relevant functionality of the networks, e.g., running trainings, retrieving values). As this thesis aims at studying a large number of NeuroBayes options and it is essential to set up an efficient workflow, that one the one hand allows to change NeuroBayes settings easily and on the other hand supports the analysis of the neural network classification output. Thus, the source code mentioned are modified and extended. `JSON` files are introduced to handle all the NeuroBayes related settings. The `JSON` files are saved with an individual

timestamp. The library of `JSON` files can later be searched for special setups or tags. Especially, by searching the `JSON` library for special setups, the user can generate own analysis files. Therefore, source code is added to visualise the performance of the trained NeuroBayes networks. The contents of the analysis file specially generated for this thesis are outlined in Chapter 4.1.3. Furthermore, a tool is developed to evaluate the output shape by fitting a particular set of analytic functions.

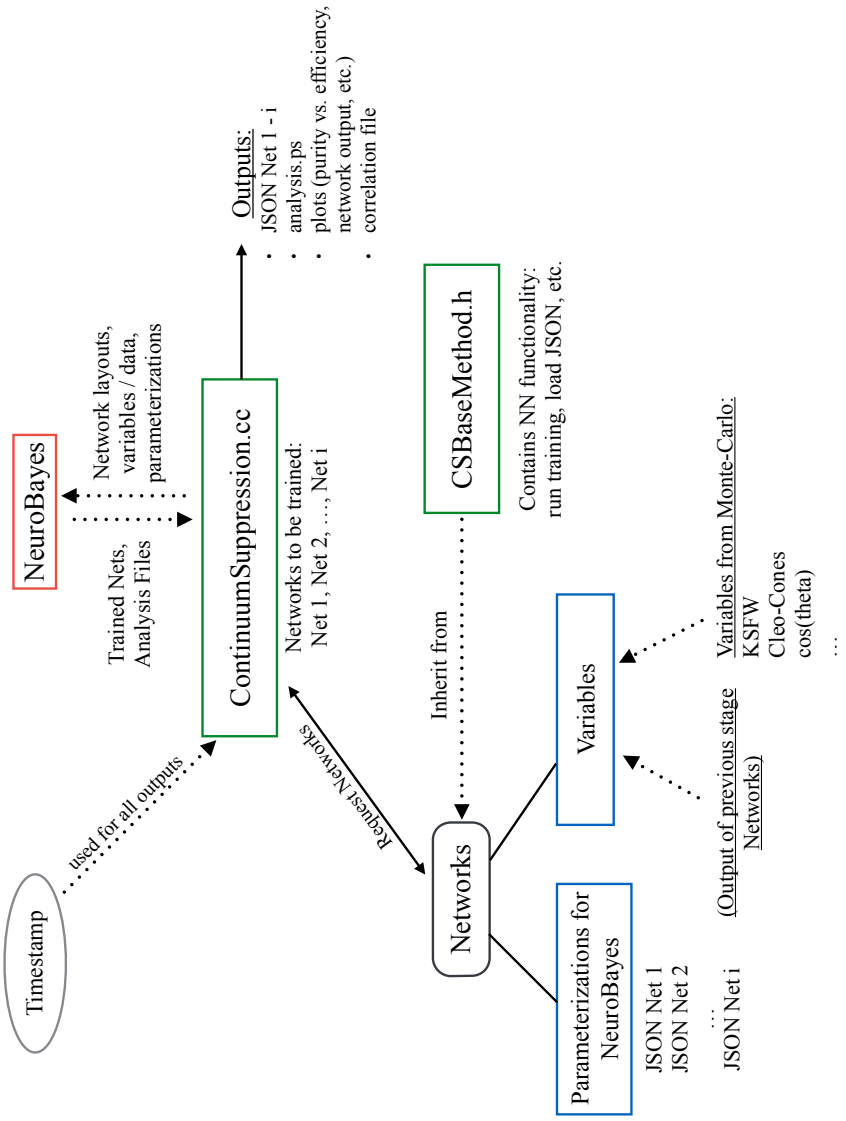


Figure 4.2: Illustration of the programme structure used during the study of continuum suppression. At the heart of the programme the `ContinuumSuppression.cc` handles all steering. Depending on the setup, it requests the networks with their settings and ingredients. All networks inherit their functionality from `CSBaseMethod.h`. A timestamp is used to identify the outputs of each training.

4.1.2 Monte-Carlo Samples of B events

For the examinations in this thesis, two independent Monte-Carlo samples have been used. The first sample is created by reconstructing both B mesons, focusing on B^+B^- events. The one B is reconstructed from the decay

$$B \to \tau\nu. \tag{4.1}$$

Reconstructing this decay becomes possible when the second B decay is reconstructed as the ν cannot be detected. In this sample, the second B is reconstructed from the semi-leptonic decays

$$B^+ \to \bar{D}^0 \ell^+ \nu_l, \tag{4.2}$$
$$B^+ \to \bar{D}^* \ell^+ \nu_l \tag{4.3}$$

($\ell = \{e, \mu\}$) and their charge conjugates. These decays have a relatively large branching ratio (cumulated branching fraction of $\mathcal{B} = (15.29 \pm 0.44)\,\%$) and are called *tag side* decays, while the B decays in Equation 4.1 are referred to as *signal side*.
The second sample reconstructs the B mesons from D mesons and charged pions while focusing only on one of the B mesons. This becomes possible as all particles in interest can be detected in contrast to the ν in Equation 4.1. A dominant decay in this regime is

$$B^0 \to D^- \pi^+ \pi^+ \pi- \tag{4.4}$$

and the respective charge conjugate with a branching fraction of $\mathcal{B} = (6.4 \pm 0.7)\,10^{-3}$. This sample features larger statistics and thus, is used primarily. The first sample described above has been used for confirmatory purposes.

4.1.3 Analysis files

In Subsection 4.1.1 the analysis file and its components were already mentioned briefly. Here, the components and their relationship are described in more detail.
An example of the analysis file is depicted in Figure 4.3. It consists of four plots illustrating the performance of the network and a caption containing information on the chosen settings and tags.
The most important plot is shown in the top left-hand corner. Here the distribution of network output is shown (often only referred to as network output). The frequency of appearances of signal or background events over network output is plotted. Often not the network output itself is taken into account, but a transformed distribution that is expected to exhibit Gaussian like shapes and can therefore be fitted with a sum of Gaussian functions [Pri13]. In the top right-hand side of Figure 4.3 the original distribution has been transformed with

$$n'_{\text{out}} = \log\left(\frac{n_{\text{out}} - n_{\text{cut}}}{1 - n_{\text{out}}}\right), \tag{4.5}$$

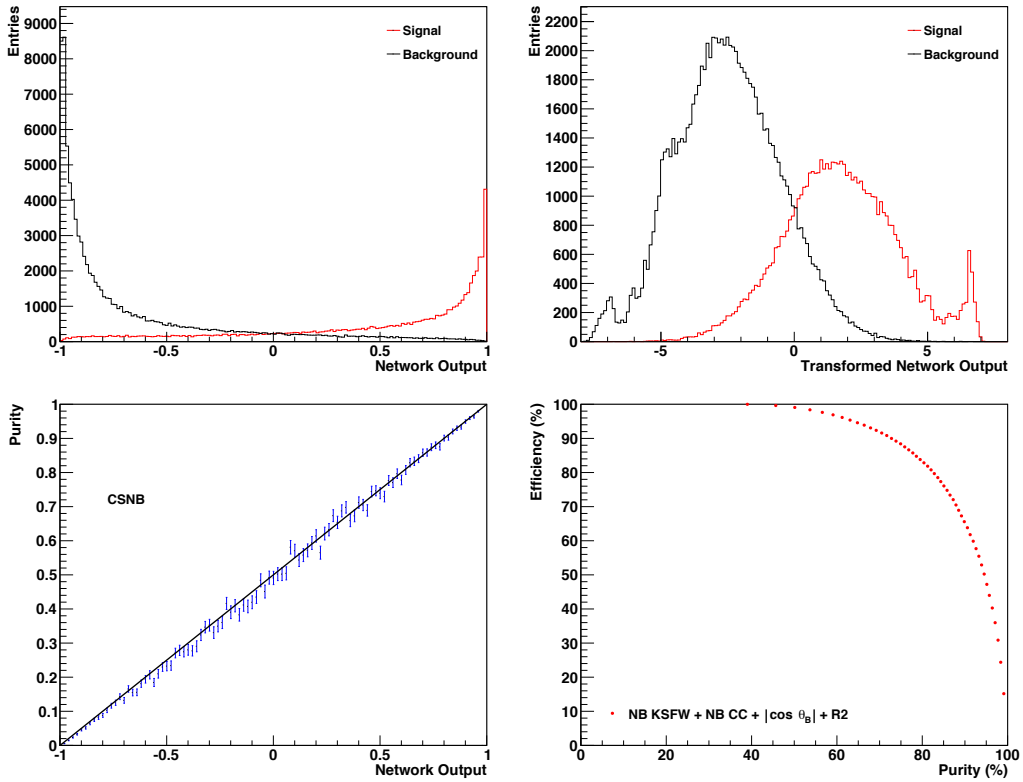


Figure 0.3: Date: 16-32-02, 2014-06-26; "pre": "612", "loss": "ENTROPY", "sample": "2", "rtrain": "1.0", "shape": "DIA", "epoch": "100", "tag": "csnb-set4", "iterations": "300", "learndiag": "1", "reg": "REG", Gini = 53.2, Ginimax = 61.0

Figure 4.3: Example of the analysis file which is used to examine a network training. The essential run parameterisations can be found in the caption. In the top left-hand plot the distribution of network output is shown with background (black) and signal (red) being presented. In the top right-hand plot a transformed distribution is shown. In the bottom left-hand plot the purity over network output is depicted. In the bottom right-hand plot the purity vs. efficiency is shown.

where n_{out} denotes the original and n'_{out} the transformed network output. A cut is introduced by n_{cut} to reject background lower than a certain threshold determined by the value of the cut. The cut can be especially useful if either background or signal vanishes for low or high network outputs. This often leads to peaks, cliffs and other irregularities in the output distribution. However, the user should keep in mind that cutting involves losing information and thus results in a decline in efficiency.

In the bottom left-hand corner of Figure 4.3 the purity vs. network output is calculated. Purity is to be understood as signal purity and describes the ratio of signal to the sum of signal and background in a bin. It is defined as

$$P = \frac{N_{\text{S}}\left(n_{\text{out}} > n_{\text{cut}}\right)}{N_{\text{S}}\left(n_{\text{out}} > n_{\text{cut}}\right) + N_{\text{B}}\left(n_{\text{out}} > n_{\text{cut}}\right)}, \tag{4.6}$$

where $N_{S,B}$ denotes the total amount of singal / background and n_{out} the network output. Thus, $N_S\,(n_{out} > \mathrm{cut})$ is the number of signal events after a cut on the network outputs has been applied.
The error bars in the purity plot are calculated with Gaussian error propagation. The error of entries per bin is estimated by $\sqrt{n}$ where $n \in \mathbb{N}$ is the number of entries in a bin according to Poisson statistics.
The diagonal can be used if the purity increases linearly and thus if a probabilistic interpretation of the network output is possible.
In the bottom right-hand corner of the analysis file the efficiency over purity is plotted. Efficiency is to be understood as the amount of correctly classified signal events. For example an efficiency of 90 % means nine out of ten signal events have been classified as such by the network. Signal efficiency is formally defined as

$$\epsilon = \frac{N_S\,(n_{out} > \mathrm{cut})}{N_S}, \tag{4.7}$$

where the nomenclature is analogous to Equation 4.6.
In the caption of each analysis file the Gini coefficient for signal efficiency vs. overall efficiency is shown. The Gini coefficient can be adduced to quantify the unequal distribution of two observables, for example signal and background. This is illustrated in Figure 4.4. Here, the Gini coefficient can be associated with the ratio

$$\frac{L}{M + L} \tag{4.8}$$

where the nomenclature from Figure 4.4 is used. The figure shows that the Gini increases if the unequal distribution of signal efficiency and overall efficiency increases. Thus, higher values of the Gini coefficient are connected to a higher separation efficiency. If signal efficiency and overall efficiency were equally distributed, this would mean no separation was possible.
The Gini coefficient is often used to break down the quality of a training to just one number. It further enables the user to compare different parameterisations. Nevertheless, a high Gini is not necessarily connected to a flat and smooth shape of the network output which is required for fitting the output. So it is often up to the user to choose between smooth shapes and high degrees of signal / background separation. Besides, the Gini coefficient is only comparable for different trainings on the same sample as its value depends on the individual ratio of signal to background of the sample.

4.2 Qualitative Analysis

In this section the relevant discoveries made while studying the behaviour of NeuroBayes with different parameterisations are presented. More than 600 trainings were performed and analysed. First, to demonstrate the effect of certain global NeuroBayes parameters, an unstaged network is examined. Focusing on an unstaged network reduces the complexity for a start and helps to understand the effect of the parameters tested. Secondly, a staged network is utilized to study individual variable preprocessing. In this part two major causes for

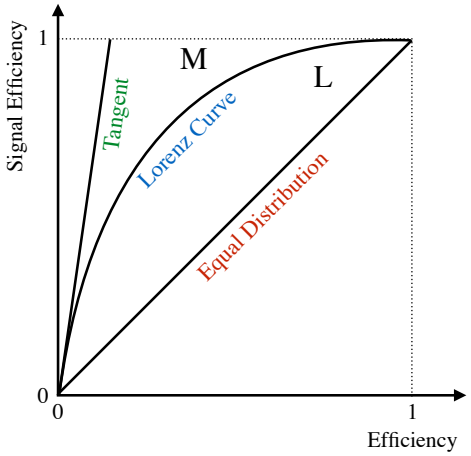


Figure 4.4: Illustration of the Gini coefficient. Depicted is the signal efficiency versus overall efficiency. The area between the equal distribution and the Lorenz curve is denote by L. The area between the tangent to the Lorenz curve at zero and the Lorenz curve is denoted by M. The tangent is determined by the amount of signal in the sample and indicates perfect separation, while the equal distribution indicates no separation.

irregularities will be put forward. Additionally, a guide to setting up an effective NeuroBayes network is presented.

4.2.1 Global Parameter Analysis

Due to the abundance of possible combinations of parameters the thesis only focuses on the most relevant ones. By systematically experimenting with different NeuroBayes options, three global parameters have proven to be of special interest. Apart from the individual preprocessing flags, which are studied later, the parameters `SHAPE`, `REG` and `PRE` have proven their relevance.

Network Setup

The mentioned parameters are studied while the remaining parameters are kept constant. For this part of the study an unstaged network using the whole set of variables is set up. The data are taken from the sample which reconstructs one D meson and charged π mesons. This sample is chosen for its higher statistics.
The number of nodes in the input layer is chosen to be one greater than the number of variables fed to the network to account for the bias node. The number of nodes in the hidden layer is chosen to be two greater than the number of variables. The number of nodes in the hidden layer solely depends on the choice of the user (see Chapter 3.2). The greater the number of nodes the potentionally more complex the model describing the data. However, simultaneously the chance of overtraining rises. Overtraining has been explained briefly in Section 3.3. If the number of nodes is chosen too small, the network does not perform a reliable classification as the model generated will not be sufficient to fit the data.
As the task performed by the network is a binary classification, the output layer consists

of only one node. The output node achieves values ranging from -1 to 1. Values close to -1 denote high probabilities for background events, while values close to 1 denote high probabilities for signal events.
The NeuroBayes parameters are chosen as follows or left at default if not stated otherwise:

- `RTRAIN`: This parameter controlls the amount of input which is used for training. Thus, this option can be used to train the network on a part of a sample, say 80%, while using the remaining part for evaluating the expertise. Especially, this parameter may and will be used to check for signs of overtraining. Here `RTRAIN` was set to 1.0 , which means the whole sample was used as there was no sign of overtraining.

- `EPOCH`: With this parameter the interval of weight update can be adjusted. In this thesis weight updates were performed every 100 samples. `EPOCH` is a useful option if computation time is a critical issue. The choice of `EPOCH` in this thesis has proven to be efficient for both computation time and separation quality.

- `ITER`: The maximum number of iterations, meaning the number of times all training patterns are presented to the network, is set to 300. In most trainings during this thesis only a handful of iterations were actually necessary (~ 20) to minimize the network error and, consequently, end the training process. Thus, a number of maximum 300 iterations is surely sufficient.

- `LOSS`: As a loss function the `ENTROPY` option is chosen, as it allows for a probabilistic interpretation of the results. This is a standard setting.

- `LEARNDIAG`: If not stated otherwise, this is set to 1 in order to force the purity over network output onto the diagonal. If this plot is not distributed along the diagonal, a probabilistic interpretation of the network output is not possible.

The remaining parameters will be varied during the course of investigation. Their setup is outlined where necessary.

REG

With this parameter the user can control the type of regularisation that is performed globally when a network is trained. The aim of regularisation has been outlined in Chapter 3.3. The possible options are:

- `OFF`: No regularisation is performed during the training.

- `REG`: A Bayesian regularisation scheme is applied, dividing weights into three different classes (weight of bias node in input layer, weights of remaining input nodes, weights of connections from hidden to output layer).

- `ARD`: *Automatic Relevance Detection* equips all input nodes with their own regularisation constants independent of each other.

- ASR: *Automatic Shape Regularisation* equips all output nodes with their own regularisation constants independent of each other.

- ALL: Combines ARD and ASR.

To examine the behaviour under variation of the REG parameter, the SHAPE parameter is set to DIA. DIA will later prove to be a good choice if smooth outputs are required. The different choices for SHAPE are compared in the next section.
The effect of different options of REG is relatively small in this study and mostly influences the shapes of the network output. No distinction can be made regarding the Gini coefficient of the related trainings. Besides, all available options are tested for overtraining (using RTRAIN = 0.8), but have not shown such effects.
As the setting REG = "REG" leads to relatively smooth transformed network outputs and is a standard NeuroBayes setting, this study will stick to this option. The transformed network output when using the option REG for regularisation is shown in Figure 4.5. This plot shows noticeable peaks and cliffs which are partially related to the distributions of the individual variables. Addressing this behaviour requires the knowledge of individual preprocessing flags, which are presented in Section 4.2.2. As the network trained with REG = "REG" seems

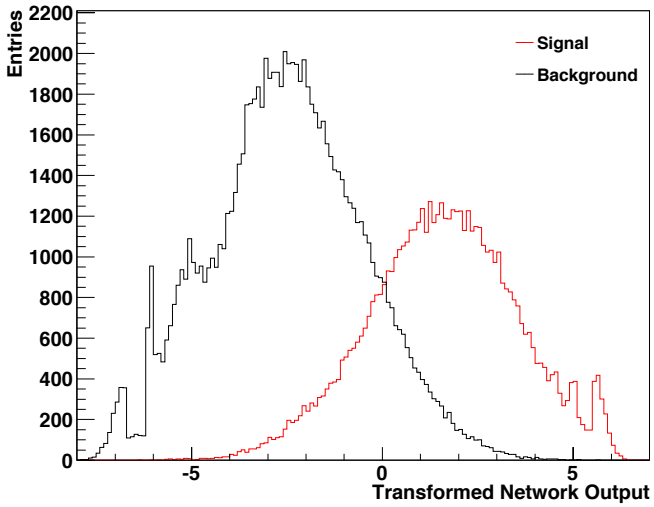


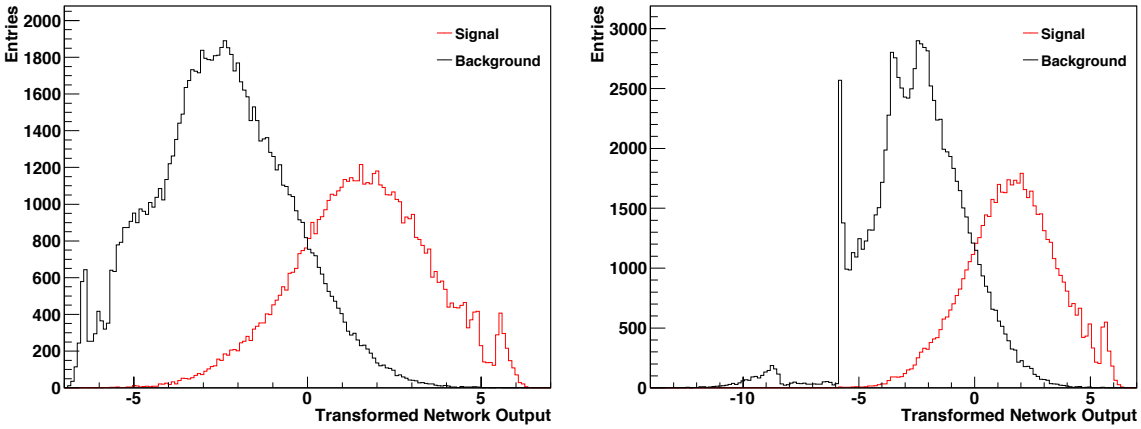
Figure 4.5: Transformed distribution of an unstaged network training using all variables. The peaks and cliffs are partially consequences of the behaviour of individual variables (see Chapter 4.2.2). The configuration is: PRE = 612, SHAPE = "DIA", REG = "REG", cut = −1 and Gini = 52.9 / Ginimax = 61.0.

to be robust, when validated using RTRAIN, all future trainings will be performed using this option. Furthermore, high Gini coefficients compared to all other options for REG are achieved when this option is used.

PRE

The parameter PRE is not only influencing the shape of the outputs, but also has an effect on separation quality measured by the Gini coefficient. This parameter has already been briefly

introduced in Chapter 3.3, but the different options will be explained and examined here. With the parameter PRE the steering of the preprocessing is accomplished. It is encoded in an integer sequence kij. The different integers will be introduced successively.
The first integer k can range from 1 to 9 and determines the required significance of a variable to be considered in the training. Almost all trainings have been performed at $k = 6$, equivalent to 3σ. With the increasing value of k, corresponding to a higher required significance of the variables taken into the network, the separation quality in terms of Gini does not decrease signficantly as one might expect. Therefore, variables pruned away by higher values of k seem to have little to no impact on the Gini.
While the network seems to be robust with respect to the Gini coefficient when k is varied, an impact on the output shape can be noticed. Figure 4.6 depicts the transformed output for a high and a low value of k.
In this study k is chosen to $k = 6$. On the one hand this ensures that the detrimental effect on the output shape is relatively small, on the other hand the chance of overtraining is kept to a minimum as only variables with high signficance are used to train the net.



(a) With $k = 1 \equiv 0.5\sigma$ (b) With $k = 9 \equiv 4.5\sigma$

Figure 4.6: Illustration of transformed network output with different choice of required variable significance in PRE = kij. The parameter k ($k = 1, ..., 9$) sets the required significance for a variable to be taken into the training to $k \cdot \frac{\sigma}{2}$. The Gini coefficient stays the same for both trainings, but omitting variables with lower added significance has a noticeable impact on the output shapes. The remaining parameters are set to: PRE = k12, SHAPE = "DIA", REG = "REG", cut = -1 , RTRAIN= 1.0.

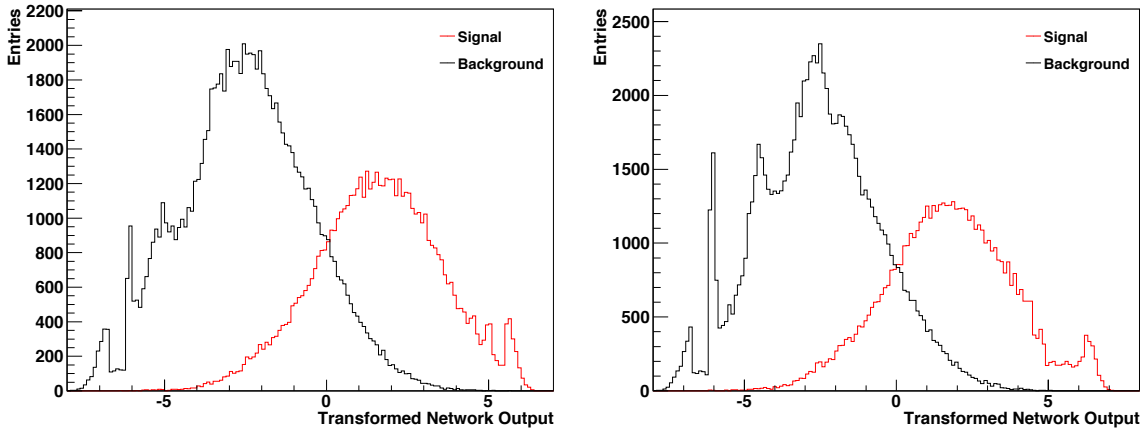
In the encoding kij, the integer i ($i = 0, ..., 3$) determines if the variables are de-correlated, normalised or rotated before training. In the case of classification, $i = 3$ can directly be ruled out as it is not a possible option in this mode. The remaining options for i are:

- $i = 0$: The input variables are fed untouched to the net, no de-correlation is performed.
- $i = 1$: The input variables are de-correlated and normalised prior to the training.
- $i = 2$: The input variables are de-correlated and all linear dependence on target is rotated to the first new input variable (Principal Component Analysis).

The integer j sets the type of transformation applied on the input variables:

- $j = 0$: No preprocessing is performed.
- $j = 1$: The distributions of input variables are flattened.
- $j = 2$: The distributions of input variables are transformed to a Gaussian distribution.

Now, specific combinations of i and j are examined. First, unreasonable choices will be excluded.
Choosing i or j zero often results in output distributions with purities not even closely distributed along the diagonal and thus is not recommended. The user should check the distribution of purity in every training done, as only a distribution along the diagonal permits a probabilistic interpretation. In Appendix C the analysis files for different settings of PRE are presented. Nevertheless, if a probabilistic interpretation is not necessary, the user may still make use of an option which does not lead to a diagonal distribution of the purity.



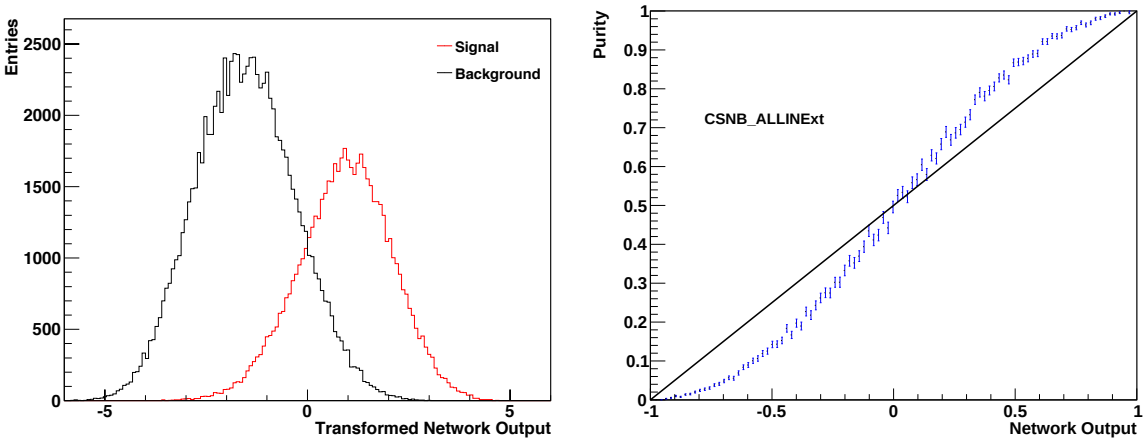
(**a**) With PRE = 612 (**b**) With PRE = 622 (boosted)

Figure 4.7: Illustration of transformed network output for the most commonly chosen settings of PRE. Due to forcing the purity on the diagonal in the purity vs. network output plot, peaks and cliffs can appear in the output. The remaining parameters are set to: SHAPE = "DIA", REG = "REG", cut = −1 , RTRAIN= 1.0.

Having excluded the unreasonable settings for PRE, the findings in this study suggest using either 612, 622 or 611 as with these settings the purity is distributed along the diagonal and the outputs have Gaussian shapes.
If 622 is chosen, a *boosted* training is performed. Boosting refers to performing a second training which uses higher weights for misclassified data and basically is a process of readjusting the connection weights. In the case of boosted trainings the NeuroBayes analysis file contains many plots showing pre- and post-boost results. In Figure 4.7 the network output of a boosted and an unboosted network are compared. For the sample analysed, the effect of boosting is a by about 2% higher Gini coefficient. Although, boosting might increase the Gini, the boosted network could be more likely to show signs of overtraining. However, the sample used in this study does not show a noticeable effect of overtraining. Apart from the

Gini, this sample shows that boosting might reduce the smoothness of the output. In 4.7b this effect can be observed on the left-hand slope of the background distribution.

A very interesting result is gained when training the network with PRE = 621. Figure 4.8 shows the transformed network output and purity vs. network output plot. This training hints, a almost perfect Gaussian shape can be achieved if the purity is not distributed on the diagonal and follows an "s" shape. When the purity is forced further on the diagonal, this can result in peaks, cliffs and other irregularities (see Chapter 4.2.2). The fundamental difference between this setting (621) and the other studied settings 612 and 622 is the type of input transformation. The option 621 flattens the distributions, while 612 and 622 perform a Gaussian transformation of the input.
The use of the option 621 is not recommended since the purity is not distributed along the diagonal. The option 621 will be useful later to demonstrate the cause of certain irregularities in the transformed network output.



(a) Transformed network output

(b) Purity vs. network output

Figure 4.8: Training results with PRE = 621. The sigmoid-like shape of purity 4.8b is connected to a Gaussian distribution in the transformed output 4.8a. The remaining parameters are set to: SHAPE = "DIA", REG = "REG", cut = −1 , RTRAIN= 1.0.

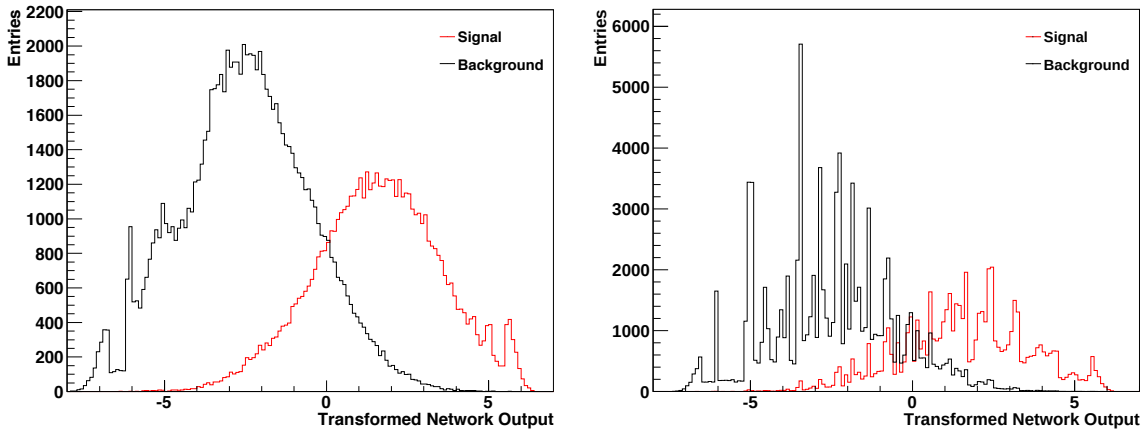
SHAPE

The SHAPE parameter controls the behaviour of direct connections from input to output layer and has a significant impact on the shape of the network outputs. The different options are:

- OFF: No connections between input and output layer are established.
- DIAG: A spline fit is applied to the output of the output nodes. The fit is required to distribute the purity versus network output along the diagonal after preprocessing and before training.
- DIA: This option is a specialization of DIAG. The technique applied matches DIAG with the exception that a smoother spline fit is performed.

- `TOTL`: Establishes direct connections between input and output layer in order to describe a linear density estimation.
- `INCL`: Direct connections between input and output layer are established to describe the inclusive distribution. This option is mainly used for shape-reconstruction and is not used in the investigations for this thesis.
- `MARGINAL`: Utilizes a binomial marginal sum method as input to the network. This setting cannot be used in this thesis as it only works for problems with uncorrelated, univariate data.

The two most commonly chosen options are "`DIA`" and "`DIAG`". Both options apply a spline fit to the result of the output node and ensure that purity versus network output is distributed along the diagonal. The main difference between the two options is the kind of fit applied. From comparing the plot of the mentioned fit for the two possibilites in the NeuroBayes analysis file (Appendix B), the user notices that `DIA` applies a smoother fit to the output node, which results generally in a smoother network output. Both options have almost equal Gini coefficients, while the "`DIAG`" option tends to show slightly better Ginis on the sample studied.
Figure 4.9 shows the distinguishable outputs of trainings performed with the "`DIAG`" as well as the "`DIA`" option. The output using "`DIAG`" contains many wiggles and peaks. Thus, it will not be reasonable to fit the outputs generated using this option. If fitting the outputs is of interest, "`DIA`" certainly proves to be the better choice.

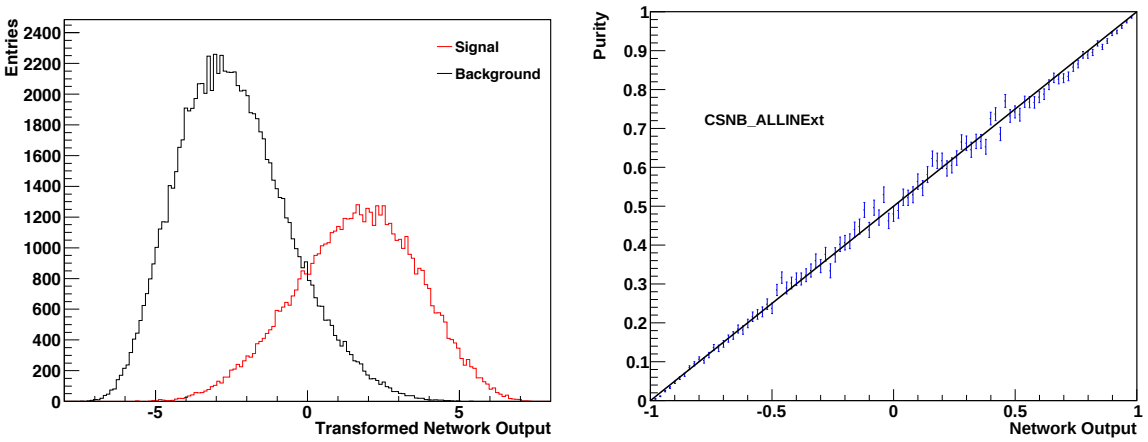


(**a**) With `SHAPE` = "`DIA`" (**b**) With `SHAPE` = "`DIAG`"

Figure 4.9: Illustration of the options "`DIAG`" and "`DIA`" for the `SHAPE` parameter. "`DIAG`" shows a characteristic wiggly and peaky shape, which has been found throughout the study in various combinations and network layouts. The remaining parameters are set to: `PRE` = 612, `REG` = "`REG`", cut = −1 , `RTRAIN`= 1.0.

In addition to the two options for `SHAPE` presented, the study found the options "`TOT`" and "`INCL`" (which is orginally recommended for shape-reconstruction) to perform even smoother fits than "`DIA`" in almost every training. With these options connections between input and output layer are established (detailed explanation [Phi12, p.42]). Their purity is distributed

along the diagonal in the purity vs. network output plot. Therefore, a probabilistic interpretation is possible. Besides, there is no sign of overtraining when using RTRAIN = 0.8 for both options. To sum it up, this study does not find any evidence contradicting the usage of the "TOT" or "INCL" option. Figure 4.10 depicts the transformed network output and purity versus network output for SHAPE = "TOT". Nevertheless, the user should make sure the LEARNDIAG option is on, when using these two options for SHAPE as they do not directly force the purity vs. network output distribution on the diagonal. In general, no signficant difference of separation quality in terms of Gini is found for the different options of SHAPE.



(a) Transformed network output

(b) Purity versus network output

Figure 4.10: Illustration of SHAPE = "TOT". With this setting a smooth, Gaussian output is received and the purity is distributed along the diagonal. The remaining parameters are set to: PRE = 612, REG = "REG", cut = −1 , RTRAIN= 1.0.

4.2.2 Individual Parameter Analysis

In this chapter the consequences of different individual variable preprocessing flags will be highlighted. The use of individual preprocessing flags is necessary if peaks and cliffs still appear in the network output after setting the global preprocessing parameters. Before, however, the contribution of individual variables to the network output will be assessed.

Relevance of Individual Variables

In this part of the individual parameter analysis the unstaged network setup explained in Section 4.2.1 is used. The global parameters are set to: SHAPE = "DIA", PRE = 612 and REG = "REG". The relevance of individual variables is now analised by gradually excluding groups of variables.

There are two main groups of variables this study focuses on: CLEO cones and Fox-Wolfram moments (charged, missing, neutral, etc.). A list of variables can be found in Appendix A.

First, the loss in Gini coefficient when excluding special groups of variables is measured. Removing the whole set of Fox-Wolfram moments (CLEO cones) reduces the Gini coefficient by about 16% (2%) on the sample studied. Broken down to different groups of Fox-Wolfram

moments, the highest decline (∼6%) of the Gini coefficient is noticed if neutral Fox-Wolfram moments (see Appendix A) are excluded. The single variable with the most influence is `r2` which is a special Fox-Wolfram moment.
With regard to CLEO cones the sample shows the higher the order (angle) of a CLEO cone the less the added significance. This means CLEO cones describing small angles (see Figure 2.4) tend to be of higher importance to the net.
Apart from the loss in Gini coefficient, the effect of excluding groups from the network training on the shape of the network output has been studied.
The group of Fox-Wolfram moments with highest importance for the output shape is `k0hooX` (see Appendix A), where $X = 1, ..., 4$. These Fox-Wolfram moments are often denoted by R^{oo} [Bev+14] and are calculated from two ROE particles. Figure 4.11 shows the consequence of excluding this group of variables. The highest peak in the background distribution noticed in Figure 4.11 can be a result of variables containing δ-functions with low purities, that rise in importance if the `k0hooX` Fox-Wolfram moments are excluded. The causes of peaks like this will be discussed in Section 4.2.2.

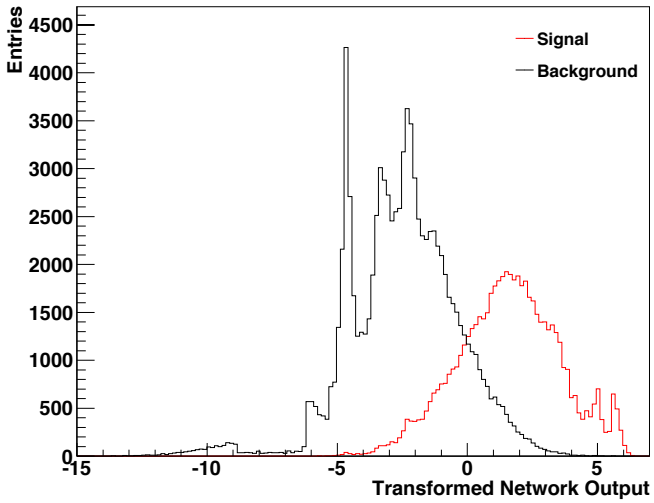


Figure 4.11: Transformed network output if `k0hooX` ($X = 1, ..., 4$) are excluded from training. Excluding this group of variables has the greatest visible consequence for the network shape, while the loss in separation quality in terms of Gini (0.37%) is comparatively small. The configuration is: `PRE` = 612, `SHAPE` = "`DIA`", `REG` = "`REG`", cut = −1. Performance: Gini = 52.7 / Ginimax = 61.0.

Besides, attention should be paid to the visualized correlation matrix of the variables from the NeuroBayes analysis file. It can help to get a quick overview of the relations of the variables. Figure 4.12 shows the correlation matrix taken from the NeuroBayes analysis containing all variables observed in this study. In general, the correlation of Fox-Wolfram moments appears to be noticeably higher than the correlation of CLEO cones.
Simultaneously, the relative proportion of Fox-Wolfram moments among the variables with highest signficance added to the network, has been larger than the proportion of CLEO cones in almost all trainings performed. Nevertheless, no direct connection of this effect to the gen-

erally higher correlations of Fox-Wolfram moments can be established.

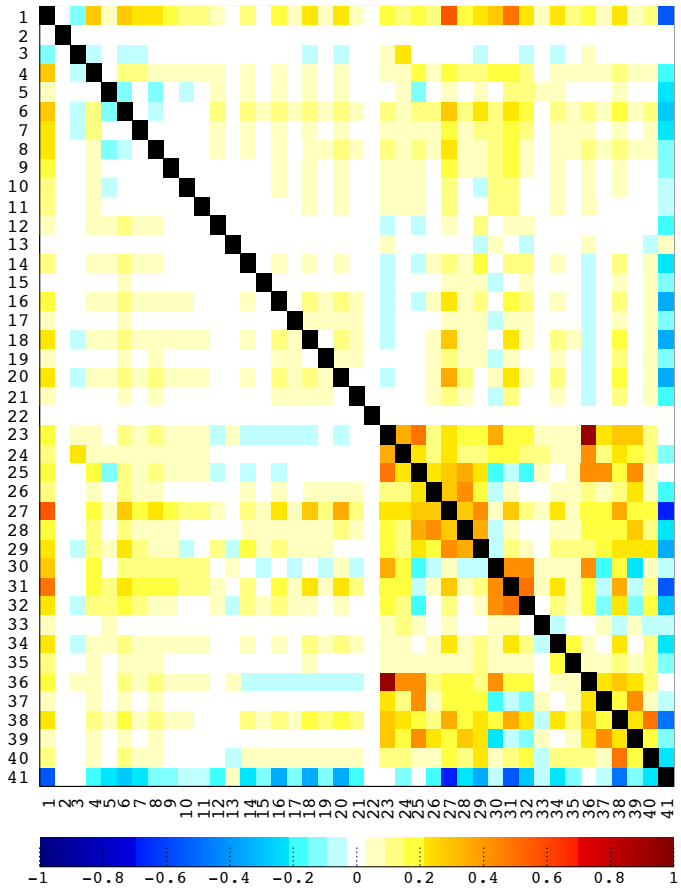


Figure 4.12: Correlation matrix of all input variables used in this study. The variables 2 to 21 refer to the CLEO cones, while 22 to 41 refer to the Fox-Wolfram moments. In the first variable the correlation of a variable to the target is shown. Generally, the correlation of CLEO cones seems to be lower than the correlation of Fox-Wolfram moments. Interestingly, the second (`ct_bb`) and 22nd variable (`cm_costh`) have no or very low correlation to other variables.

All tests presented above have been repeated using `SHAPE` = "`TOT`". With this setting no difference in the output distributions could be noticed if single variables or groups of variables are excluded. Thus, "`TOT`" not only leads to smooth outputs (see Figure 4.10), but is also robust if special groups of variables like Cleo-cones or Fox-Wolfram moments are excluded from the training.

Individual Preprocessing Flags

In contrast to the global preprocessing flags presented before, individual variable preprocessing flags are used to adjust the preprocessing for individual variables.
The individual variable preprocessing flag consists of two integers ij. There is a high number of different possible options for ij, which are listed in [Phi12, p.29f].

The options used in this study are $i = 1, 3, 9$ and $j = 2, 4$. The digit j defines the transformation a variable is subjected to:

- $j = 2$: Transforms variable to a Gaussian distribution.
- $j = 4$: Uses result of regularised fit to mean values of target to transform variable.

The digit i can be used to modify the action defined by j. Especially, it is useful to determine the way NeuroBayes works with missing input. Missing input denoted by the value -999 which has to be set beforehand can be modelled with δ-functions. The different options of i used are:

- $i = 1$: Use mean target and flatten the variables distribution. δ-functions will not be considered.
- $i = 3$: Use mean target and flatten the variables distribution. δ-functions are used and left untouched during transformation.
- $i = 9$: Use mean target and flatten the variables distribution. δ-functions are used and set to zero during the transformation process defined by j, while the remaining distribution is transformed to have mean zero and unit width.

Studying individual variable preprocessing flags works best with a staged network like the one illustrated in Figure 4.1 as the user gains an additional level of control as compared to a network utilizing all variables simultaneously.
In general, the settings applied in final stage networks (see Figure 4.1 are more influential on the overall output distribution than settings in pre-stage networks. Thus, it is very important to find a beneficial set of individual preprocessing flags for the unification network shown in Figure 4.1. Nevertheless, the user has to adjust settings of lower-stage networks if individual variables are believed to have a negative impact on the output distribution.
The individual networks are set up like the network in Section 4.2.1 if not stated otherwise. Only the individual variable preprocessing flags are varied in this part of the analysis. The "flagged" CLEO cones (see Appendix A) are ignored as they only add few significance to the network.

Causes of Irregularities To set up the individual preprocessing flags it has been useful to first search for regularly appearing irregularities like cliffs and peaks in the network outputs and to research their causes. During the examination of various network outputs two main irregularities have been established:

- Constraining the purity plot to the diagonal when fitting the output can lead to irregularities in the tails of the distributions of signal and background. From Figure 4.8 it can be deduced that forcing purity onto the diagonal can have an effect on the signal at high and background at low network outputs. Especially, if the transformed network output shows symmetric effects, this can be a consequence of constraining the purity to the

diagonal. This problem cannot be adressed by adjusting individual preprocessing flags, but by changing global parameters. Figure 4.13 illustrates a symmetric irregularity.

- If the user decides to model missing input (denoted by value -999) with δ-functions, this can lead to sharp peaks in the output. Peaks are encouraged, if the δ-functions have relatively high or low purity. Low purites in δ-peaks may be influential on the distribution of the background and vice versa. In this case it is useful to first exclude the variable with highest or lowest purity in the δ-function. If the irregularity (peak) is reduced, the user can focus on adjusting the individual preprocessing flag of this variable or decide to eliminate the variable completely. This procedure should be repeated if other individual variables are believed to cause irregularities. Nevertheless, the user should always keep in mind that smooth outputs are not necessarily connected to high separation quality.

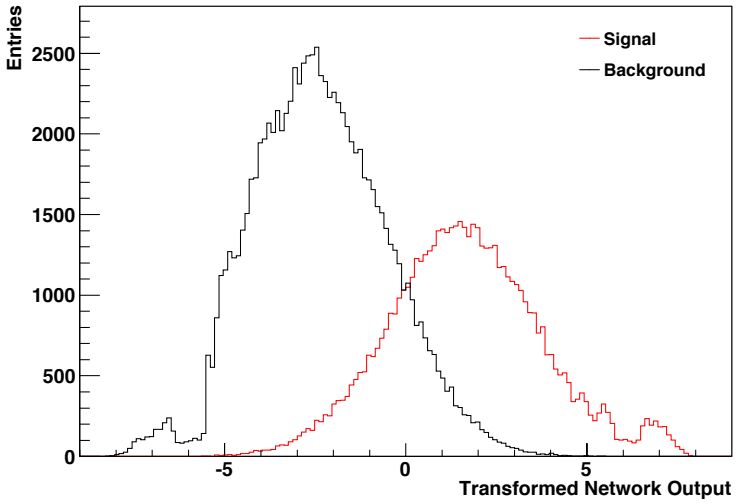


Figure 4.13: Transformed network output of a staged network showing a symmetric irregularity at the left slope of background and right slope of signal. Symmetric irregularities cannot be adressed with individual preprocessing flags as they are caused by the global preprocessing. The configuration is: `PRE` = 612, `SHAPE` = "`DIA`", `REG` = "`REG`", cut = −1. Performance: Gini = 52.5 / Ginimax = 61.0.

Setting up Individual Preprocessing Flags Based on the two major causes of irregularities several recommendations can be made. First of all, the network output will generally be smoother if the inputs are smooth. Thus, it proves to be beneficial if a variable can either be transformed to a Gaussian shape (e.g. flags $ij = 12$ & $ij = 92$) or if the purity distribution of the variable using the fit to the mean values of target transformation (e.g. flags $ij = 34$ & $ij = 94$) is relatively smooth (see Figure 4.14) and shows no peaks or wiggles at especially high or low purities. Figure 4.15 illustrates the two different Gaussian transformations of the network input ($ij = 12$ & $ij = 92$) for the variable `k0hso03`. In many instances peaks in the network output are reduced when variables containing single peaks at especially low or high network inputs, like shown in Figure 4.15a, are either excluded from the training or the individual preprocessing flag ij is adjusted. Figure 4.15b shows an example of a variable's

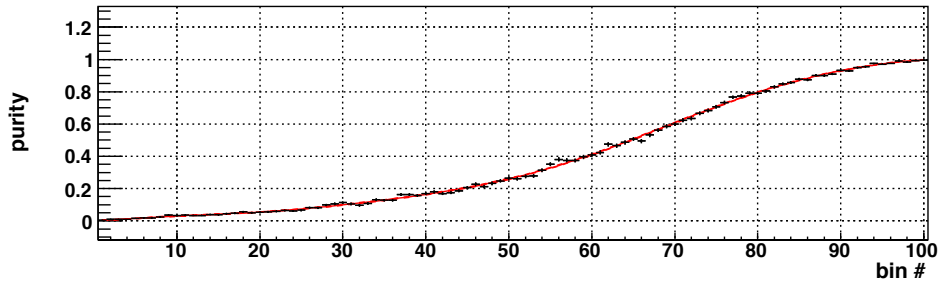
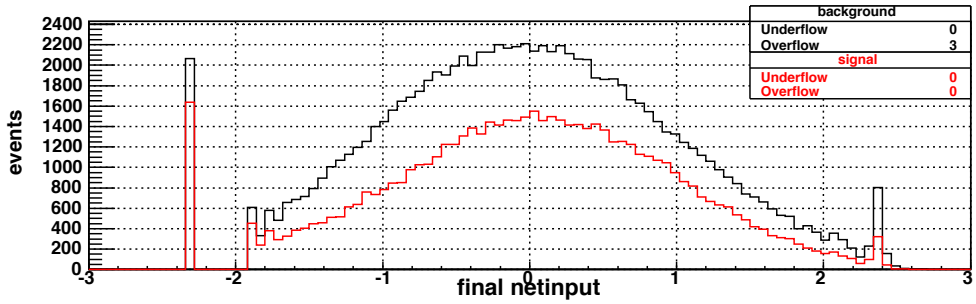
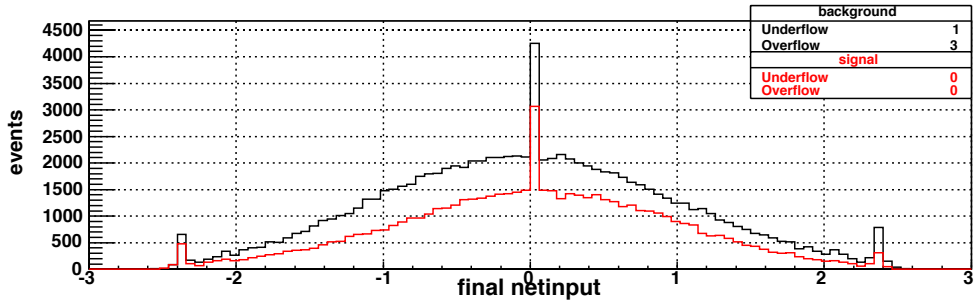


Figure 4.14: Example of the purity versus bin plot from the NeuroBayes analysis file wit $ij = 34$.

network input with beneficial effect for the overall network output. If peaks appear within in the Gaussian distribution and not outside like in Figure 4.15a, the overall network output has proven to be smoother. If a variable does not contain any δ-functions there is no difference between $ij = 12$ and $ij = 92$.



(**a**) With Individual Preprocessing Flag $ij = 12$



(**b**) With Individual Preprocessing Flag $ij = 92$

Figure 4.15: Illustration of different treatment of missing input for the variable `k0hso03`. The study suggests peaks in tails of network input distributions can have a noticeable impact on the network output which then shows peaks or cliffs itself. The peak to the left of the Gaussian in Figure 4.15a is an example for preprocessing with a negative consequence for the overall network, whereas the bottom figure illustrates a better network input although peaks appear within in the Gaussian distribution.

Ideal network setup on sample studied When examining different options for the individual preprocessing flags, the user should start at the highest level network and proceed to lower level networks like motivated above.
Here, in the unification network (see Figure 4.1) four variables are processed. Based on the general remarks made above, the study found the following preprocessing flags to be beneficial:

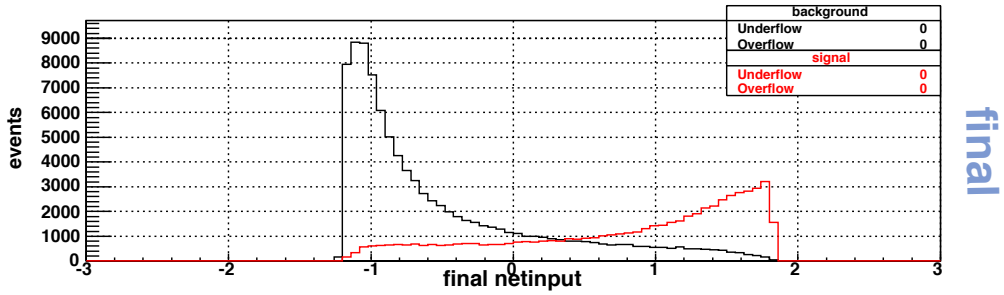
- `CSNBKSFW`: 34
- `CSNBCC`: 34
- `r2`: 12
- `abs(cm_costh)`: 12

The Gaussian transformation (12) works very well on `r2` and `abs(cm_costh)`. For the Fox-Wolfram moments and CLEO cones the option $ij = 34$ has proven to be best. The use of the option $ij = 94$ would make no difference as there are no missing inputs in these variables. Figures 4.16 illustrate the network inputs of the four variables to the unification network with the settings introduced above.

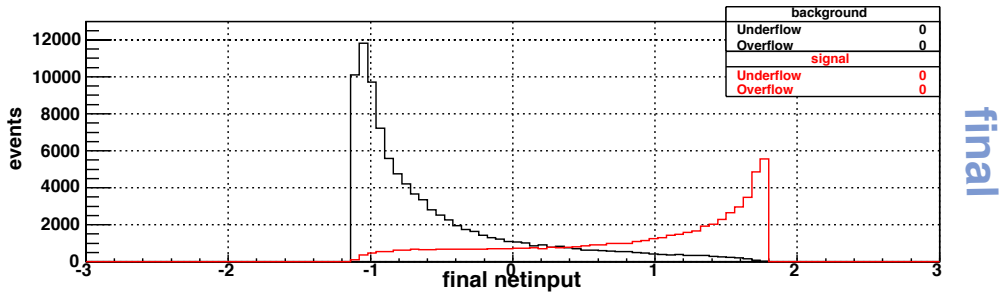
If a beneficial setting for the unification network has been found, the user can proceeded with the lower stage networks. Apart from the variable `ct_bb` (see Appendix A) the CLEO cones are not suited for Gaussian transformation and should rather be processed with the preprocessing flag $ij = 94$ (if missing input exists) or $ij = 34$. The variable `ct_bb` has a very good Gaussian shape after transformation with $ij = 92$. However, the impact on the overall network output is unfavourable and thus, it should be processed like the remaining CLEO cones. The Fox-Wolfram moments should be processed with $ij = 94$. With this option most of the variables attain a shape like `CSNBKSFW` in Figure 4.16b. With a Gaussian transformation (92) shapes like the one in Figure 4.15a appear and thus, a detrimental effect on the overall network output is noticed.
The network output shape with the settings explained above is depicted in Figure 4.17a. Figure 4.17b depicts the network output with a cut applied at -0.8. Cutting at this value removes the cliff on the left slope of background and thus, yields an almost Gaussian distribution of background.

4.3 Quantitative Analysis

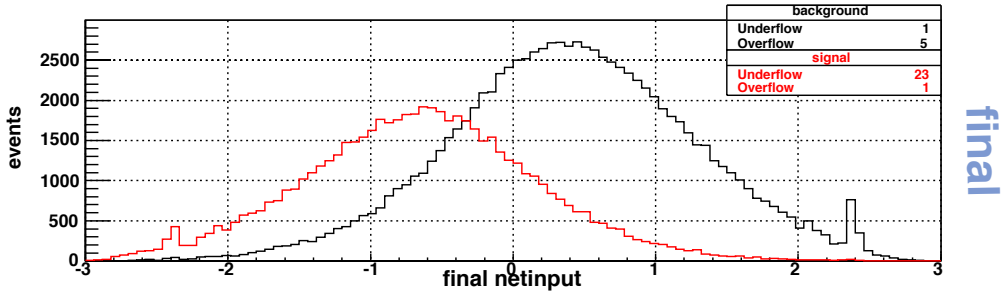
The previous sections have examined qualitative effects of various combinations of network parameterisations. The following section is dedicated to exploring a functional description of the transformed network output. As explained in Chapter 4.1.3 the transformed network output is expected to exhibit a Gaussian shape. Thus, a combination of Gaussian functions will be fitted to the transformed network outputs. The reduced χ^2 of the fits will be used as an indication of goodness of the fit.



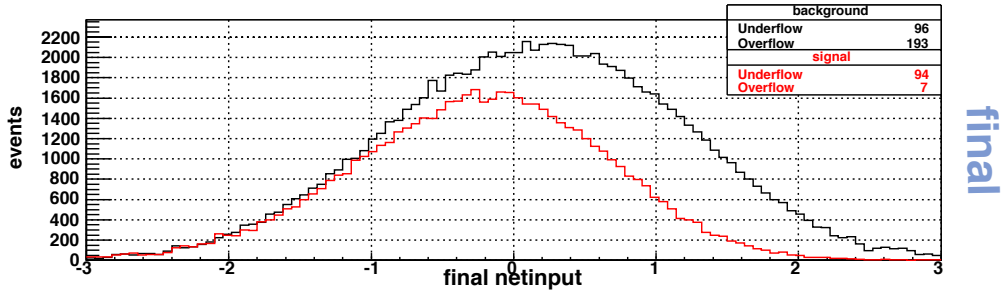
(a) `CSNBCC` with $ij = 34$



(b) `CSNBKSFW` with $ij = 34$



(c) `r2` with $ij = 12$

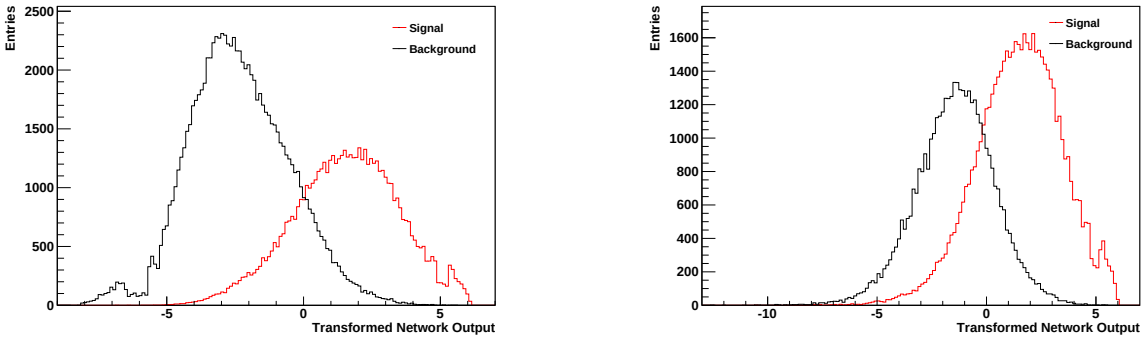


(d) `abs(cm_costh)` with $ij = 12$

Figure 4.16: Illustration of different unification network input variables with optimal individual preprocessing flag using the sample that reconstructs B mesons from D and pions.

Fit Setup

The binned, transformed network output for background and signal is fitted with a combination of a bifurcated Gaussian and an ordinary Gaussian. The fitting procedure is performed in `RooFit` using a maximum likelihood fit. Having achieved a suitable description of back-



(**a**) Cut at -1.

(**b**) Cut at -0.8.

Figure 4.17: Illustration of the final transformed network output with cuts applied at different values of untransformed network output. The settings are: `PRE` = 612, `REG` = "`REG`", `SHAPE` = "`DIA`"

ground and signal each, the superposition of background and signal distribution is fitted with a weighted sum of the estimated density functions. The fit performed has the following structure

$$S = c_S P_S + c_B P_B, \tag{4.9}$$

where P_S and P_B denote the fixed probability density functions (PDF) for signal and background and c_i with $i = \{S, B\}$ are coeffiencts to be determined through the fit. Here, the coefficients are normalised to 1, so the coefficients represent the relative proportion of signal and background. Later, an extended fit will be performed where the coefficients denote the absolute amount of signal and background. For each fit the reduced χ^2 is calculated. The χ^2 test is applied to determine whether a deviation of the data from a given description (fit) occurs by chance. It assumes the errors in the data have a Gaussian distribution. The reduced χ^2 has an expectation value of 1.

Fit Performance on different Network Layouts

The fit described above is performed on four representatively chosen network outputs. The networks have each been set up using the best practices established in the qualitative analysis Chapter 4.2.
On the one hand two settings for the parameter `SHAPE` will be compared ("`DIA`" & "`TOT`"). These two options have proven to entail the smoothest outputs. On the other hand a staged network is compared to an unstaged network. The remaining parameters in all trainings have been set to: `REG` = "`REG`", `PRE` = 612, `RTRAIN` = 1.0 and `LEARNDIAG` = 1. The individual preprocessing flags have been set to values outlined in Section 4.2.2 that have proven to lead to smooth transformed network outputs. All networks are trained using the sample that reconstructs B mesons from a D and pions. The results obtained by fitting the PDFs described in the previous section are summarized in Table 4.1. Although, the reduced χ^2 are quite the overall distributions are well described by the fits. The high χ^2 can be explained by the peaks and cliffs presented in Chapter 4.2, which cannot be described by the set of functions used in this thesis. Nevertheless, the fits describe the overall structure of the distributions

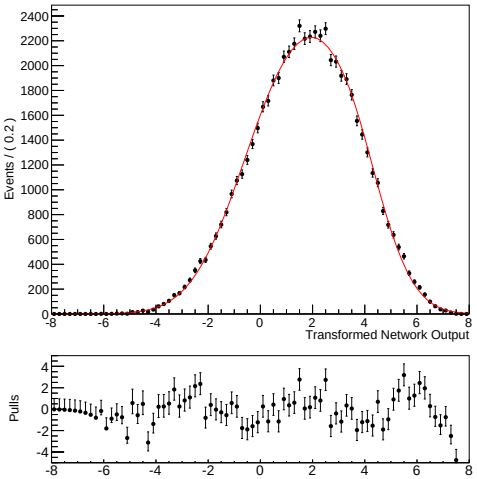
Table 4.1: Results of χ^2 for different network layouts and options of `SHAPE` using the whole sample for fitting signal and background as well as the superposition.

Network Layout	Shape	red. χ^2 Signal	red. χ^2 Background	red. χ^2 Sum
Staged	DIA	31.45	21.53	44.63
Staged	TOT	71.61	71.57	61.62
Unstaged	DIA	17.14	30.96	42.04
Unstaged	TOT	2.27	3.35	3.79

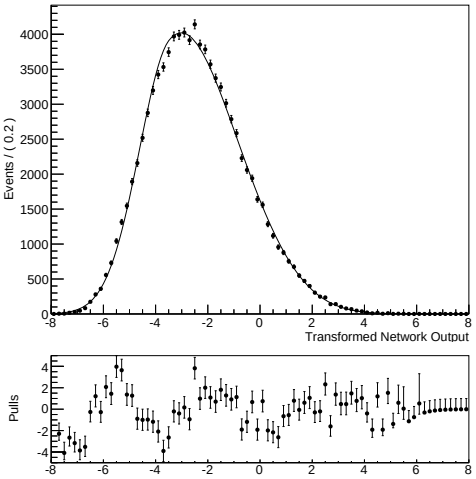
well.
As for these fits, the whole sample was used to fit both signal and background as well as superposition, these fits will have to be verified later to rule out overfitting. However, the χ^2 results can be used as an indicator for the smoothness of different network setups.
By far the best description of the network output is obtained for the unstaged training using `SHAPE` = "`TOT`", while the staged network with `SHAPE` = "`TOT`" shows the worst χ^2. In addition to the remarks concerning unstaged and staged networks in previous chapters, it can now be assumed that unstaged networks not only tend to show smoother outputs but can also be fitted better. The fitted PDFs for the mentioned network are depicted in Figure 4.18.

Extended Fits The indications by the χ^2 for the different network setups are validated by performing extended fits. Therefore, the PDF from Equation 4.9 is adjusted. The coefficients c_i are not normalised to 1, but left to be determined completely by the fit. Thus, their values should resemble the number of entries of signal and background in the sample fitted.
Superposition samples are constructed that consist of different proportions of signal and background. From the fit the coefficients c_i are obtained and compared to the *de facto* amount of signal and background events in the sample tested. If the fit performs well, the estimation of signal and background entries should agree with the actual number of entries within the parameter errors from the fit.
The PDFs P_S and P_B from Equation 4.9 are estimated using the remaining entries after constructing the sample mentioned above.

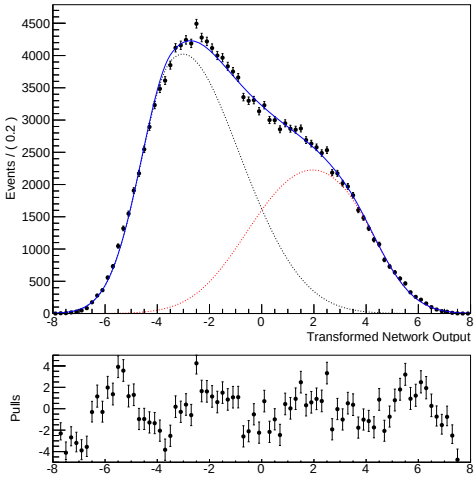
Results The examination shows only the fit to the output of the unstaged network with `SHAPE` = "`TOT`" leads to valid results for the coefficients for various signal ratios and confirms the indication given by the χ^2 above. The number of entries of signal and background in the constructed sample does not deviate significantly from the numbers of entries estimated by the fit. The fits for different signal ratios for the unstaged network with `SHAPE` = "`TOT`" are shown in Figure 4.20. In second place comes the unstaged network with `SHAPE` = "`DIA`" (Appendix D, Figure D.1). Only for signal ratios of 100% the fit estimations for signal and background entries deviated signficantly from the *de facto* number of entries.
The staged networks with `SHAPE` = "`TOT`" and `SHAPE` = "`DIA`" performed worst in this test. Especially, for signal ratios of 30% and 80% the deviations of data from the fit have been signficantly large. Nevertheless, this does not mean those networks are badly trained. Their



(a) Fit of signal distribution.



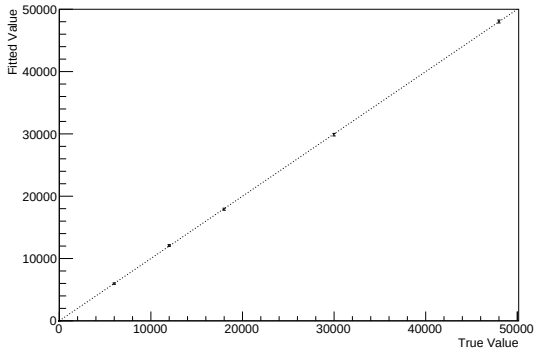
(b) Fit of background distribution.



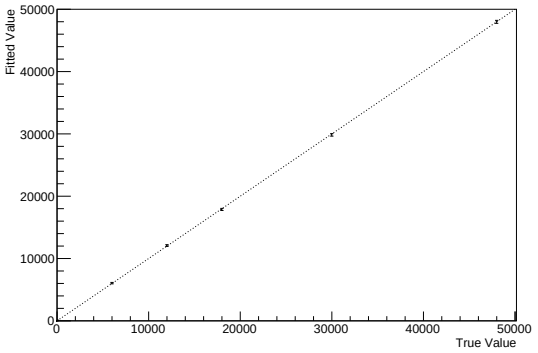
(c) Fit of superposition distrubtion.

Figure 4.18: Illustration of the fitted network output and their respective pull histograms for an unstaged network with `SHAPE` = "`TOT`". Here the whole sample was used to fit signal and background as well as their superposition. The χ^2 results of all fits are summarized in Table 4.1

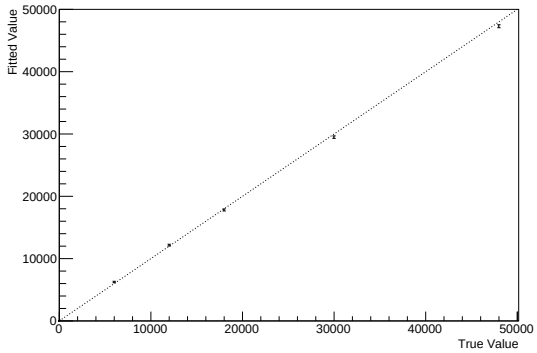
Gini coefficients are as high as the Gini of the best performing unstaged networks or even better. This study, only suggests smooth outputs are more likely to be obtained from unstaged networks. In Figure 4.19 the fitted value of signal entries in the samples constructed is plotted over the true value.
The fits for the staged network setups show distinctly higher reduced χ^2 and thus, the outputs of these networks are not suitable to be fitted with the analytic function that was put forward in this analysis. Interestingly however, the fits for the staged networks in Appendix D seem to perform better on smaller signal ratios.
Most of the findings presented above could also be confirmed on the much smaller sample



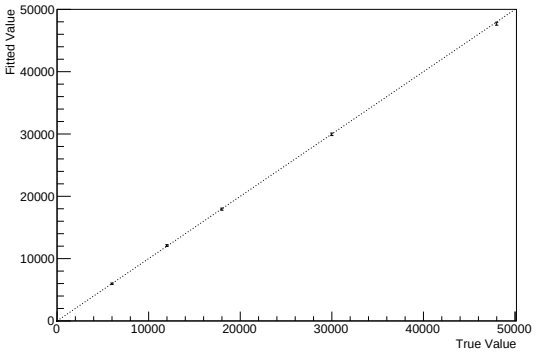
(**a**) Unstaged network, `SHAPE` = "`TOT`".



(**b**) Unstaged network, `SHAPE` = "`DIA`".



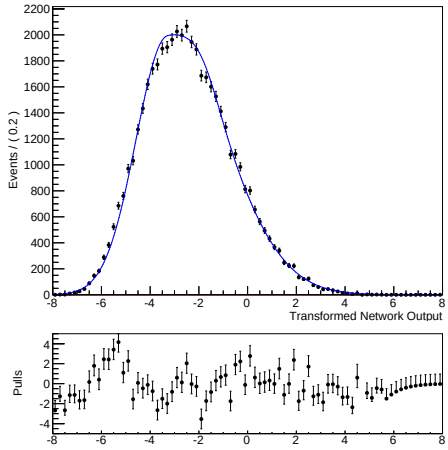
(**c**) Staged network, `SHAPE` = "`TOT`".



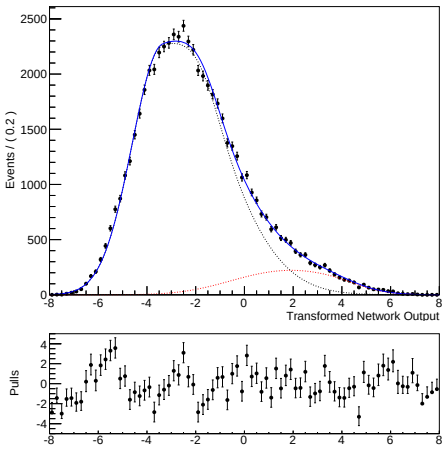
(**d**) Staged network, `SHAPE` = "`DIA`".

Figure 4.19: Plot of fitted value of signal entries over true value of signal entries for different network types.

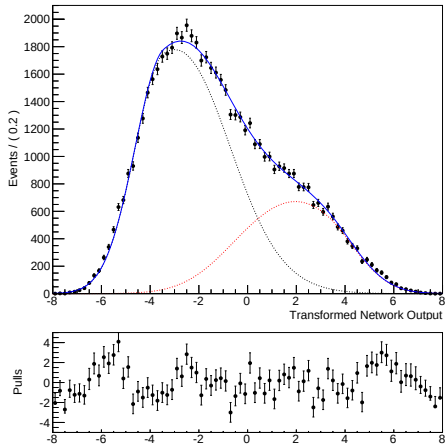
that reconstructs both B mesons described in Chapter 4.1.2. For the smaller sample it has proven useful to fix the mean of the additional Gaussian function for the signal and background fits to the mean of the bifurcated Gaussian. Otherwise, the additional Gaussian often described random fluctuations in the tails of the network output.
In summary, using an unstaged network in contrast to staged networks in previous analyses, for example in [Pri13] and [RÏ2], is advantageous if smooth outputs are required, but only if `SHAPE` is set to "`TOT`". Due to the very wiggly and peaky output of `SHAPE` = "`DIAG`" (see Figure 4.9b) this option has not been presented here. Fitting these outputs does not lead to a satisfying result.



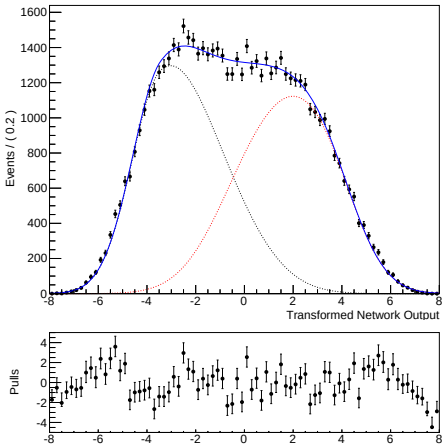
(**a**) Signal ratio 0%, $\chi^2 = 2.5$.



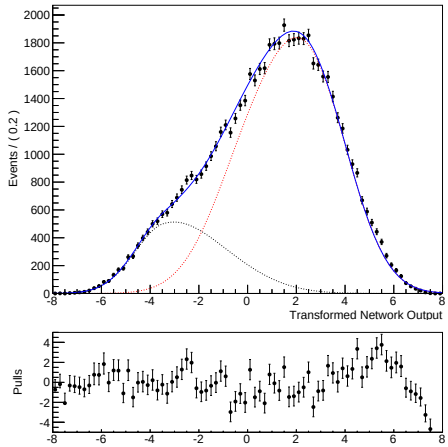
(**b**) Signal ratio 10%, $\chi^2 = 2.2$.



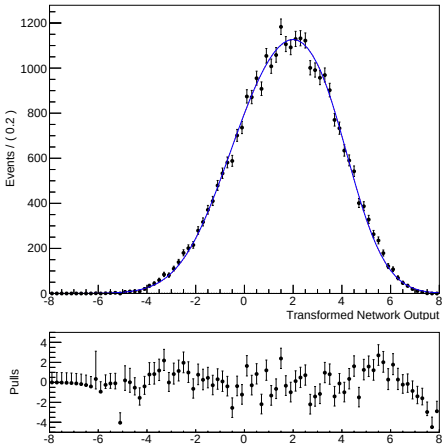
(**c**) Signal ratio 30%, $\chi^2 = 2.3$.



(**d**) Signal ratio 50%, $\chi^2 = 2.0$.



(**e**) Signal ratio 80%, $\chi^2 = 2.3$.



(**f**) Signal ratio 100%, $\chi^2 = 1.5$.

Figure 4.20: Fits of superposition of signal and background for different signal proportions in the sample for an unstaged network with `SHAPE` = "`TOT`". The reduced χ^2 of the fits are stated. The dotted curves represent the signal (red) and background (black) from which the superpositon is composed.

5 | Conclusion

In this study a multivariate technique for continuum suppression was successfully evaluated. Over 600 NeuroBayes classifications were analysed and direct connections between a large number of possible settings and their consequences on the classification were established. Many options were examined with respect to both separation efficiency and output shape and thus, this thesis will support future analyses to perform well-balanced multivariate classifications.
The possible settings were successfully studied on a global level as well as for individual variables. Furthermore, the relevance of certain groups of variables was examined.
Global preprocessing options and their influence on Gini coefficients and shape of the network output distributions was studied. To receive almost Gaussian transformed network outputs, this study suggests the use of the options `DIA` and `TOT` for the `SHAPE` parameter in NeuroBayes. The option `DIAG` used in previous analyses features equal separation quality, but is advised against if smooth network outputs are required.
The tests showed that the Gini coefficient is almost independent from the required level of significance of the variables. Nevertheless, smooth outputs will become more unlikely if the required level of significance is increased.
With respect to individual variables, Fox-Wolfram moments appear to be the most important group of variables. Especially, excluding a special group of Fox-Wolfram moments made up of two ROE particles showed to be very detrimental for the shape of the output distribution.
For individual variable preprocessing a guide has been put forward (see Chapter 4.2.2) to help the user to set up an efficient continuum suppression. In this context, two main causes for irregularities in the network output distributions have been identified and linked to the shape of the input distribution of certain variables.
In the last part of this study, the results from previous chapters were gathered to set up four networks with especially smooth output distributions. Here, unstaged networks were compared to staged networks. The network output distributions were fitted with a combination of a bifurcated Gaussian and an ordinary Gaussian. The study suggests, the outputs of unstaged networks tend to be smoother than the outputs of staged networks. Staged network outputs cannot or only badly be described with the set of analytic functions mentioned above.
In Appendix F a condensed step by step guide to set up a NeuroBayes network is presented that summarizes the results gained in this study.
Further studies on the topic of this thesis should examine individual variables more closely as the influence of individual variables on the output shape can be high. In addition to

that, it would surely be helpful to evaluate the neural networks used, in the context of rule extraction, e.g. to link certain network input distributions to the connected output shapes.

Appendix

A List of Variables

The variables used in this study modelled by Monte-Carlo are:

- Cleo-cones: `cc_0`, `cc_1`, ..., `cc_8`
- Cleo-cones flags: The eight "flagged" Cleo-cones are derived from the Cleo-cones and are a binary representation of the Cleo-cones. See the `CSBaseMethod.h` script (function: `SetValueToDelta`) for precise definition.
- `cm_costh`: Angle between momentum of reconstructed B candidate and beam axis
- `ct_bo`: Angle between thrust axis and rest of event.
- `ct_bb`: Angle between thrust axis and beam axis.
- Fox-Wolfram moments:
  - `k0hso00` - `k0hso04`: Charged momentum components
  - `k0hso10`, `k0hso12`, `k0hso14`: Neutral momentum components
  - `k0hso20`, `k0hso22`, `k0hso24`: Missing momentum components
  - `k0hoo0` - `k0hoo4`: components from ROE
  - `k0mm2`: Missing mass
  - `k0et`: Transverse momentum of B candidate
  - `r2`: Ratio of second to zeroth Fox-Wolfram moments

B DIA versus DIAG network output

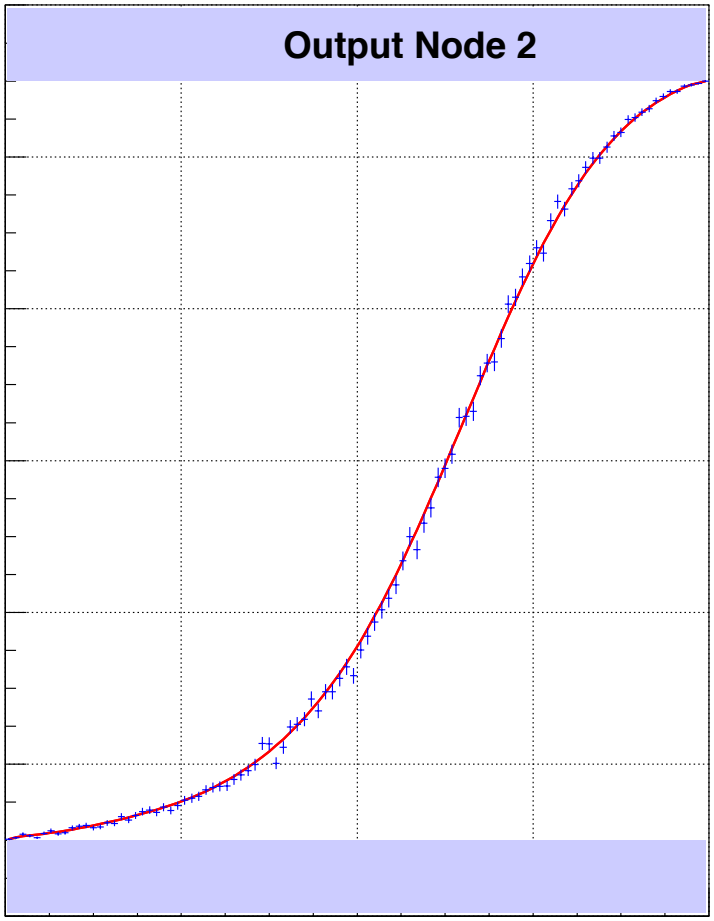


Figure B.1: NeuroBayes fit of output node distribution using `SHAPE` = "`DIA`"

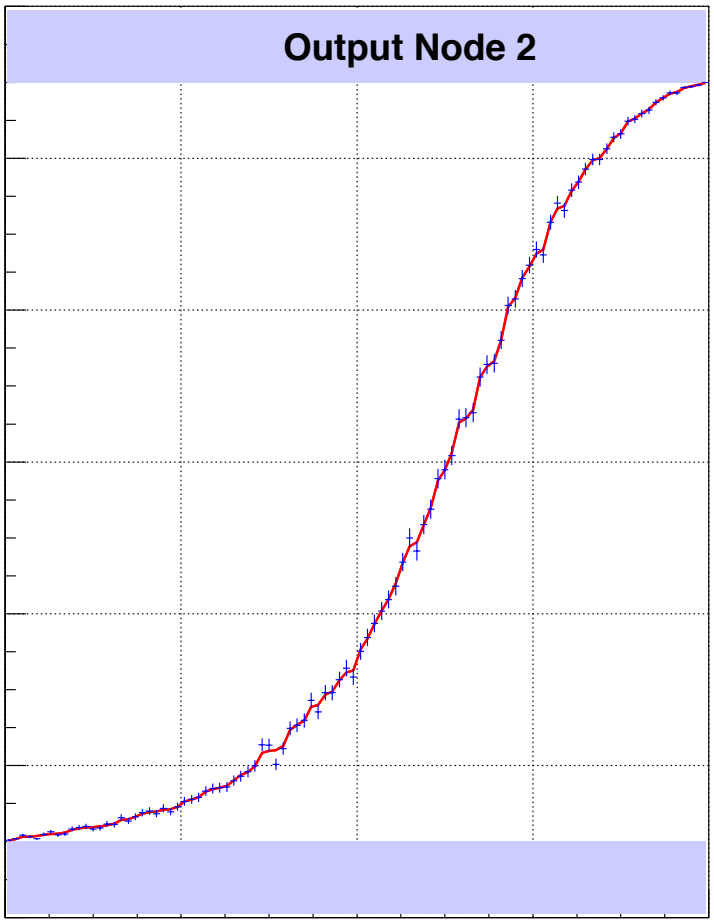


Figure B.2: NeuroBayes fit of output node distribution using `SHAPE` = "`DIAG`"

C Effects of PRE

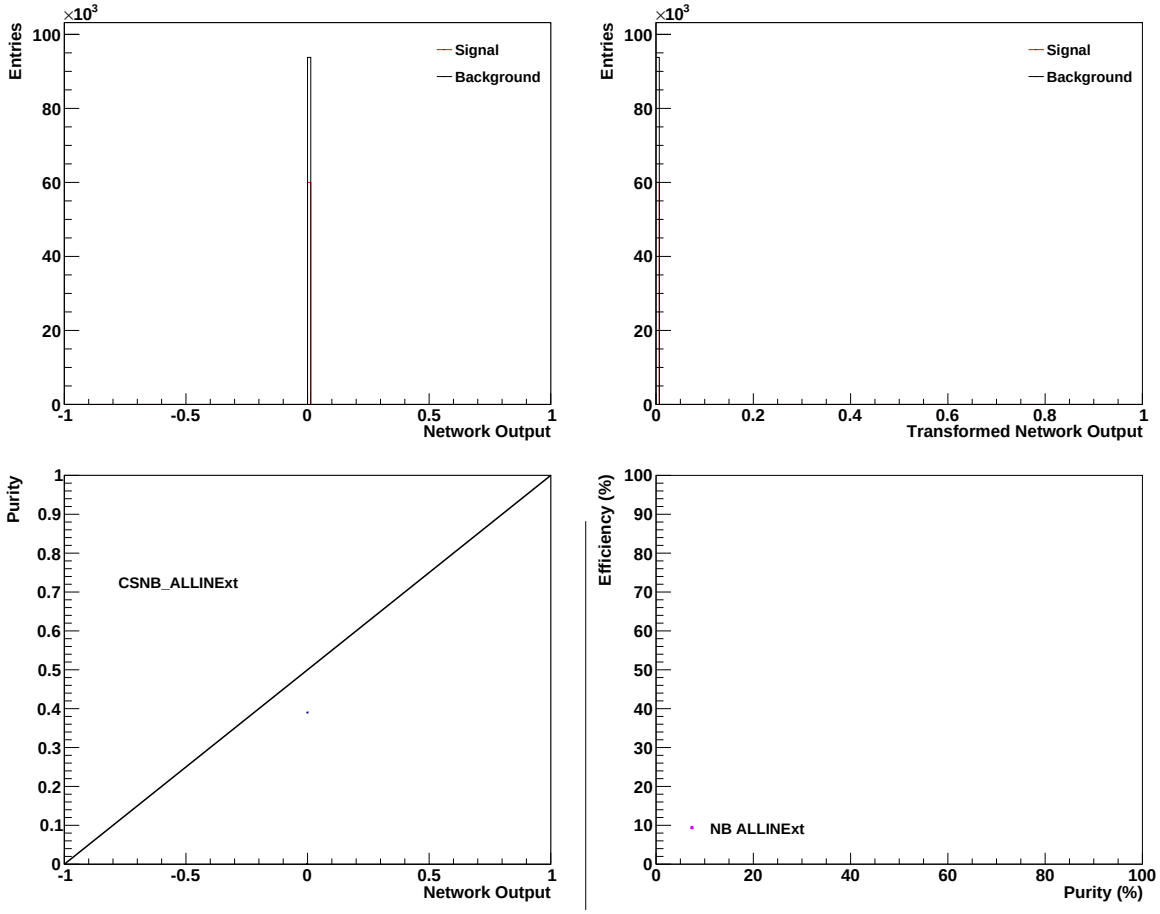


Figure C.1: Date: 13-04-06, 2014-08-06; "pre": "601", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 52.9, Ginimax = 61.0

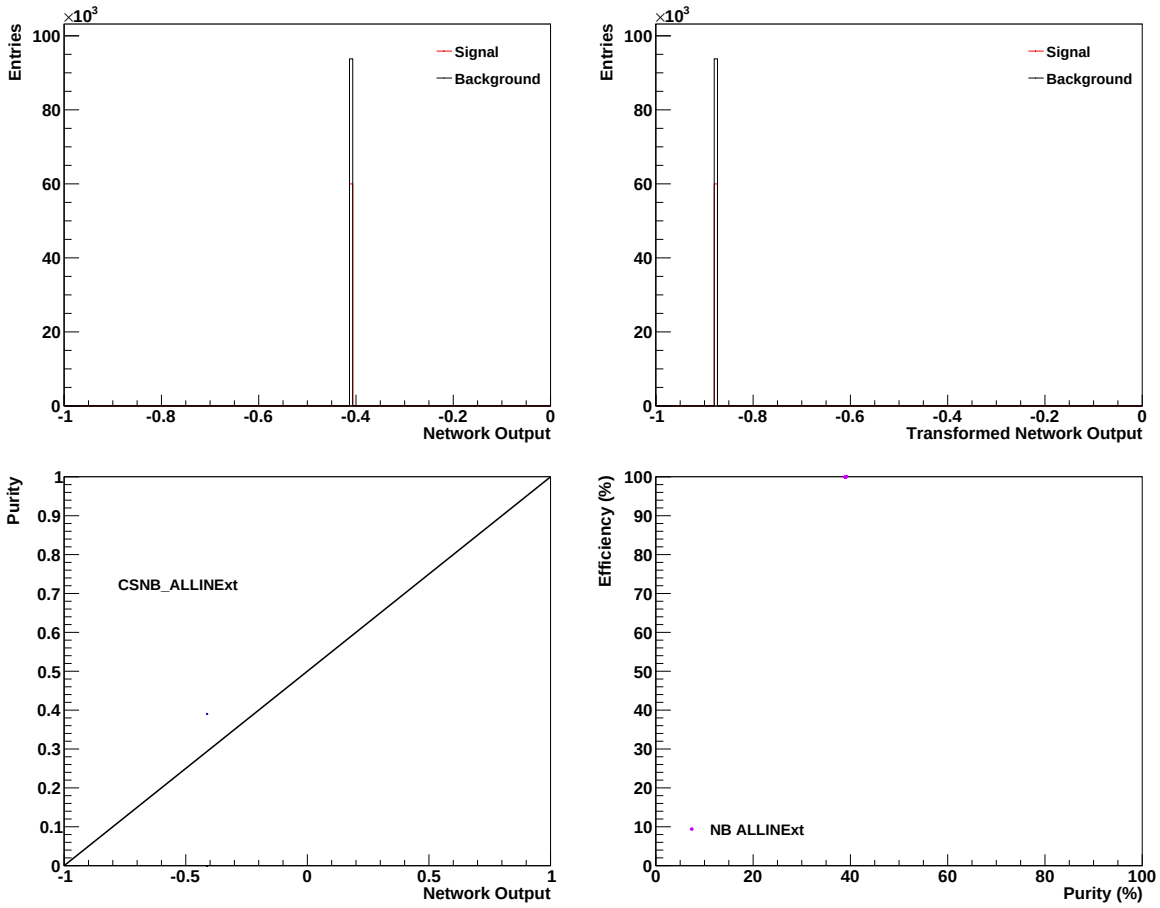


Figure C.2: Date: 12-57-51, 2014-08-06; "pre": "620", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 53.7/ 52.8 final-preboost, Ginimax = 61.0

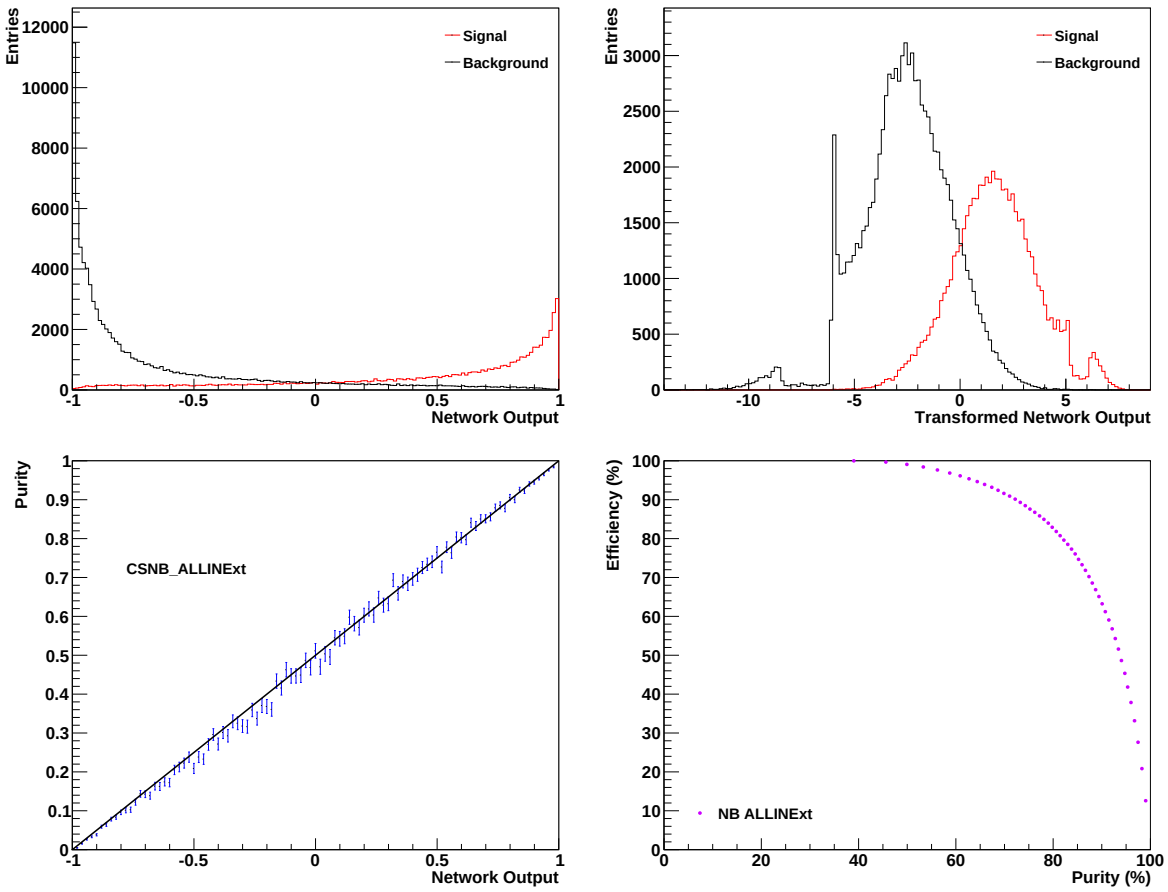


Figure C.3: Date: 11-01-06, 2014-08-06; "pre": "612", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 52.9, Ginimax = 61.0

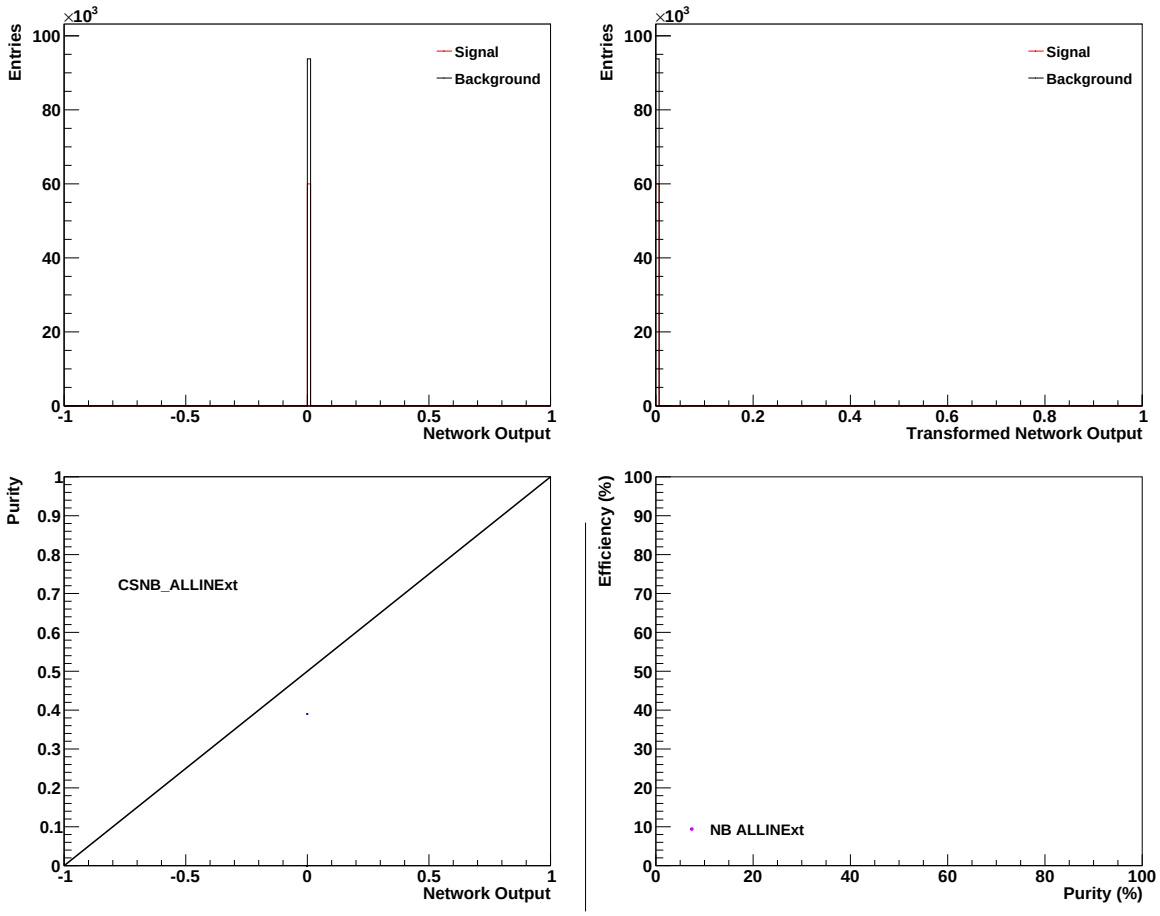


Figure C.4: Date: 13-14-10, 2014-08-06; "pre": "602", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 52.9, Ginimax = 61.0

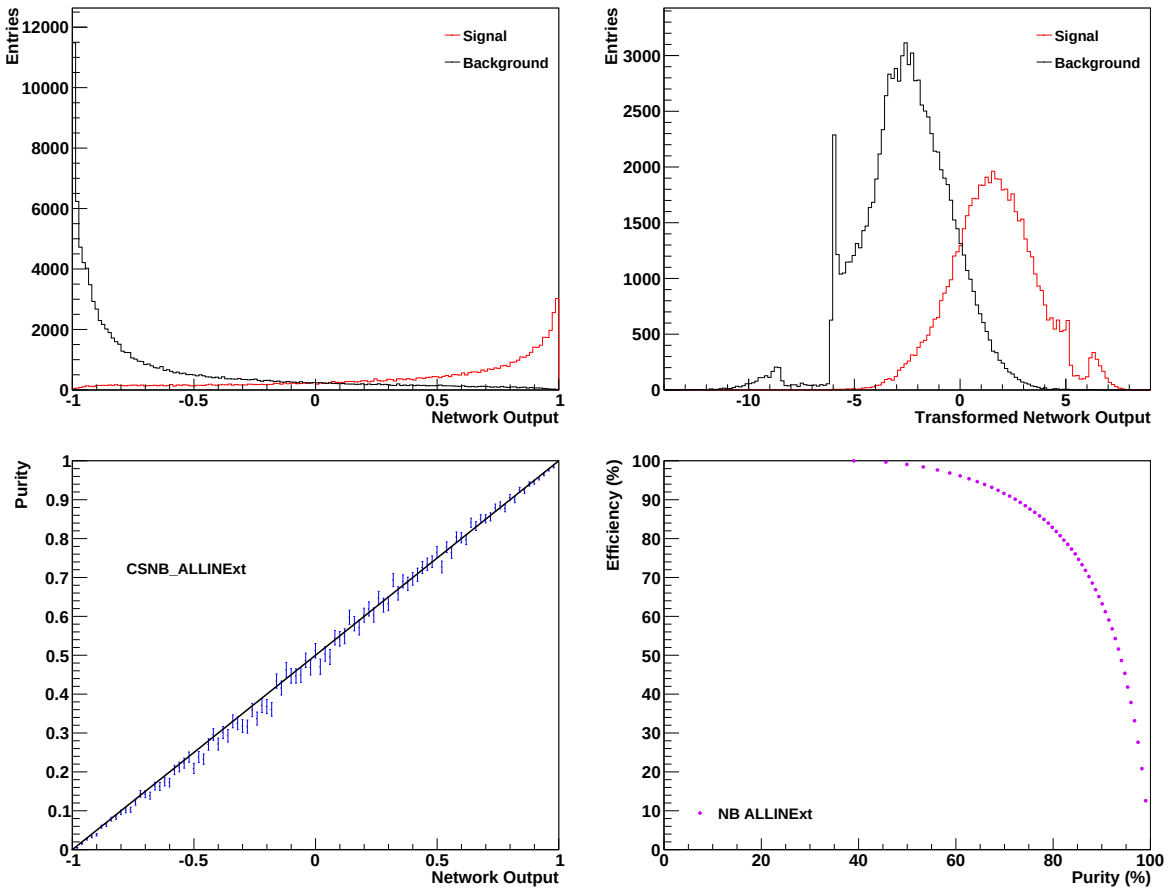


Figure C.5: Date: 11-29-09, 2014-08-06; "pre": "611", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 52.9, Ginimax = 61.0

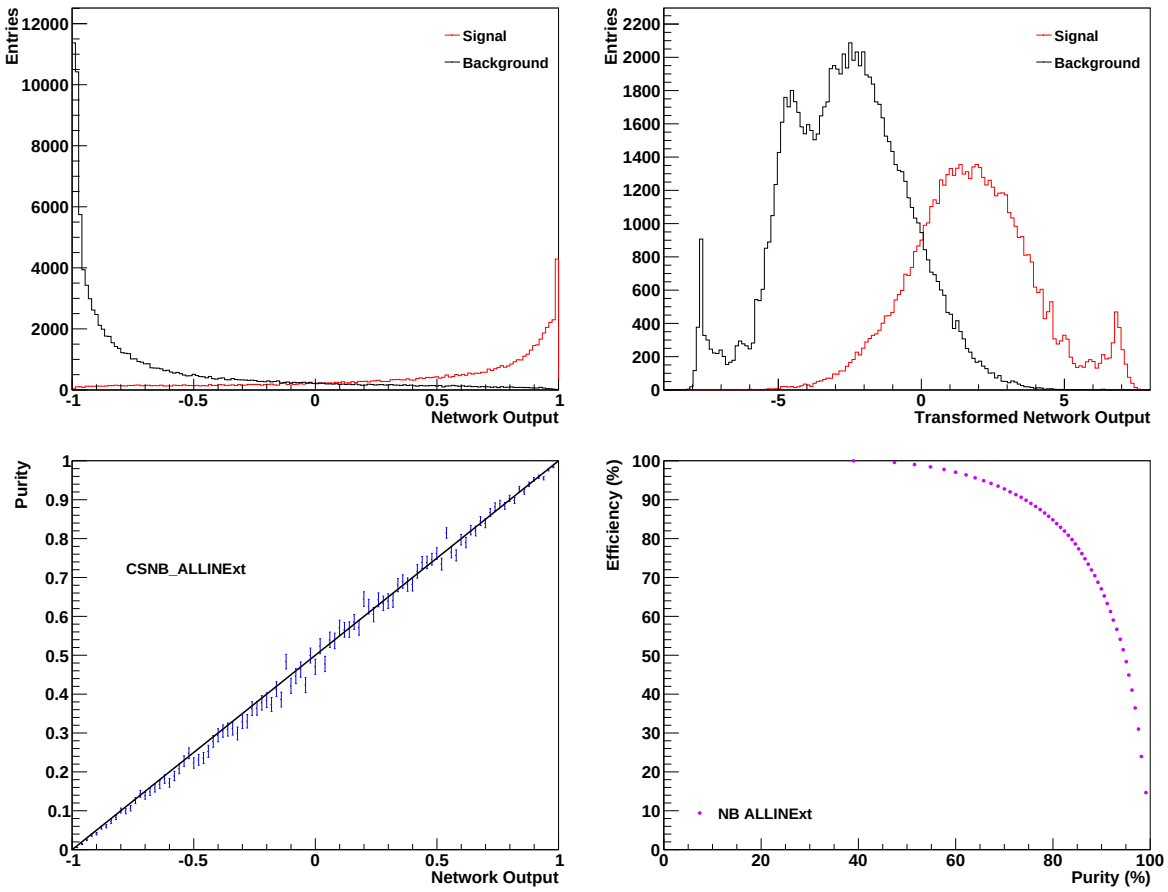


Figure C.6: Date: 11-07-36, 2014-08-06; "pre": "622", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 52.7/ 52.8 final-preboost, Ginimax = 61.0

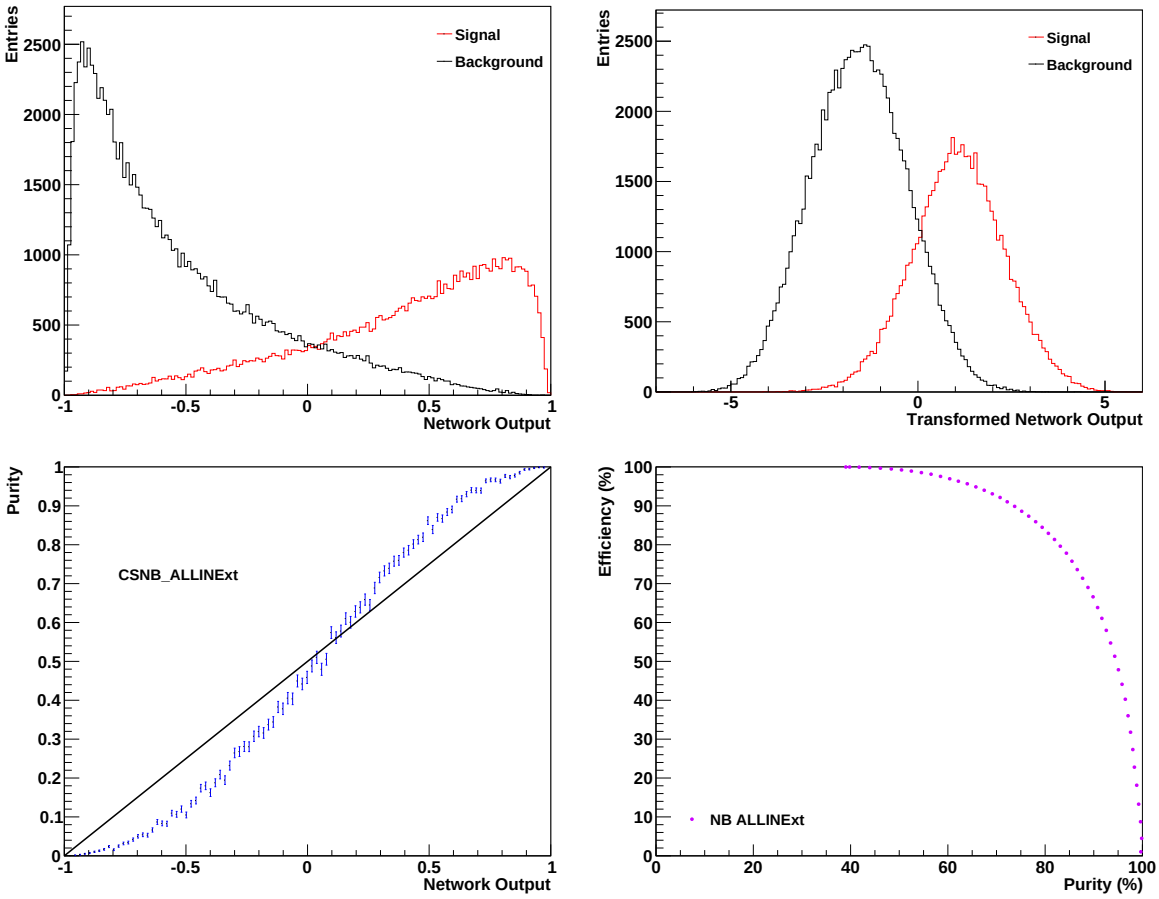


Figure C.7: Date: 11-09-32, 2014-08-06; "pre": "621", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 53.7/52.8 final-preboost, Ginimax = 61.0

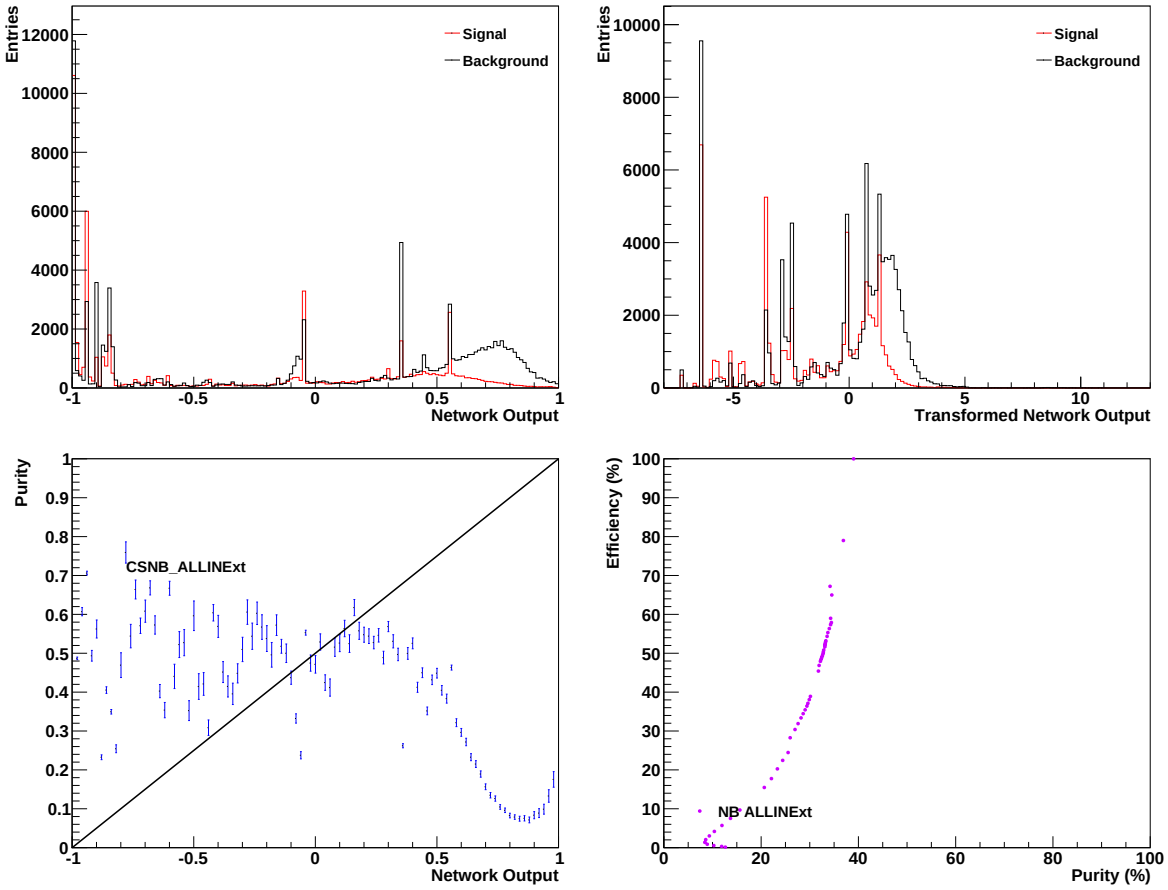
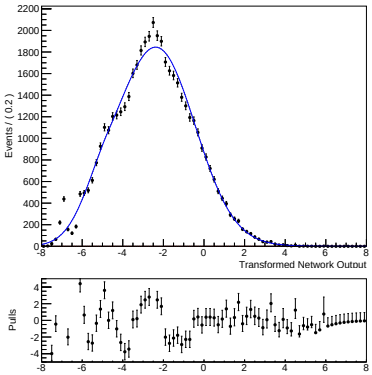
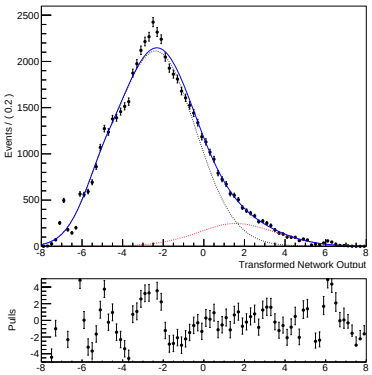


Figure C.8: Date: 11-30-01, 2014-08-06; "pre": "610", "loss": "ENTROPY", "sample": "2", "rtrain": "0.8", "shape": "DIA", "cut": "-1", "epoch": "100", "tag": "PREtest", "iterations": "300", "learndiag": "1", "reg": "OFF", "preproflags": "as original", Gini = 52.9, Ginimax = 61.0

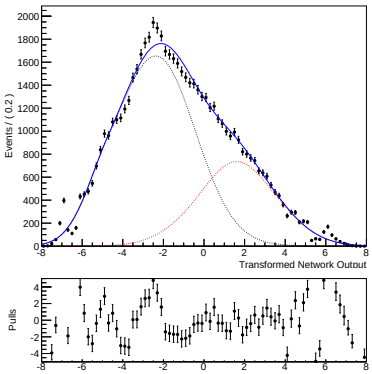
D Fit of Network Outputs



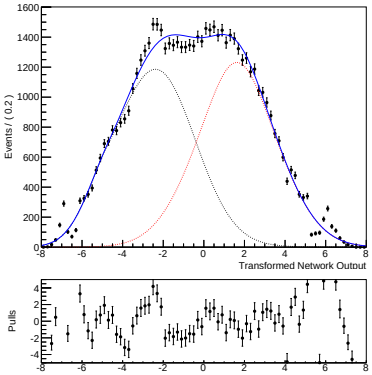
(**a**) Signal ratio 0%, $\chi^2 = 12.2$.



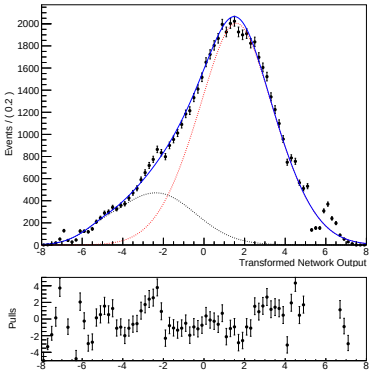
(**b**) Signal ratio 10%, $\chi^2 = 14.4$.



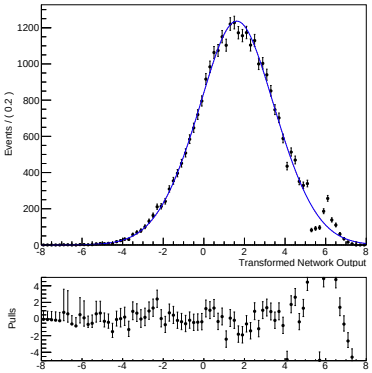
(**c**) Signal ratio 30%, $\chi^2 = 14.2$.



(**d**) Signal ratio 50%, $\chi^2 = 14.5$.

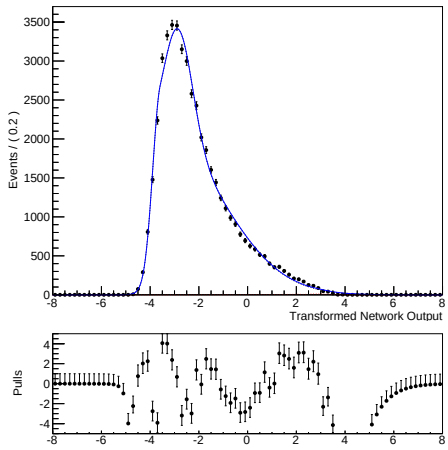


(**e**) Signal ratio 80%, $\chi^2 = 14.7$.

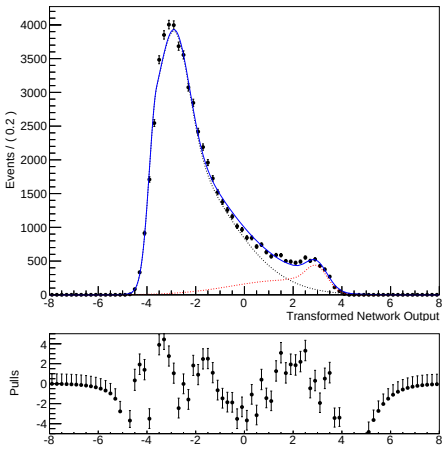


(**f**) Signal ratio 100%, $\chi^2 = 8.5$.

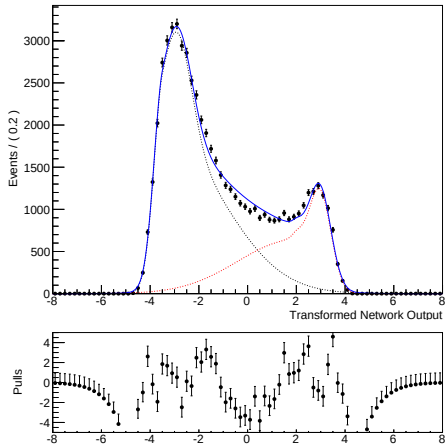
Figure D.1: Fits of superposition of signal and background for different signal proportions in the sample for an unstaged network with `SHAPE` = "`DIA`". The reduced χ^2 is stated. The dotted curves represent the signal (red) and background (black) from which the superpositon is composed.



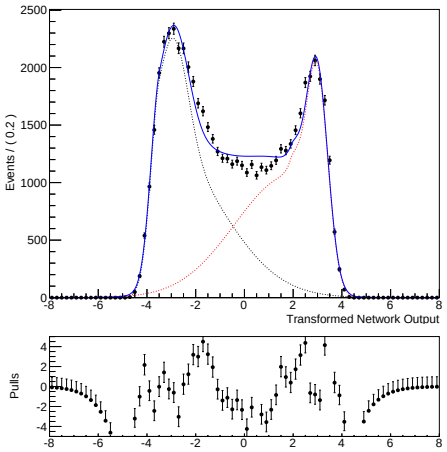
(**a**) Signal ratio 0%, $\chi^2 = 9.0$.



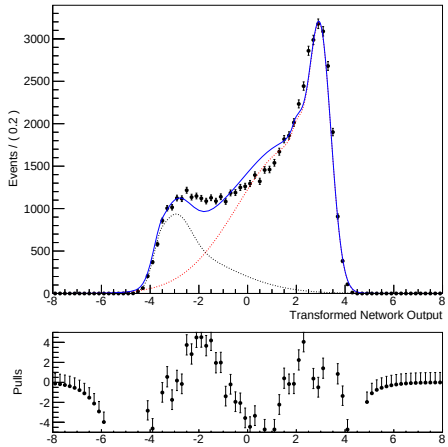
(**b**) Signal ratio 10%, $\chi^2 = 6.7$.



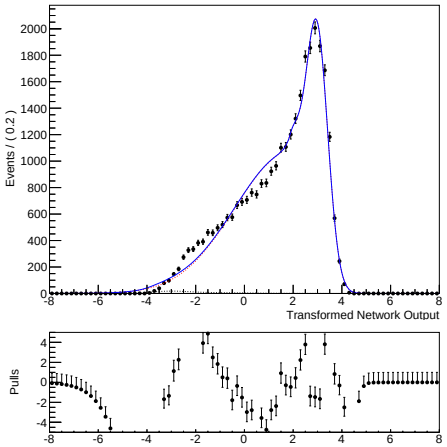
(**c**) Signal ratio 30%, $\chi^2 = 8.6$.



(**d**) Signal ratio 50%, $\chi^2 = 9.0$.

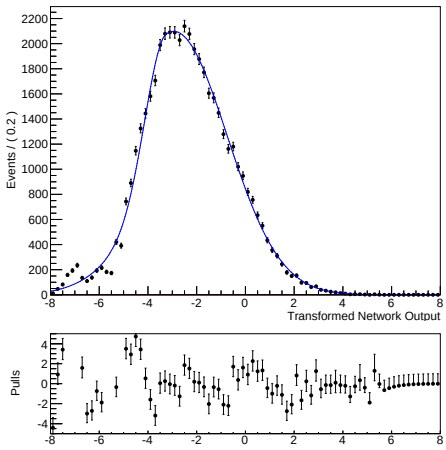


(**e**) Signal ratio 80%, $\chi^2 = 17.9$.

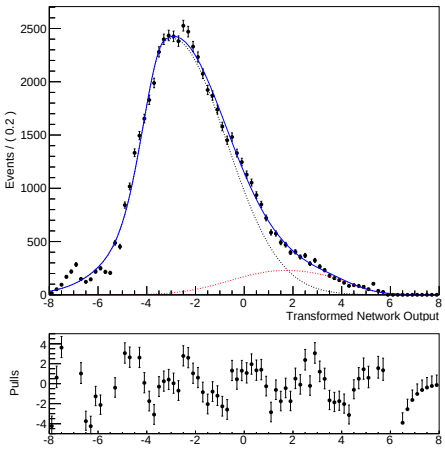


(**f**) Signal ratio 100%, $\chi^2 = 20.0$.

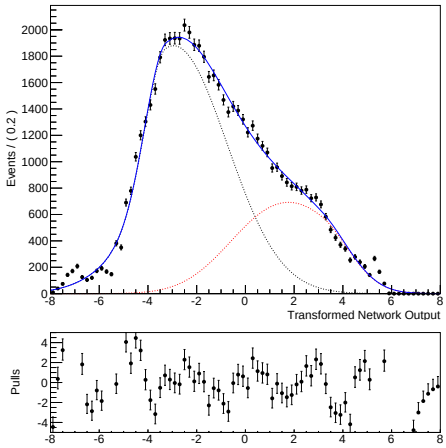
Figure D.2: Fits of superposition of signal and background for different signal proportions in the sample for an staged network with `SHAPE` = "`TOT`". The reduced χ^2 is stated. The dotted curves represent the signal (red) and background (black) from which the superpositon is composed.



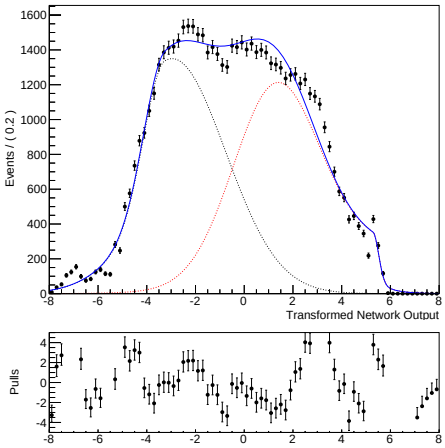
(**a**) Signal ratio 0%, $\chi^2 = 10.4$.



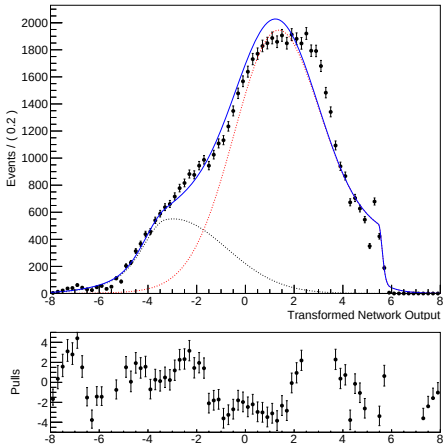
(**b**) Signal ratio 10%, $\chi^2 = 12.4$.



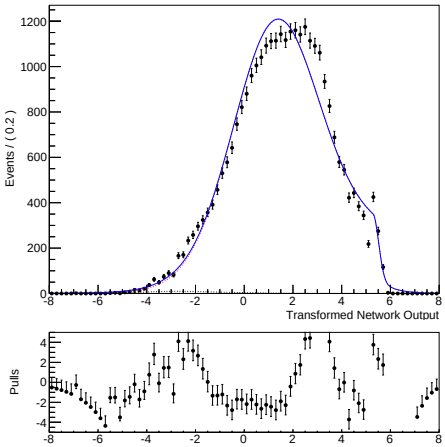
(**c**) Signal ratio 30%, $\chi^2 = 15.5$.



(**d**) Signal ratio 50%, $\chi^2 = 14.0$.



(**e**) Signal ratio 80%, $\chi^2 = 15.0$.



(**f**) Signal ratio 100%, $\chi^2 = 11.1$.

Figure D.3: Fits of superposition of signal and background for different signal proportions in the sample for a staged network with `SHAPE` = "`DIA`". The reduced χ^2 is stated. The dotted curves represent the signal (red) and background (black) from which the superpositon is composed.

E Summary of NeuroBayes Settings

In this chapter the NeuroBayes settings that are useful for continuum suppression or a binary classification task in general are presented in a condensed way.

E.1 Global Settings

There are a number of parameters that control the basic training procedure. These parameters are:

- `RTRAIN`: This parameter controls the amount of input which is used for training. Thus, this option can be used to train the network on a part of a sample, say 80%, while using the remaining part for evaluating the expertise. Possible values for `RTRAIN` range from 0 to 1. Especially, this parameter can be used to check for signs of overtraining.
- `EPOCH`: With this parameter the interval of weight update can be adjusted. `EPOCH` is a useful option if computation time is a critical issue. The default setting for `EPOCH` is 200.
- `ITER`: The maximum number of iterations, meaning the number of times all training patterns are presented to the network, is set by this parameter. However, if the network error is minimized before the number of iterations set has ellapsed, the training is ended automatically. The default is 100 iterations.
- `LOSS`: With this parameter the type of loss function used during network training is set. The three possible options are `ENTROPY`, `QUADRATIC` and `COMBINED`. The default setting `ENTROPY` will be an efficient choice in many applications and allows a probabilistic interpretation of the network output distributions. Only with `ENTROPY` the network error has a physical meaning.
- `LEARNDIAG`: With this parameter the user can add a term to the loss function corresponding to the deviation of the signal purity plot from the diagonal and thus force the network on the diagonal. The possible options are 1 (on) and 0 (off).

PRE

The global preprocessing flag consists of three integers kij and controls the type of preprocessing performed.
The integer k attains values from 1 to 9 and sets the required level of significance for a variable to be taken into the neural network training. The level of significance is computed by $k \cdot \frac{\sigma}{2}$. The parameter k should be chosen empirically.
In the encoding kij, the integer i ($i = 0, ..., 3$) determines if the variables are de-correlated, normalised or rotated before training. In the case of classification, $i = 3$ can directly be ruled out as it is not a possible option in this mode. The remaining options for i are:

- $i = 0$: The input variables are fed untouched to the net, no de-correlation is performed.
- $i = 1$: The input variables are de-correlated and normalised prior to the training.

- $i = 2$: The input variables are de-correlated and all linear dependence on target is rotated to the first new input variable (Principal Component Analysis).

The integer j sets the type of transformation applied to the input variables:

- $j = 0$: No preprocessing is performed.
- $j = 1$: The distributions of input variables are flattened.
- $j = 2$: The distributions of input variables are transformed to a Gaussian distribution.

The two most usefull options for kij in continuum suppression are $k12$ and $k22$. With both preprocessing flags the distributions of the input variables are transformed to a Gaussian. However, the main difference is the type of training performed.
With $k22$ a *booosted* training is performed. Boosting refers to performing a second training which uses higher weights for misclassified data and basically is a process of readjusting the connection weights. In the case of boosted trainings the NeuroBayes analysis file contains many plots showing pre- and post-boost results. Note, boosted trainings have higher chances of overtraining and should be evaluated by using `RTRAIN`.

REG

With this parameter the user can control the type of regularisation that is performed globally when a network is trained. The aim of regularisation has been outlined in Chapter 3.3. The possible options are:

- `OFF`: No regularisation is performed during the training.
- `REG`: A Bayesian regularisation scheme is applied, dividing weights into three different classes (weight of bias node in input layer, weights of remaining input nodes, weights of connections from hidden to output layer). This option is the standard setting.
- `ARD`: *Automatic Relevance Detection* equips all input nodes with their own regularisation constants independent of each other.
- `ASR`: *Automatic Shape Regularisation* equips all output nodes with their own regularisation constants independent of each other.
- `ALL`: Combines `ARD` and `ASR`.

For most trainings the option `REG` will be an efficient choice and lead to networks with high separation efficiency.

SHAPE

The `SHAPE` parameter controls the behaviour of direct connections from input to output layer. The different options are:

- `OFF`: No connections between input and output layer are established.
- `DIAG`: A spline fit is applied to the output of the output nodes. The fit is required to distribute the purity versus network output along the diagonal after preprocessing and before training.
- `DIA`: This option is a specialization of `DIAG`. The technique applied matches `DIAG` with the exception that a smoother spline fit is performed.
- `TOT`: Establishes direct connections between input and output layer in order to describe a linear density estimation.
- `INCL`: Direct connections between input and output layer are established to describe the inclusive distribution. This option is mainly used for shape-reconstruction and is not used in the investigations for this thesis.
- `MARGINAL`: Utilizes a binomial marginal sum method as input to the network. This setting cannot be used in this thesis as it only works for problems with uncorrelated, univariate data.

The study suggests the use of `DIA` and preferably `TOT`, if smooth output distributions are required. The highest separation efficiency is achieved with `DIAG`. However, the output distributions with this option are very unsmooth.

E.2 Individual Variable Settings

Preprocessing cannot only be performed globally, but on individual variable level. This enables the user to treat input variables differently.
The individual variable preprocessing flag consists of two integers ij. The options used in this study are $i = 1, 3, 9$ and $j = 2, 4$. The digit j defines the transformation a variable is subjected to:

- $j = 2$: Transforms variable to a Gaussian distribution.
- $j = 4$: Uses result of regularised fit to mean values of target to transform variable.

The digit i can be used to modify the action defined by j. Especially, it is useful to determine the way NeuroBayes works with missing input. Missing input denoted by the value -999 which has to be set beforehand can be modelled with δ-functions. The different options of i used are:

- $i = 1$: Use mean target and flatten the variables distribution. δ-functions will not be considered.
- $i = 3$: Use mean target and flatten the variables distribution. δ-functions are used and left untouched during transformation.

- $i = 9$: Use mean target and flatten the variables distribution. δ-functions are used and set to zero during the transformation process defined by j, while the remaining distribution is transformed to have mean zero and unit width.

In general, the user has to decide if a Gaussian transformation is applied ($j = 2$) or if the target is used for transformation ($j = 4$). The choice of i then solely depends on the treatment of δ-functions which may be in the input variables distribution if input is missing.

F Continuum Suppression Recipe

To give the user a quick introduction to the use of NeuroBayes for continuum suppression, the results presented throughout the thesis are gathered here and condensed to a few simple setup steps. The recommendations that are made, have been established using two samples (see Section 4.1.2) and thus do not assert to be absolute.
The following options are set globally and are a good point to start from: `REG` = "`REG`", `PRE` = 612, `LOSS` = "`ENTROPY`", `LEARNDIAG` = 0. For the cut which was introduced in Chapter 4.1.3 it is useful to start with the default of -1. If the user is satisfied with all other settings the cut value can be adjusted to get rid of peaks and cliffs. Now a step by step procedure is given to set up the network:

1. Start with setting up an unstaged network which takes all the variables which shall be trained. Use `SHAPE` = "`TOT`". The individual preprocessing flags which are a good starting point are outlined in Chapter 4.2.2. Then train the net.

2. Check if the purity of the network trained is distributed along the diagonal. If not try using `LEARNDIAG` = 1 to force purity onto the diagonal.

3. Use `RTRAIN`, which can assume values between zero and one, to check the net for signs of overtraining. Alternatively, use a second sample to test the expertise for overtraining.

4. If the trained network seems to be overtrained, try using the `SHAPE` = "`DIA`" option and `LEARNDIAG` = 0.

5. As soon as you feel comfortable with the network, you may try using a staged network and elaborate on the individual variable preprocessing flags.

Bibliography

[Asn+96] D. Asner et al. "Search for exclusive charmless hadronic B decays". In: *Phys. Rev. D* 53.3 (Feb. 1996), pp. 1039–1050. ISSN: 0556-2821. DOI: 10.1103/PhysRevD.53.1039.

[Bev+14] A. J. Bevan et al. "The Physics of the B Factories". In: (June 2014), pp. 109–119. arXiv:1406.6311.

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Vol. 4. 2006, p. 738. DOI: 10.1117/1.2819119. arXiv:0-387-31073-8.

[BS93] D. Besson and T. Skwarnicki. "Upsilon Spectroscopy: Transitions in the Bottomonium System". In: *Annu. Rev. Nucl. Part. Sci.* 43.1 (Dec. 1993), pp. 333–378. ISSN: 0163-8998. DOI: 10.1146/annurev.ns.43.120193.002001.

[Cab63] Nicola Cabibbo. "Unitary Symmetry and Leptonic Decays". In: *Phys. Rev. Lett.* 10.12 (June 1963), pp. 531–533. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.10.531.

[Chr+64] J. H. Christenson et al. "Evidence for the 2π Decay of the K20 Meson". In: *Phys. Rev. Lett.* 13.4 (July 1964), pp. 138–140. DOI: 10.1103/PhysRevLett.13.138.

[FK06] M. Feindt and U. Kerzel. "The NeuroBayes neural network package". In: *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.* 559.1 (Apr. 2006), pp. 190–194. ISSN: 01689002. DOI: 10.1016/j.nima.2005.11.166.

[HKP12] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining : concepts and techniques*. 3rd ed. Amsterdam: Elsevier, Morgan Kaufmann, 2012. ISBN: 978-0-12-381479-1.

[KM73] Makoto Kobayashi and Toshihide Maskawa. "CP-Violation in the Renormalizable Theory of Weak Interaction". In: *Prog. Theor. Phys.* 49.2 (1973), pp. 652–657. DOI: 10.1143/PTP.49.652.

[LY56] T. Lee and C. Yang. "Question of Parity Conservation in Weak Interactions". In: *Phys. Rev.* 104.1 (Oct. 1956), pp. 254–258. ISSN: 0031-899X. DOI: 10.1103/PhysRev.104.254.

[Neu11] Sebastian Neubauer. "Search for $B \to K^* \nu \bar{\nu}$ Decays Using a New Probabilistic Full Reconstruction Method". PhD Thesis. Karlsruhe Institute of Technology, 2011.

[Pri13] Michael Prim. "Angular Analysis of $B \to \Phi K^*$ and Search for CP Violation at the Belle Experiment". PhD Thesis. Karlsruhe Institute of Technology, 2013.

[RÏ2] M. Röhrken. "Time-Dependent CP Violation Measurements in Neutral B Meson to Double-Charm Decays at the Japanese Belle Experiment". PhD Thesis. Karlsruhe Institute of Technology, 2012.

[Phi12] Phi-t Physics Information Technologies GmbH. *The NeuroBayes® User's Guide*. Tech. rep. 2012.

List of Figures

List of Tables

Danksagungen

Ich bedanke mich bei Prof. Dr. Michael Feindt für viele hilfreiche Diskussionen zum Thema meiner Arbeit und für die Möglichkeit diese Arbeit in seiner Arbeitsgruppe durchzuführen.

Ich danke Priv. Doz. Dr. Thomas Kuhr und Dr. Pablo Goldenzweig für die Übernahme des (Kor-)Referats und die hilfreiche Unterstützung besonders in der letzten Phase der Arbeit.

Besonderer Dank gilt Bastian Kronenbitter für die umfassende Betreuung und Hilfe in vielen Belangen und letztlich auch für den spannenden Themenvorschlag.

Mein Dank gilt weiterhin allen Mitgliedern der Arbeitsgruppe, die mich durch Korrektur- und Verbesserungsvorschläge unterstützt haben und dem gesamten Institut für Experimentelle Kernphysik für eine ausgezeichnete Arbeitsatmosphäre.

Bei Timothy Gebhard bedanke ich mich für das LaTeX Template dieser Arbeit und viele motivierende Diskussionen.

Abschließend danke ich meinen Eltern, Agnes und Ulrich-Michael Büchner, und meinem Bruder Martin für ihre kontinuierliche Unterstützung in allen Lebenslagen. Ein besonders lieber Dank gilt Theresa.