University of Zurich^UZH

Universität Zürich

Using a feed forward neural network to identify Drell-Yan events at the LHCb experiment

Author
Thomas Neuer

Supervisor
Prof. Nicola Serra
Andreas Weiden

Zurich, August 2018

Abstract

A study of the application of feed-forward neural networks to the signal-background separation for the Drell-Yan process $q\bar{q} \to \mu^- \mu^+$ is performed. Data from proton-proton collisions collected at the LHCb in 2012, corresponding to a total integrated luminosity of about 2 fb^{-1} at a centre-of-mass energy of $\sqrt{s} = 8\,\text{TeV}$ is used. The results are compared with a conventional method using a χ^2 template fit and boosted decision trees (BDT). It was found that the neural networks yielded slightly better results than the BDT, independent of the used features, and comparable results to the χ^2 fit.

Contents

1 Introduction

1.1 LHC

The Large Hadron Collider (LHC) is a high-energy particle accelerator at CERN in Geneva. The ring-tunnel of the proton-proton synchrotron is situated more than 100 m below the ground. It has a circumference of more than 27 km and includes more than 9500 liquid-helium-cooled superconducting magnets. Figure 1.1 shows a sketch of the LHC accelerator chain. As two equally charged particles are collided, the LHC houses two beam pipes, which are kept at an ultra-high vacuum of the order 10^{-10} mbar. Particle beams consist of a few thousand particle bunches, each containing about 10^{11} protons. The centre-of-mass energy ($\sqrt{s}$) at the LHC started in 2010 at 900 GeV, but has been increased gradually to the final design energy of $\sqrt{s} = 14$ TeV. The data used in this thesis was taken in 2012 with a centre-of mass energy of 8 TeV.

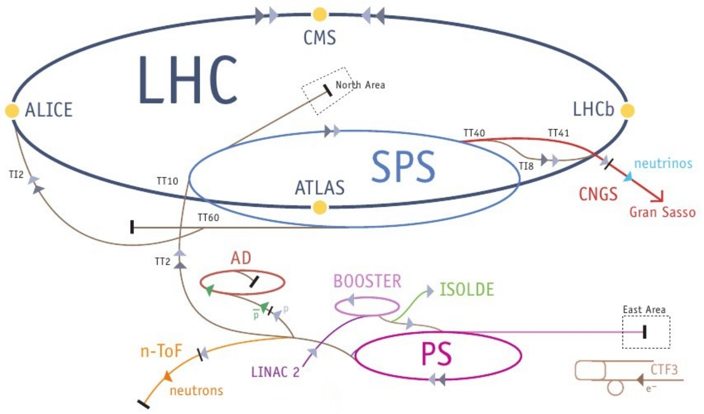


Figure 1.1: The CERN Accelarator chain [33].

The protons are collided at four collision points within the accelerator pipes at a rate of 40 MHz, one collision every 25 ns. The results of the interactions are measured by four big experiments. Two of them are huge general purpose detectors called CMS (Compact Muon Solenoid) and ATLAS (A Toroidal LHC Apparatus), designed to measure particles along both beam directions with a large angular coverage around the interaction point. ALICE (A Large Ion Collider Experiment) has another goal in mind: the study of heavy-ion collisions in order to produce a quark-gluon plasma. Similar conditions are believed to have existed a fraction of a second after the Big Bang before quarks and gluons bound together to form hadrons and heavier particles.

1.1.1 LHCb experiment

The Large Hadron Collider beauty (LHCb) experiment is the fourth large experiment at the LHC and is mainly concerned with B-physics, which is the study of particles containing the bottom quark (also known as beauty quark, see figure 1.3). The LHCb experiment, unlike the other three experiments, has no cylindrical symmetry around the interaction point, as can be seen in figure 1.2.

Its angular coverage in θ is approximately 30 to 300 (250) mrad in the

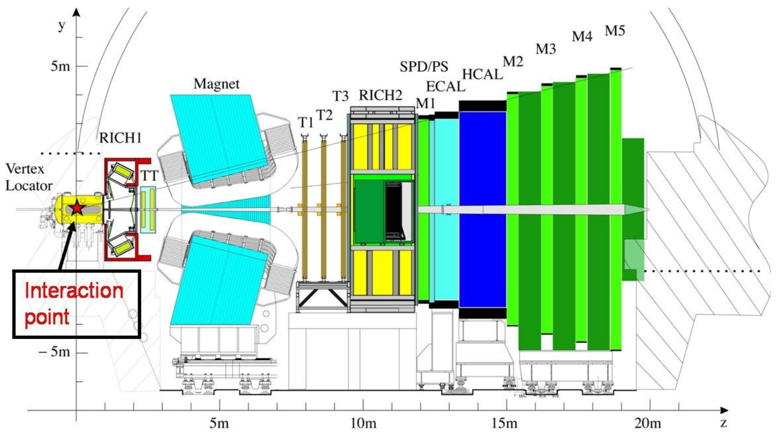


Figure 1.2: Schematic side view of the LHCb experiment. Collisions take place at the interaction point and are then measured by the one sided detector arm[35].

bending (non-bending) plane [21], corresponding to a pseudorapidity η between 2 and 4.5, where

$$\eta = -\log\left[\tan\left(\frac{\theta}{2}\right)\right]. \tag{1.1}$$

The particles produced at the interaction point in forward direction traverse several detector layers. These detectors measure different properties of the particles, often with multiple redundancy. The detector consists of a precise tracking system (vertex locator (VELO), Tracker Turicensis (TT), Magnet and straw tube detectors T1-T3), measuring the trajectory and hence momentum component transverse to the magnetic field as well as the charge of the particles [22]. Several types of charged hadrons and leptons are differentiated via a measurement in the two Ring-imaging Cherenkov detectors (RICH 1/2). Candidates for hadrons and charged leptons as well as photons are differentiated using a system of pre-shower (PS) and silicon-pad detectors (SPD), electromagnetic calorimeter (ECAL) and hadronic calorimeter (HCAL). Muons can traverse all calorimeters and are detected in the Muon system (M1 in coincidence with M2-M4).

Vertex locator

The Vertex Locator (VELO) of the LHCb experiment consists of a silicon strip detector surrounding the proton-proton collision point. It is designed to precisely measure the particles trajectories immediately after being produced and determine the vertex they originated from. It consists of 21 modules, each with two silicon microstrip sensors. During data taking, the inner edge of detector is only 8 mm away from the beam [21].

Muon system

The muon stations M1-M5 serve the purpose of detecting high transverse momentum particles including a measurement of the particles position. M1 is stationed before the calorimeter system for a better transverse momentum (p_T) resolution while the other stations are in the outer regions of the LHCb. Mostly muons are able to penetrate through the electromagnetic and hadronic calorimeter as they are minimally ionizing particles (MIPs) over a wide range of momenta. Therefore they leave the cleanest signal of all elementary particles in the detectors. The spatial resolution is diminished the further away the detector is positioned from the interaction point, as more area can be covered with a detector of lower granularity.

Magnet

The dipole magnet consists of two saddle shaped aluminium coils inside an iron yoke and is operated at environment temperature. The field created by the magnet has its main component along the y-axis. In the z-direction the field has a large gradient, resulting in a field of less than 2 mT inside the RICH envelopes but an overall bending power of 4 Tm. The magnet deflects charged particles in the positive or negative x-direction (coordinate system as defined in figure 1.2) allowing the determination of the charge of a passing particle based on the observed curvature. To minimise systematic effects, the polarity of the magnetic field can be reversed. Data taken with the two different polarities are assigned to the "magnet-up" or the "magnet-down" dataset, depending on the direction of the field's y-component.

Trigger

The Trigger has a hardware and multiple software stages. The hardware trigger is called level zero trigger (L0) and it reduces the rate to about 1 MHz. It evaluates information from the calorimeters and Muon system. The trigger fires if one or more pre-defined conditions, called trigger lines, are satisfied. In this thesis, the cuts based on muon information are important, see table 3 for a summary of trigger conditions. At this rate the whole detector information can be read-out and used in the software stage. If at least two muons in a collision pass the trigger conditions, the event is saved and further processed. In the next step, the software trigger, called High Level Trigger (HLT), implements at first a partial event reconstruction in order to reduce the rate to 43 kHz (HLT1). Then a more complete event reconstruction algorithm, a Kalman-Filter, is used in the final stage of the trigger system. If more than two muons are triggered, one of which comes from some background source (see chapter 1.3.1), all possible combinations of muons are saved as different events.

1.2 The Standard Model of particle physics

The Standard Model (SM) is a collection of quantum field theories, using local gauge invariance as the fundamental axiom. It includes all known forces except gravity and explains the interaction of spin-1/2 fermions via the mediation of spin-1 vector bosons. The dynamics are fully determined from the Lagrangian in eq. (1.2) and the principle of least action.

$$\mathcal{L}_{SM} = \mathcal{L}_{fermion} + \mathcal{L}_{gauge} + \mathcal{L}_{Yukawa} + \mathcal{L}_{Higgs}[30], \tag{1.2}$$

where

- $\mathcal{L}_{fermion}$: kinetic term for fermions
- $\mathcal{L}_{gauge}$: kinetic term for gauge bosons
- $\mathcal{L}_{Yukawa}$: mass terms for fermions
- $\mathcal{L}_{Higgs}$: Higgs term

A list of all particles contained in the SM is shown in figure 1.3. For clarity the different colours for quarks and the corresponding antiparticles are not shown. So far the Standard model has withstood every experimental test. Especially in the sector of Quantum Electrodynamics, the quantum field theory of electromagnetism, the agreement between theoretical predictions and experiment are of an unprecedented precision. The final ingredient of

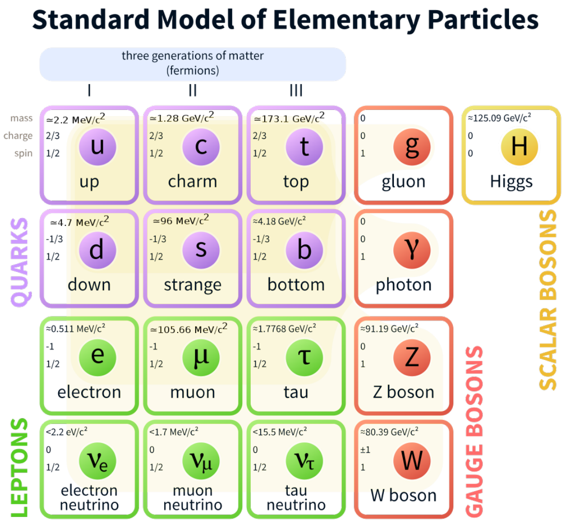
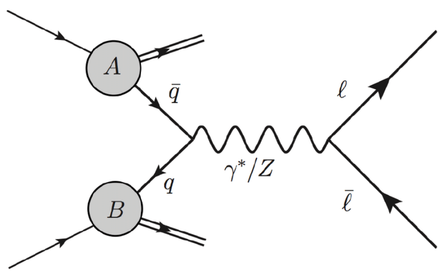


Figure 1.3: The Standard Model of particle physics. All of the quarks additionally come in three different colours (red, green and blue). For every particle there is a corresponding anti-particle with opposite quantum numbers [34].

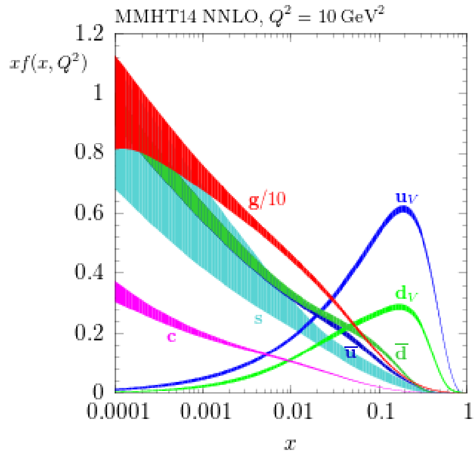
the Standard Model was found experimentally in 2012 with the discovery of the scalar Higgs boson [16, 17], which is a crucial prediction of the Higgs mechanism. It describes the process of spontaneous symmetry breaking of the SU(2) vacuum state, needed to obtain a consistent description of the massive fermions and vector bosons ($W^{\pm}, Z$), while keeping the gluons and photon massless.

1.3 The Drell-Yan process

The Drell-Yan process describes the electroweak quark/anti-quark annihilation mediated by a neutral vector boson, so either the photon or the Z-Boson [1]. Several particle/anti-particle final states are possible, but in the following only the di-lepton, specifically the di-muon, final state is considered. Those final state particles are most easily detected and differentiated from other elementary particles at the LHCb experiment. Figure 1.4a shows the corresponding (tree-level) Feynman diagram. Theoretical predictions are available up to next-to-next-to-leading order (NNLO) [2].



(a)



(b)

Figure 1.4: (a) Tree-level Feynman diagram of the Drell-Yan process. In a pp or $p\bar{p}$ collision, a quark/anti-quark pair interacts via the weak or electromagnetic force to produce a muon/anti-muon pair [38]. (b) Parton distribution function of the constituents of a proton. One can see that a large proportion of the moment is carried by sea-quarks and soft gluons [37].

For theoretical predictions the parton distribution function (pdf) of the protons is needed. As the Drell-Yan process involves one anti-quark, in a proton-proton collision one sea anti-quark needs to participate in the interaction. All the valence quarks, which form a proton, are quarks, not anti-quarks. The LHCb experiment measures particles in the forward direction and at a high rapidity. Therefore the momenta of the interacting quarks need to be highly asymmetric, meaning that one has to carry a large fraction of the proton's momentum (called x), while the other has to have a small x. While the pdf has already been constrained by previous experiments (HERA, Tevatron, see figure 1.4b), the low x region is still not well explored. Therefore the measurement and analysis of this process at the LHCb experiment allows for new constraints of the low momentum region of the pdfs.

1.3.1 Backgrounds

The signature of the Drell-Yan process considered in this thesis is the di-muon final state, identified by the Muon system of the LHCb experiment. These muons should originate directly from the primary vertex (PV) and should be highly isolated (see section 2.1.1). But other processes can imitate this behaviour and are therefore considered background [22]. Some of those are:

1. Pions or kaons can decay into muons in flight ($\pi^{\pm} \rightarrow \mu^{\pm}\nu_{\mu^{\pm}}$ or $K^{\pm} \rightarrow \mu^{\pm}\nu_{\mu^{\pm}}\gamma$) or they themself can punch through the calorimeter system into the Muon system, therefore being misidentified as a muon.

2. Heavy-flavour hadrons can decay semi-leptonically producing a muon (e.g. $B^{\pm} \rightarrow X\mu^{\pm}$), which is then possibly combined with another muon in the event.

3. The Drell-Yan process $q\overline{q} \rightarrow \tau^{+}\tau^{-} \rightarrow \mu^{+}\mu^{-}\overline{\nu_{\mu}}\nu_{\mu}\overline{\nu_{\tau}}\nu_{\tau}$ is considered as a background.

In this thesis only background from the second source was considered. For more information on how the background sample was obtained and its measured properties, see section 2.2

2 Data

The data used for training the machine learning algorithm is composed of a signal and background dataset.

The signal (sg) was obtained from Monte Carlo simulation of the Drell-Yan process via the standard software PYTHIA 8 for simulating the particle interaction momentum space. GEANT4 was used in order to simulate the interaction of the elementary particles with the detector material of LHCb. The background (bg) sample was obtained directly from data of the LHCb measurements in 2012 at a centre-of-mass energy of $\sqrt{s} = 8\,\mathrm{TeV}$. The heavy flavour background sample was selected by performing a cut on the quality of the reconstructed Z^0 end-vertex $\chi^2_{Z^0} > 15$. This property measures the quality of the reconstructed di-muon candidate, since for signal events both muons need to come from the same point. A low value meaning that both tracks originated with high probability from the same point, while a high value indicates that at least one muon originated from a secondary process, such that our background sample is rather clean.

For the final evaluation, a dataset from the 2012 run at the LHCb at $\sqrt{s} = 8\,\mathrm{TeV}$ is used, with a cut $\chi^2_{Z^0} < 5$. This dataset has contributions from both signal and background. The dataset serves for the final application of the feed forward neural network and the following comparison to the χ^2 template fit commonly used. A comparison of both the training dataset (simulation and clean background) and the evaluation dataset is shown in fig 2.1.

All datasets also exist with both polarizations of the magnet, as described in 1.1.1, respectively. In a perfect and symmetric detector such an operation should leave the final result unchanged. They are often used to account for detector asymmetries and inefficiencies during the run. This is an important aspect in the LHCb experiment, because of its many asymmetry measurements.
In the following thesis the training and calibration of the neural networks was done with the upward polarization of the magnet exclusively. The final algorithm was then also applied to the downward polarization data to check for the validity of the analysis, as those samples are statistically independent and of almost equal size.

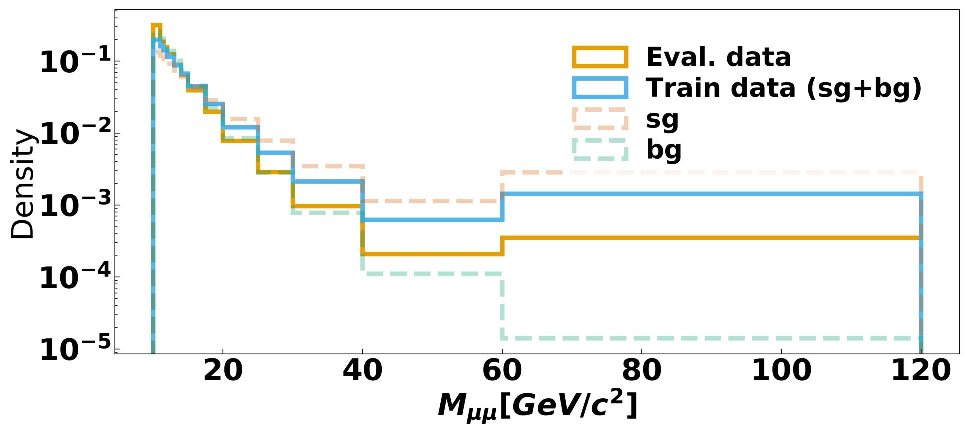


Figure 2.1: Comparison of the evaluation dataset and the training data for the reconstructed mass.

2.1 Features

Different properties of the particles are measured in the detectors of the LHCb experiment. An incomplete overview over the used features is given in table 1 and table 2. Some of these properties are given in different representations, meaning redundant information. For example the particle momentum is given in different coordinate systems (normal euclidean space p_x, p_y, p_z or spherical coordinates p_T, η, ϕ). The isolation features served as a high level baseline.

2.1.1 Isolation

One special high-level feature is called the *isolation*. It is calculated by summing up the transverse momentum of particle tracks within a cone around the seed track and subtracting the transverse momentum of particles within a smaller cone of radius r, see figure 2.2.

The transverse momentum in the inner cone is subtracted to account for Bremsstrahlung photons in the vicinity of the particle in question. The transverse momentum of those originally belonged to the signal muon and did not originate from secondary vertices. This feature was determined for each muon and finally the larger of the two isolation values was used during training.

As the name suggests this feature measures the isolation of a particle, being zero for fully isolated muons. As the signal muons in a Drell-Yan process are produced without other particles being involved and at a high transverse

Table 1: Description of the most important features used in the analysis, resulting in 1057 features in the transverse case. The number of features originates from 6 muon measurements, the number of tracks and the track measurements. The maximal number of tracks per event was chosen to be 150 (see section 2.2), resulting in 1050 features for the non-muon tracks. The track features therefore are an array of tracks, originating from the same primary vertex as the Drell-Yan candidate.

Feature (Transverse)	Explanation
muminus_PT, muplus_PT	Transverse momentum of the triggering muons
muminus_TrEta, muplus_TrEta	Polar angle of the triggering muons
muminus_TrPhi, muplus_TrPhi	Azimuthal angle of the triggering muons
nTracks	Number of tracks reconstructed for the event
tracks_PT	Transverse momentum of the remaining tracks
tracks_eta	Polar angle of the remaining tracks
tracks_phi	Azimuthal angle of the remaining tracks
tracks_IP	Impact parameter, minimal distance of extrapolated tracks from the primary vertex
tracks_IPCHI2	Quality of reconstructed impact parameter
tracks_charge	Charge of the measured particles
tracks_isMuon	Muon indicator

Table 2: Description of the most important features used for the base line fit using the isolation variable.

Feature (Isolation)	Explanation
muminus_PT, muplus_PT	as above
muminus_TrEta, muplus_TrEta	as above
max_Iso	Maximal isolation value chosen from both signal muons.
max_MINIP	Maximal impact parameter chosen from both signal muons

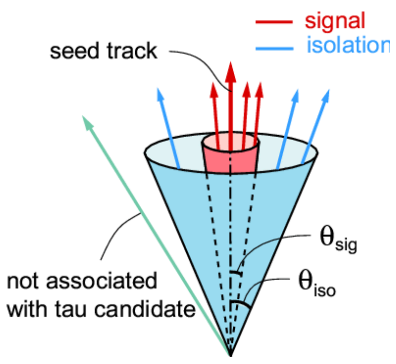


Figure 2.2: Schematic calculation of the isolation. The inner cone accounts for Bremsstrahlung photons which are not a contamination but part of the signal [31].

momentum we expect them to be well-isolated. Muons originating from background processes like combinatorial (= random) backgrounds or particle decays (i.e. $B \to \mu\nu_\mu\gamma$) are often accompanied by remains of the secondary vertex.
The isolation is a high-level feature which is often used in the analysis of electroweak processes to differentiate background from signal. The deep learning approach of this thesis should test an alternative using only low level features in order to make faster decisions and using less input from the analysts. However, using the isolation has been shown to work well in this kind of problem and is hence used as a baseline comparison.

2.2 Selection

The selection of events is an important step, as it influences all further analysis of the data and application of the neural networks. The most important cuts and the reason for their application are summarised in table 3. Most of these selections are performed to reduce the background and the large statistics at the LHCb experiment.
However, one of those features needs further explanation, as it was added specifically because of the requirements of the feed-forward neural network approach.

Number of tracks

The trigger for this analysis requires at least two signal muons with opposite sign coming from a primary vertex but naturally, due to the debris

Table 3: Preselection cuts performed on the data. The starred properties (*) were cut by the trigger before the offline analysis. About 56% of the data passed the remaining selection.

Cut	Description	Reason
$*p_T > 3\,\mathrm{GeV}$ $*p > 10\ \mathrm{GeV}$	Transverse and absolute momentum of the muon obtained from tracker measurements.	Boost in one direction, due to selection of high energy events.
$*2 < \eta < 4.5$	The rapidity is a measure of the direction of the particle, equivalent to the usage of the polar angle Θ.	Fully instrumented region of the LHCb experiment.
$*P(\chi^2_{tr}) > 0.001$	After applying the Kalman filter the probability of the track coming from a certain vertex can be obtained.	Only reasonably reconstructed tracks are used for analysis.
$m_{\mu\mu} > 10\,\mathrm{GeV}/c^2$ $m_{\mu\mu} < 120\,\mathrm{GeV}/c^2$	The invariant mass of the di-muon system obtained from energy and momentum measurements.	Imprecise simulation in low energy spectrum. Barely any statistics for high energy.
$\mathrm{Hits}_{SPD} \leq 600$	Multiplicity of the event, number of hits in the silicon pad detector.	High multiplicity events are not well simulated.
$nTrack \leq 150$	Number of reconstructed tracks (=particles) in the event. The two signature muon tracks are excluded.	See description in text
$\min(IP) < 1000$	Minimal distance of the reconstructed from the primary vertex. Signal events are expected to come directly from the PV.	This value should be small, and such a high value for the impact parameter indicates an overflow.

of the collision, there are a lot more particles produced in hadron-hadron collisions. The number of reconstructed tracks per event is called n_{tracks}. Of course, the number of those tracks is not fixed, but statistically distributed. As many neural network types require a fixed size of the input vector over all instances (=training examples), only a fixed amount of tracks can be used for this algorithm. Therefore, the length of the tracks' properties (p_T, η, ϕ, minimal impact parameter, χ^2_{IP}, charge and muon identifier, see table 2) in every event is truncated to 150 identified particles after sorting by transverse momentum p_T. This number was chosen as it results in a reasonable amount of input features, while leading to almost no loss of events (see table 7 in the appendix). Events with less than the required amount of tracks are filled with zeros for the features p_T, η, ϕ, minimal impact parameter and χ^2_{IP}. For the charge ($\in \{-1, 0, 1\}$) the filler-value was negative two. For the muon identifier ($\in \{0, 1\}$) a filler-value of negative one was chosen.

2.3 Preprocessing

Preprocessing the data is an important and often necessary step before applying any machine learning algorithm. Often the two major steps involved are *cleaning* and *scaling* the data [6].

Cleaning

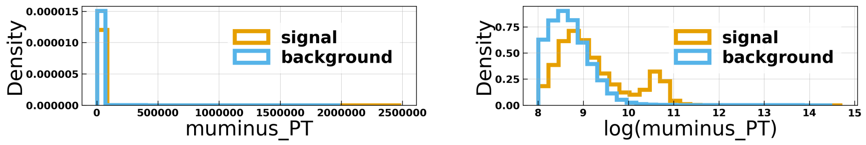
Data cleaning consists of removing missing or meaningless values/features and outliers which would make it harder for the algorithm to learn useful relations. The dataset used in this study was already very pure due to considerable pre-processing, so data cleaning was mostly concerned with equalizing all instance lengths and filling missing values in the tracks variables, as explained at the end of the previous chapter. Also the performed cuts described in the previous chapter were part of the data cleaning process.

Scaling

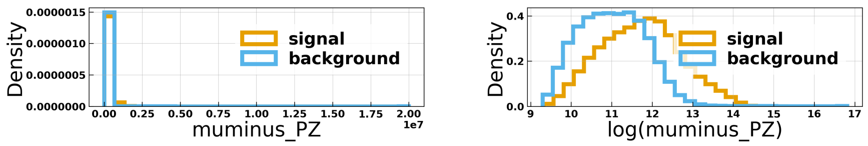
As the name suggests scaling is used in order to make different features more comparable to each other. Some features have intrinsically higher values than others, for example transverse momentum $p_T \in [0, \infty)$ but $Q \in \{-1, 0, 1\}$. It may then be difficult for many algorithms to compare features with these vastly different ranges of values without proper scaling. Two prominent techniques of scaling were applied on the dataset used for this work.
The first being a log-transformation. It is calculated on features with a range $[0, \infty)$ using the equation

$$f_i^{log} = \log(f_i + 1), \tag{2.1}$$

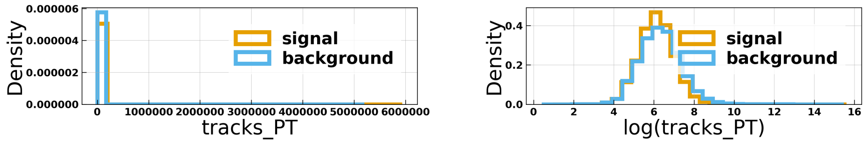
where f is the original feature and f^{log} is the transformed one. A "+1" is added to improve the properties of the transformation, mainly the faithful mapping of zero to zero and a continued positive range of the new feature. As seen in figure 2.3, the log-transformation greatly improves the behaviour of the feature distributions.



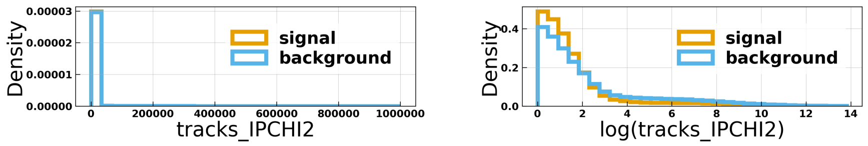
(a)



(b)



(c)



(d)

Figure 2.3: Comparison of some features before (left) and after (right) performing a logarithmic transformation.

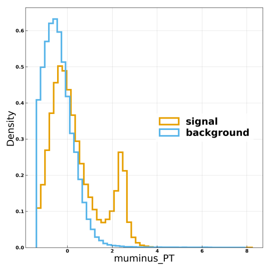
The second technique is called standardization. The transformed feature is calculated by

$$f_i^{stand} = \frac{f_i - \langle f \rangle}{\sqrt{\langle f^2 \rangle - \langle f \rangle^2}}, \tag{2.2}$$

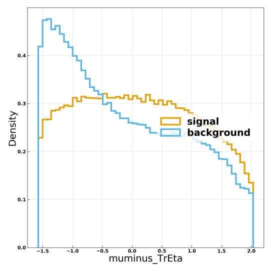
where $\langle f \rangle$ indicates the mean over all values of the feature f. The main advantages of this technique are:

1. the new feature is unitless in a sense that it is now given in units of standard deviations away from the mean and
2. the mean of the new feature is zero and its standard deviation is one.

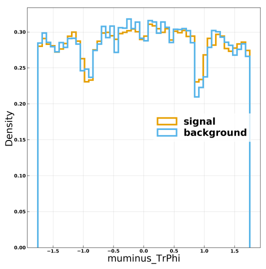
The result of this transformation is shown in figure 2.4.



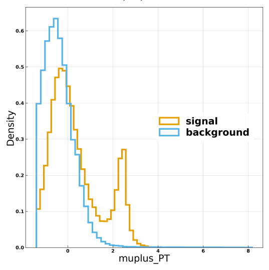
(a)



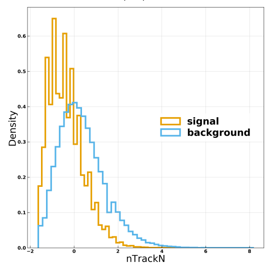
(b)



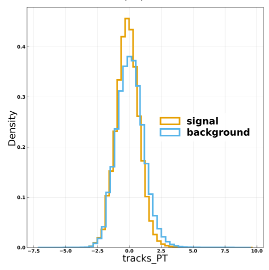
(c)



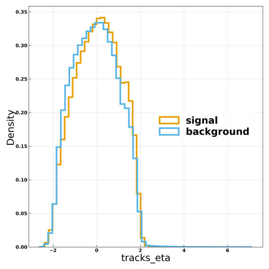
(d)



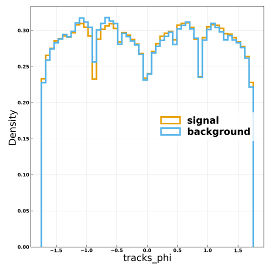
(e)



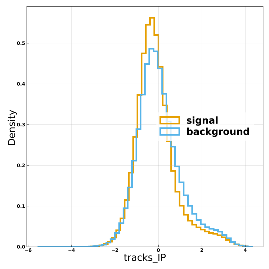
(f)



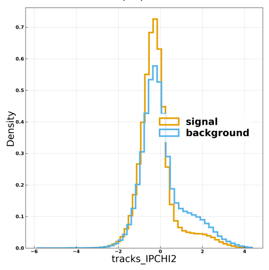
(g)



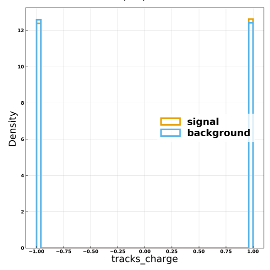
(h)



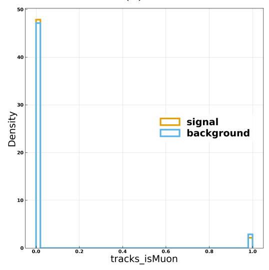
(i)



(j)



(k)



(l)

Figure 2.4: Most important processed features used during training.

2.3.1 Train-Test-Validation split

A common problem for many machine learning algorithms is called *overfitting*. It describes the tendency of a model with a large amount of parameters to be too sensitive to its input, meaning that if the input is slightly changed, the model might change drastically, see figure 2.5. In other words, the model does not only fit a useful relation between input and response, but also the statistical fluctuations leading to a loss in generalization.

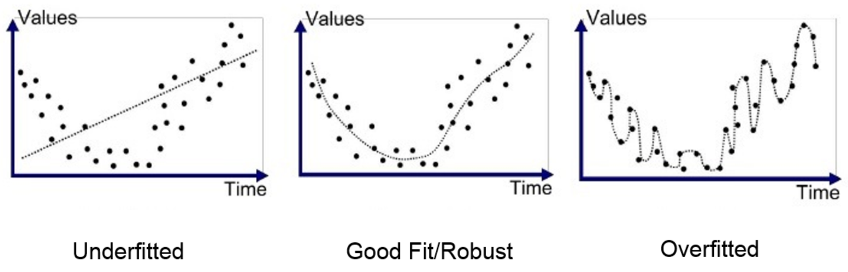


Figure 2.5: Showing the effects of an underfitted, good and overfitted model [36].

As described in chapter 3, the amount of parameters in a neural network, especially in deep learning, is huge and therefore these models tend to overfit the data. In order to monitor this tendency one of the most common practice is to split the data into three parts [15]:

1. Training data (∼65%): The algorithm learns all relations on this data and performs its optimization using these instances.
2. Validation data (∼15%): After convergence, different scores measuring the quality of the model can be calculated on this sample. This will result in a good estimate of the model's generalization capability as it has not seen this sample during training. In most problems many different models are fitted, so these scores allow for comparison of the different models.
3. Test data (∼20%): There are often al lot of hyperparameters (for example parameters tuning the optimization algorithm) for every model. Often the best initial configuration of these hyperparameters is far from obvious, therefore some sort of (randomized/sparse) grid search needs to be performed. The danger is, that these hyperparameters

overfit the validation data and their scores. Hence, the test data is a completely independent dataset which can be used to assess the quality of the final model, chosen according to the scores on the validation set and evaluate its generalization power.

The opposite effect is called underfitting and is mostly due to overly constrained and inflexible models. This problem is less pressing and rarely an issue in deep learning and other modern machine learning techniques.

2.3.2 Principal Components Analysis

As mentioned earlier, neural networks contain a lot of free parameters which need to be learned. One way to reduce the amount of these parameters is to reduce the number of features, as this allows for the usage of smaller networks. However, information is precious in every machine learning algorithm and loss of content should be kept at a minimum. To accomplish this, a dimensionality reduction technique can be applied. A popular method is called *Principal Components Analysis* (PCA). This corresponds to a rotation of the data into a lower dimensional space [10, 7]. It involves the diagonalization of a distance matrix and a following projection on the

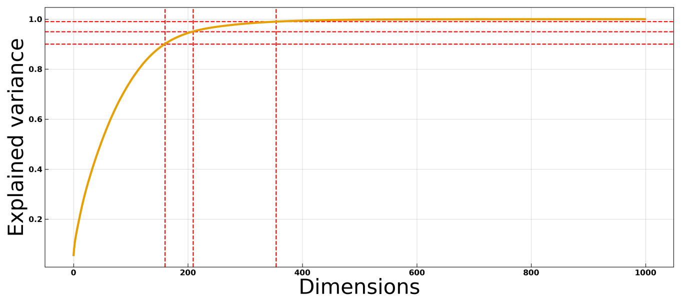


Figure 2.6: Result of the principal components analysis. The explained variance of the data is given as a function of the dimensions used. The red line on the far right indicates that with 354 features about 99% of the variance can be explained. The other pairs drawn are (209, 0.95) and (160, 0.9)

eigenvectors, called the principal components. In figure 2.6 the analysis of the PCA performed on the training data is shown.

The explained variance ($\cong$ inverse of lost information) is given as a function of the number of principal components used in the rotation of the vector space. It is worth mentioning, that with only about a third of the

input features the explained variance is still kept at 99%.
A major disadvantage of this method is the difficulty to interpret the new features. Sometimes a deeper meaning of the new representation can be found after thorough analysis of the effects of the dimensionality reduction, but often this meaning is blurred and hard to apprehend.
Neural networks with and without the usage of the principal components analysis were used during training to evaluate the loss of information. Less features are preferred in deep learning as it may greatly improve the speed of the algorithm.

3 Neural Network algorithms

Neural networks are a machine learning algorithm first introduced in the 1950's as a simplified model which can be implemented mathematically. However, they were not effectively used until the 1980's [27, 9]. With increase in computing power and affordability of fast hardware, like GPU's, they became more and more popular. Nowadays they are generally accepted to be one of the most powerful and flexible machine learning algorithms, mastering tasks from standard classification and regression to highly complex problems of image recognition and speech processing, hence being able to deal with spatial as well as temporal data. The type of neural networks used in this thesis are called *feed forward neural networks*, as the data used in this thesis had no temporal structure. A schematic view of a feed-forward neural network is shown in figure 3.1.

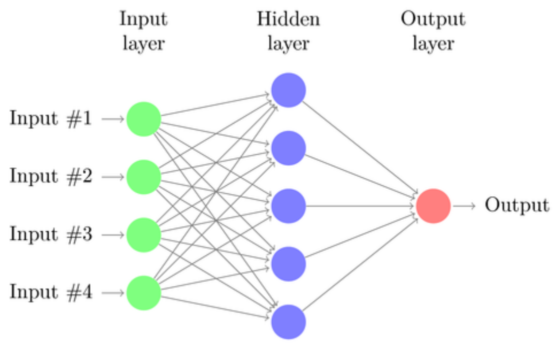


Figure 3.1: Schematic view of a shallow one layer network, with five nodes, and four input features [32].

It is a supervised machine learning algorithm, meaning that for every training example ($\hat{=}$ instance), there exists a true label. In the problem at hand the goal is to differentiate between signal and background. A feed forward neural network consists of an input layer, taking the measured properties ($\hat{=}$ features) as input, several hidden layers and an output layer, predicting the class label of an instance. As there always needs to be an input and output layer, the stated number of layers usually refers to the number of hidden layers.
Every layer consists of n_{nodes} nodes. The prescription for every layer in a neural network is given by

$$a_i^{(l)} = f_i^{(l)}\left(z_i^{(l)}\right) \equiv f_i^{(l)}\left(\sum_{j=1}^{n^{(l-1)}} W_{ij}^{(l)} \cdot a_j^{(l-1)} + b_i^{(l)}\right), \tag{3.1}$$

where $a_i^{(l)}$ is the output of the ith node in the lth layer, $f_i^{(l)}$ is called an activation function, $n^{(l-1)}$ is the number of nodes in the previous layer, $W_{ij}^{(l)}$ the weight matrix, $a_i^{(l-1)}$ the output of the previous layer and $b_i^{(l)}$ is its bias. The weight matrix and bias are learned during training, the activation function is a hyperparameter chosen according to the problem, see section 3.4 for more information.

3.1 Parameters & Hyperparameters

One of the greatest advantage as well as disadvantage of a neural network is the immense number of parameters and hyperparameters. The great number of parameters, including weights and biases, allows for very flexible and complex models, simultaneously being more prone to overfitting the data. There exists methods in order to prevent this, but the effectiveness strongly depends on the data. Some of these methods are described below.

Table 4: Summary of the parameters used in the neural network algorithm.

Parameters	
Symbol	Description
w_{ij}^l	weight in the lth layer of the ith node for the jth feature.
b_i^l	bias in the lth layer of the ith node.
n_{layer}	number of layers for the whole network structure.
n_{nodes}^l	number of nodes for the lth layer.

Hyperparameters are parameters which control not the size of the model itself but rather parameters used for algorithms within the model, one of the most important being the learning rate, see section 3.2.1. They need to be

separately tuned for every fixed structure of a network in order to obtain the best model possible.
An overview of the most important standard parameters and hyperparameters is given in table 4 and table 5.

Table 5: Summary of the hyperparameters used in the neural network modelling. The most important ones are explained further below.

Hyperparameters	
Symbol	Description
$init(\vec{\beta})$	initialization method of the weights and biases.
$f^l(\cdot)$	activation function of the lth layer.
p^l_{drop}	Dropout probability of the lth layer.
$o(\vec{\alpha})$	Optimizer o with parameters $\vec{\alpha}$, including the learning rate.
$L(y, p; \lambda_{reg})$	Loss function of the target and prediction with possible regularization terms.
s_{batch}	number of instances used per loss calculation.
n_{epochs}	maximum number of epochs for the network to be trained. One iteration over the whole dataset in batches is called epoch.

3.2 Loss functions

The loss function L, often referred to as cost function or error function, measures in some way the distance between the prediction and the known truth. It's main properties are

$$L\left(\vec{y}_i, \vec{o}_i\right) \geq 0 \quad \text{and} \quad L\left(\vec{y}_i, \vec{o}_i\right) = 0 \Leftrightarrow \vec{y}_i = \vec{o}_i \quad \forall i, \tag{3.2}$$

where $\vec{y}_i$ is the true label of the ith instance and $\vec{o}_i$ its predicted output by the network. So the loss is a positive function of the known labels and the prediction, being zero only if all the predicted labels are equal to the truth. The goal is therefore the minimization of that loss.
There are several popular loss functions [28], two of them being

- quadratic loss: $L(\vec{y}_i, \vec{o}_i) = \sum_i (\|\vec{y}_i - \vec{o}_i\|)^2$
- cross entropy for K-class classification:

$$L(\vec{y}_i, \vec{o}_i) = \frac{1}{n} \sum_{i=0}^{n} \sum_{j=0}^{K} \left(y_i^{(j)} \log\left(o_i^{(j)}\right) + (1 - y_i^{(j)}) \log\left(1 - o_i^{(j)}\right)\right)$$

.

Both loss functions satisfy equation (3.2) as can easily be checked by the reader. The algorithm learns by minimizing the loss function iteratively until a minimum is found.

3.2.1 Gradient Descent

Gradient Descent is an algorithm trying to minimize a function by moving in the direction of the steepest descent. In the case of the neural network (and many other machine learning techniques), the minimum value of a loss function L needs to be obtained. As shown in equation (3.2), the loss is a function of the known label and the prediction of the algorithm. The true label will not change during training and can therefore be considered a constant parameter, not a variable. The prediction $\vec{o}_i$ of instance i is a function of the model parameters. Those need to be updated iteratively in order to obtain a minimum. Its update rule is

$$\vec{\beta}_{n+1} = \vec{\beta}_n - \alpha \nabla L(\vec{\beta}_n) \quad n = 0, 1, 2..., \tag{3.3}$$

where $\vec{\beta}_n$ is the vector of all model parameters (weights and biases) at step n and α is called the learning rate or step size. Its influence on the convergence of the Gradient Descent algorithm can be seen in figure 3.2. The weights at step $n = 0$ can be initialized in various ways and may influence the behaviour of the optimization algorithm [14].

However, as it turns out in praxis it is much more efficient to calculate an approximate gradient using a small part of the data, called batch, instead of the exact gradient calculated from all data. The result is called *Stochastic Gradient Descent* (SGD), as it is not ensured that every step brings one

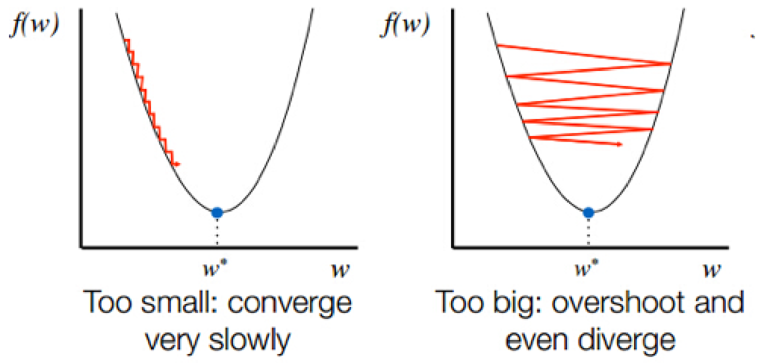


Figure 3.2: Schematic application of Gradient descent with a learning rate which is too small (left) and to large (right) [41].

closer to the minimum but it is only true on average, over a large number of steps. Still, this version of the algorithm converges much faster in most applications. One then iterates over all batches (usually of size 32, 64, 128,...) and then starts a new training iteration, called an epoch.
In real world applications, more sophisticated algorithms than the one stated in equation (3.3) are used for the minimization, including regularization terms, momentum estimates and decay rate schedulers, which lower the step size when approaching a minimum.
The currently most popular among these is called *Adam* (Adaptive momentum estimate), which implements all of the stated improvements [24].

3.3 Backpropagation

Backpropagation is the mathematical method explaining how to calculate the central quantity in equation (3.3): the gradient of the loss function [3]. It is given by four equations [27].

$$\delta^N \equiv \nabla_{z^N} L = \nabla_{a^N} L \circ f'(z^N) \tag{3.4}$$
$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \circ f'(z^l) \tag{3.5}$$
$$\nabla_{b^l} L = \delta^l \tag{3.6}$$
$$\frac{\partial L}{\partial (W^l)_{jk}} = a_k^{l-1} \delta_j^l, \tag{3.7}$$
$$\tag{3.8}$$

where δ^l is the error calculated in layer l $(= 1, ..., N)$, a^l (z^l) is the vector containing the output of all nodes in layer l after (before) applying the activation function, $f'(x)$ is the derivative of the activation function resulting in one output per node, W^l is again the weight matrix of the corresponding layer and b^l is its bias. The operation " $\circ$ " is the Hadamard product defined as

$$\vec{a} \circ \vec{b} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \circ \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} a_1 \cdot b_1 \\ a_2 \cdot b_2 \\ \vdots \\ a_n \cdot b_n \end{pmatrix}. \tag{3.9}$$

All four equation (3.4)-(3.7) are a direct consequence of the chain rule of derivatives and can be calculated using equation (3.1). The first two equations (3.4) and (3.5) state a formula to calculate the error in the last layer and how to propagate it backwards to previous layers (hence the name: *backpropagation*). Equations (3.6) and (3.7) then relate these errors to the derivatives with respect to the weights and biases. Exactly these derivatives are needed in the calculation of the next Gradient Descent step, see again equation (3.3).

3.4 Activations

Activation functions are one of the most important hyperparameters used in neural networks. The activations are the non-linear functions applied on every node after multiplying the input with the weights and adding the bias. It is essential for the learning procedure that these activations are non-linear functions as linear functions would result in a simple neural network imitating the results from classical multivariate linear regression and making it impossible to learn complex relations between features. A comparison of the most important ones and their derivatives is shown in figure 3.3. While the *sigmoid*, given by [26]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.10}$$

was the first activation function being used in bigger, successful neural networks, its importance in modern applications has faded. The most used activations today implement the *hyperbolic-tangent* (*tanh*), calculated as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.11}$$

or the *rectified linear units* (*relu*), given by

$$\text{relu}(x) = \max(0, x). \tag{3.12}$$

Two version called *Lrelu* and *elastic linear unit* (*elu*), improved in certain aspects, are given by

$$\text{Lrelu}(x) = \begin{cases} 0.01x & x < 0 \\ x & x \geq 0 \end{cases} \tag{3.13}$$

$$\text{elu}(x) = \begin{cases} \exp(x-1) & x < 0 \\ x & x \geq 0. \end{cases} \tag{3.14}$$

The last layer in classification problems often features a special activation function, called softmax function, given by

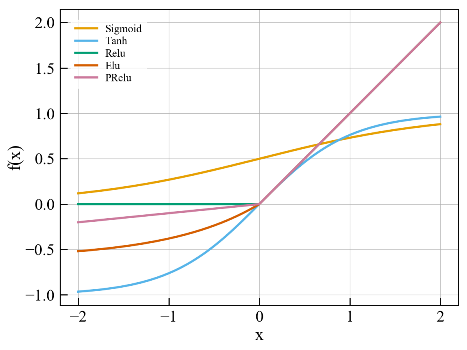
$$\text{s}_i(\vec{x}) = \frac{\exp(x_i)}{\sum_{k=1}^{K} \exp(x_k)} \quad \text{for K classes.} \tag{3.15}$$

This function transforms not only the output of one node, but uses all nodes as input to return a single number for every node in the layer. Its main advantages are that

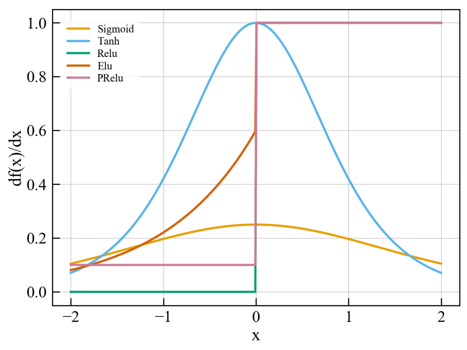
1. the output lies in the interval [0, 1] and
2. the sum over all outputs is equal to one.

These properties allow for the output to be interpreted as a probability mass function. Therefore, the value in the ith output node can be interpreted as the probability of the instance belonging to class i. Additionally, the output of the *softmax* in the last layer of the network is especially well-suited for the cross entropy loss. The combination of those two was used during training of all feed-forward neural networks.

As mentioned earlier, the backpropagation is a successive use of the chain rule of derivatives and hence it is only natural in neural network modelling that the derivative of the activation function is not less important than the activation itself [4]. As the chain rule dictates to multiply the derivatives of functions, the advantage of the *tanh* function over the *sigmoid* becomes obvious. The derivative of the *sigmoid* drops to small values quickly which slows down the learning process of the network.
The advantage of the rectified linear units is twofold. First its derivative for $x > 0$ is always unity, resulting in no disadvantageous effects in the chain rule [19]. Second, it is much faster to compute than any of the other activation functions, speeding up deep networks, especially during training.



(a) Activation functions



(b) And their derivatives

Figure 3.3: Functional form of the most common activation functions and their derivatives. Note that all activations belonging to the "relu-family" are identical in the positive half of the abscissa

3.5 Preventing overfitting

In section 2.3.1, the problem of overfitting was briefly summarized. As neural networks easily contain hundreds of thousands or even millions of parameters, while training on approximately the same amount of instances, methods to prevent overfitting need to be implemented [13, 18]. This helps the network to generalize to unseen data.

3.5.1 Regularization

This method is not exclusive to the field of neural networks but can be used in any algorithm where the goal is the minimization of a loss function.

$$L' = L(\vec{y}_i, \vec{o}_i) + \lambda L_{reg} = L(\vec{y}_i, \vec{o}_i) + \begin{cases} \lambda \sum_i |\beta_i| & \text{LASSO[5]} \\ \lambda \sum_i \beta_i^2 & \text{Ridge[29]} \end{cases} , \quad (3.16)$$

where β are the coefficients of the model, corresponding to the weights and biases in the neural network and λ is the regularization parameter. Hence the regularization term penalizes a higher value for the weights and biases. This, in return, tends to lower the variance of the model. In applications the Ridge-regularization often turns out to be more restrictive than LASSO, setting more parameters to zero. The latter is a more continuous function of the regularization parameter λ. Often a mixture of these two is implemented, called Elastic Net [8].

3.5.2 Dropout

Dropout is a technique exclusively used in neural network modelling [23]. As the name suggests, it involves the random de-activation of several nodes

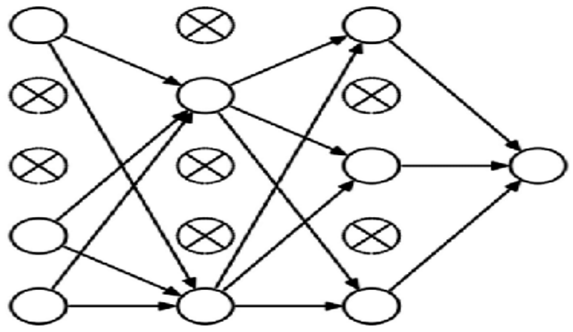


Figure 3.4: Applying dropout to a layer of a neural network [39].

during one training epoch, as schematically seen in figure 3.4.

The goal is to make the nodes of the network more independent from each other. One can imagine this process as training a different neural network at each epoch which are then combined into a final result. Dropout usually slows down convergence, but improves generalisation [27].

4 Results

4.1 Training

During training a lot of different models and hyperparameters were used in order to find the optimal configuration to classify the data and gain insights into the problem. As there are a lot of training instances (much more than most computers' RAM can handle), some of them could easily be neglected such that there is an equal number of signal and background instances in the training set. Thus, the problem of class imbalance, meaning that an algorithm has a good accuracy if classifying all instances in the most prominent class, was evaded. This would render the accuracy an ineffective evaluation metric. Most networks were then trained on 200'000 instances from each signal and background. A sign in support of the assumption, that this is enough for the network to learn all significant relations, is shown in figure A.3 in the appendix.

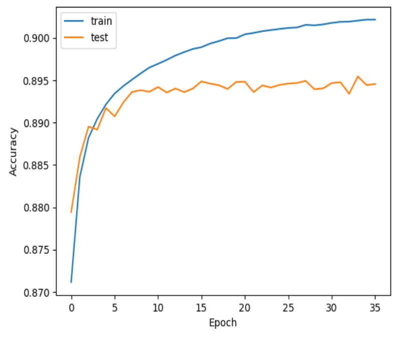
Table 6: Incomplete list of the results for seven neural networks using different structures and features. The last network uses the calculated isolation of the muons. The first column refers to the network structure, each element representing the number of nodes in a hidden layer, the second refers to the used activation function, the third indicating the Dropout percentage per layer, the fourth the number of features and the last indicating if Principal Components Analysis was used.

#Nodes	Act.	Drop[%]	#Features	PCA	Accuracy[%]
[100, 100, 50]	Elu	0	1057	False	90.13
[100, 100, 50]	Elu	0	354	True	89.52
[750, 500, 500]	Elu	0.5	354	True	89.10
[750, 500, 500]	L. Relu	0	354	True	89.10
[100, 100, 50]	Elu	0	71	True	89.70
[750, 500, 500]	tanh	0	1057	False	88.41
[6, 6, 5, 5, 4, 4, 3, 3, 2]	Elu	0	6	False	97.61

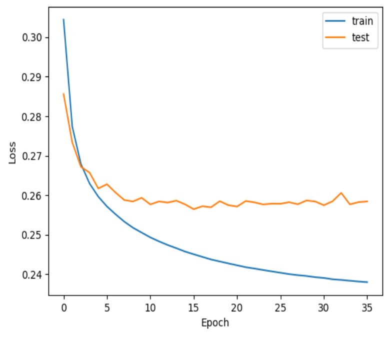
As mentioned in chapter 2, various features are given in redundant form and should not be passed together, for example the equivalent set of coordinates (p_x, p_y, p_z) in Cartesian space and (p_T, η, ϕ) in spherical coordinates. Different combinations of features were tried, but as it turned out the best combination is composed of the features described in table 1, from now on called transverse features. Those, and the isolation baseline features in 2, are the only ones for which the results are shown in the main text. Many different networks were then trained in order to find the best possible constellation of parameters and hyperparameters for these feature sets. The structure of some of the best is summarized in table 6. A more complete table is shown in the appendix in table 8.

These networks are then compared using the validation data and the final model is then applied to the real world data in order to compare it to other methods like boosted decision trees and the χ^2 template fit.

After analysing and fine tuning the neural network parameters it was found that a certain accuracy ($\sim 85\%$) can be obtained with nearly any reasonable network structure meaning there is an obvious pattern in the data which can easily be learned. But to obtain an accuracy 5% better than this baseline is still an important task as this reduces the error by 33% (only having 10 instead of 15 misclassification in 100 examples).



(a) Development of the accuracy.



(b) Development of the loss.

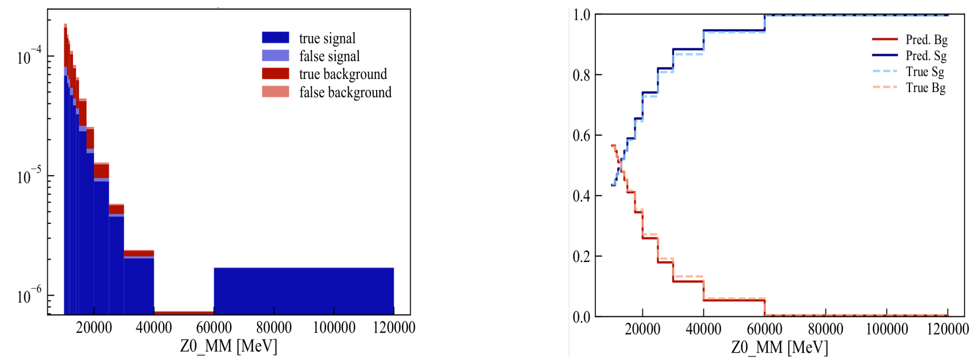
Figure 4.1: Development of some important evaluation metrics on training and validation data for one of the trained neural networks.

The training progress of one of these networks is shown in figure 4.1. As expected, the training accuracy increases steadily while the training loss decreases. However, the validation loss and accuracy flatten out faster than for the training set, which means that the network practically stops generalizing to unseen data but rather finds a function which fits the noise, showing the effect of overfitting.

This result was obtained at the end of this study, after fine tuning most of the hyperparameters and regularizations, hence the early stopping of the network after about 35 epochs.
As table 6 shows, many network structures give a rather good accuracy. The last network in the table has by far the largest accuracy with the least amount of features. The explanation is that during training of this network the information about the isolation was used which served as a baseline comparison.

4.2 Transverse features

In figures 4.2 and 4.3 the results of the best neural net using transverse features are shown. The neural network is able to classify most of the examples under the Z^0 peak as signal, whereas it is more difficult to classify events in the low energy region. The same behaviour is observed for other algorithms as well.



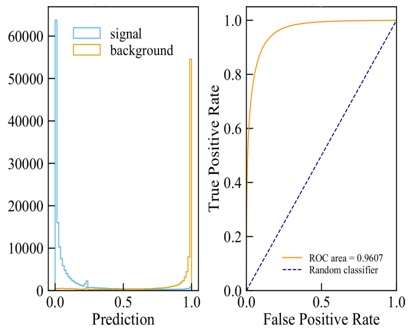
(a) Density prediction per mass bin using transverse features.

(b) Predicted signal/background fraction per mass bin.

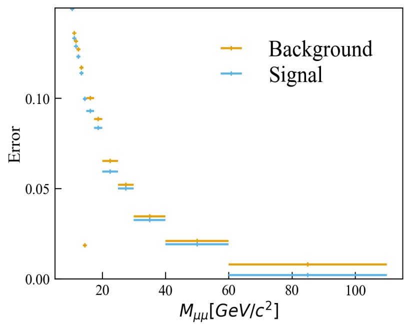
Figure 4.2: Prediction on the validation data using the best neural network with the transverse features.

In figure 4.3a the separation of background and signal in the final node is shown. A perfect network trained on ideal data would yield a plot with two isolated peaks, one for the background being at one and the other for signal at zero. The ROC [11] is shown in the same figure on the right and shows the true positive rate as a function of the false positive rate. The dotted blue diagonal shows the baseline of a network which just classifies by random guessing.
The error per class was estimated by dividing the number of falsely classified

examples by the total number of examples predicted in this class. For the estimation of the signal error, for example, the number of events in one of the light blue bins of figure 4.2a is divided by the number of events in the corresponding dark and light blue bins.



(a) Predicted class value (0=background, 1=signal).



(b) Error on prediction for the validation set

Figure 4.3: Separation power and error of a trained network using the transverse features.

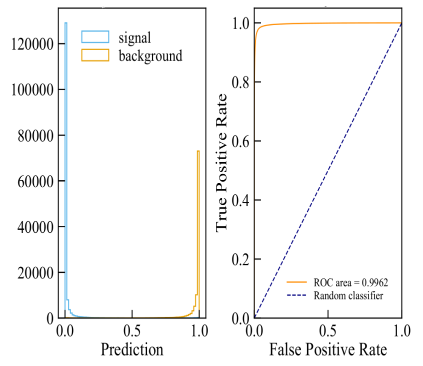
As expected, it is very low in the high mass bins under the Z^0 peak and increases to the lower mass ranges where the background is dominant.

4.3 Isolation features

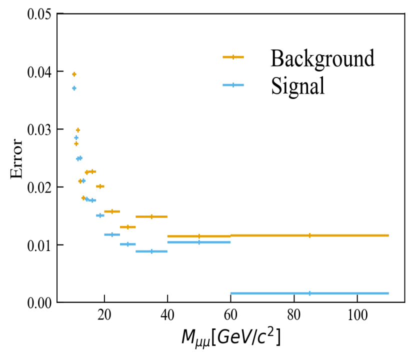
The same analysis as before was done by using the six isolation features, including the transverse momenta and rapidity for both signal muons as well as the maximum value of the isolation and impact parameter for either of the muons. This resulted in figure 4.4.

This result shows a better separation compared to figure 4.3a using the transverse features. Also the ROC-AUC is almost at its maximal value of one.

One should also keep in mind that training on the isolation features allows for a much smaller network (compare 6 input features to 1057), resulting in a much lower training time as well as a much lower consumption of precious RAM. The isolation network could easily be trained on any PC while training on the transverse features almost certainly requires the usage of a computing cluster or a cloud, providing at least 30GB of RAM.



(a) Predicted class value (0=background, 1=signal).



(b) Error on prediction for the validation set

Figure 4.4: Prediction on dataset using isolation features.

4.4 Comparison

The results of the main algorithm used in this thesis, being neural networks, was finally compared to other (machine learning) algorithms. First to boosted decision trees as implemented by the popular python library XGBoost v0.70. The second is a χ^2 template fit using the isolation variable.

4.4.1 Boosted decision trees

The boosted decision tree algorithm implements a specialised variant of the classical decision tree [20]. Its main application is in constructing several dozens or hundreds of smaller decision trees, each time giving a higher weight to misclassified examples. The overall prediction is obtained by taking a weighted average over all outputs from the trees.

BDTs are highly popular in modern data science as often they are computationally very efficient and highly flexible. Another advantage is, that the importance of each feature can easily be estimated, see figure A.6 in the appendix. The optimal neural network using transverse features obtained in this study was then compared to the BDTs, increasing the number of features from one to 1057. The result is shown in figure 4.5. One can see that the neural network is ahead of the BDTs for any number of features. However, it should be noted that the BDT algorithm was not optimized via a grid search of hyperparameters and only the default values of XGBoost v0.70 were used [25].

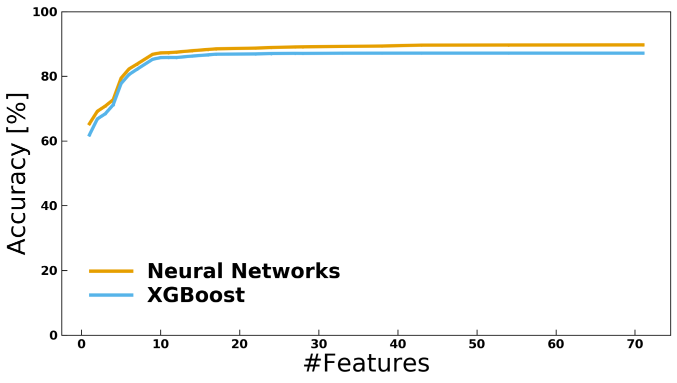


Figure 4.5: Comparison to the boosted decision trees using XGBoost v0.70.

4.4.2 χ^2 template fit

This technique is taken as baseline and compared to the best neural network using the transverse and isolation features. The differences of the results are shown in figure 4.6 and figure 4.7. The absolute results are shown in figures A.4 and A.5 in the appendix.

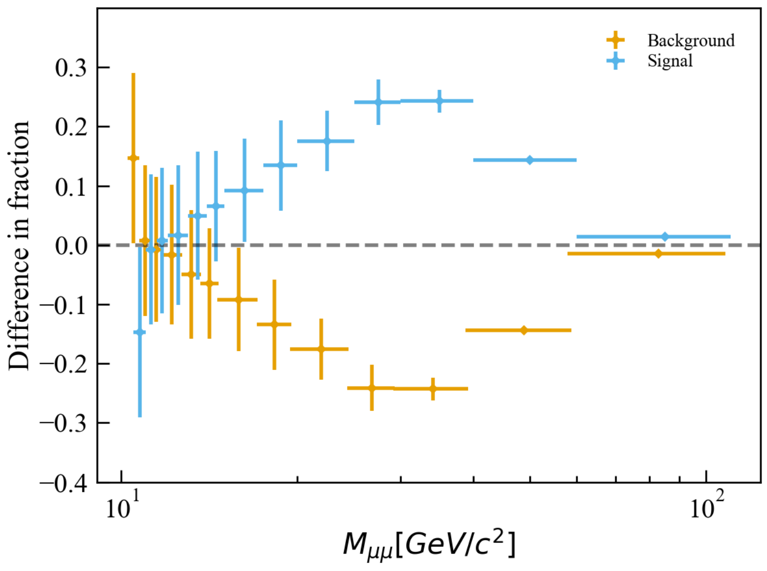


Figure 4.6: Difference of the predicted fraction per mass bin of the χ^2 template fit and the best neural network using the transverse features. The background difference is shifted slightly to the left for better visibility.

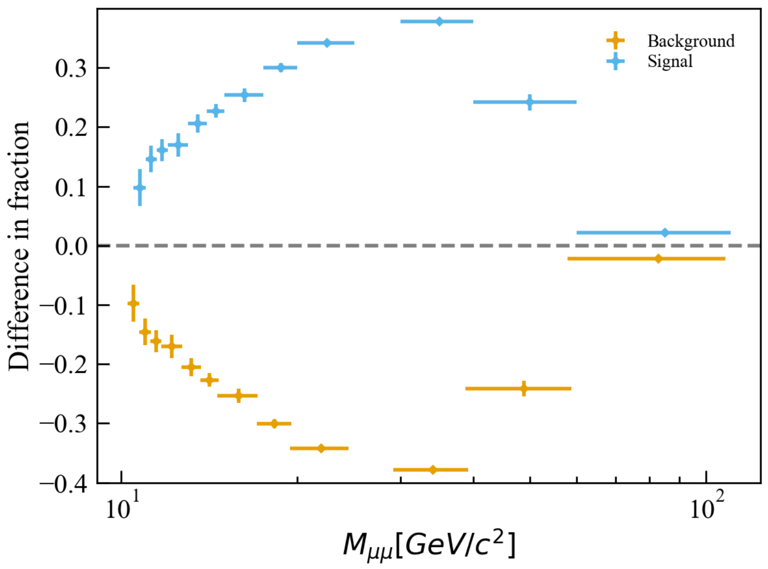


Figure 4.7: Difference of the predicted fraction per mass bin of the χ^2 template fit and the best neural network using the isolation features. The background difference is shifted slightly to the left for better visibility. The errors are too small to be drawn for most mass bins.

The errors on the difference were propagated using

$$\sigma_\Delta = \sqrt{\sigma_1^2 + \sigma_2^2 - 2 \cdot \rho\sigma_1\sigma_2}, \tag{4.1}$$

where σ_1 is the error on the χ^2 fit, σ_2 is the error using the neural networks and ρ is their correlation coefficient.
The errors on both techniques are not uncorrelated as they were performed on the same dataset. So for the new error the correlation was set $\rho = 1$ resulting in

$$\sigma_\Delta = |\sigma_1 - \sigma_2|\,, \tag{4.2}$$

The different methods result in a comparable estimation of the signal fraction in the highest mass bins, where the signal is the dominant contribution. However, in the lower mass bins the methods yield vastly different predictions. One source of this discrepancy might be the absence of other background sources in the training set. In this thesis the network was only trained to discriminate between heavy flavour background and Drell-Yan signal, whereas there is of course every sort of background in the data sample

used for final comparison with the χ^2 fit.
Another reason for the different results could be, that the Monte Carlo sample does not sufficiently resemble reality. Hence when the network is exposed to true signals in the data it might not recognize it due to significant differences in the training and evaluation data. It should be noted that all common algorithms have trouble classifying events in the low energy, background dominated mass bins.

4.4.3 Systematic uncertainties

It was mentioned in section 2 that also "magnet-down" data and simulation with opposite magnet polarity are available. The difference in the predicted fraction is shown in figure 4.8. This provides a useful check of consistency and indicates that both datasets can interchangeably be used during training, providing a much larger dataset.

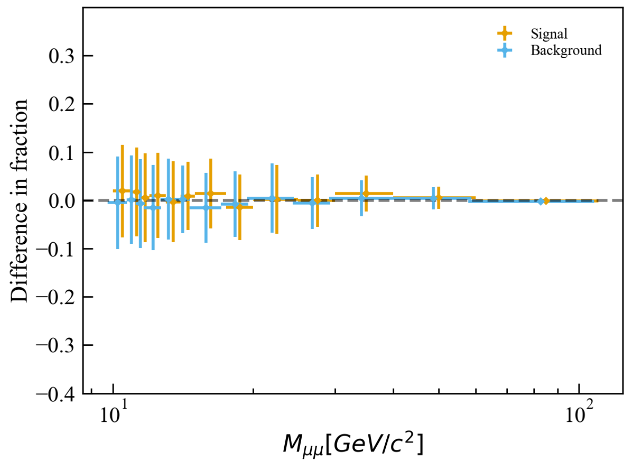


Figure 4.8: Difference in "magnet-up" and "magnet-down" polarization prediction. The background fraction is shifted to the left in x-direction for a clearer representation. One can see that the difference is well below 5% in all mass bins.

For the propagation of the error equation (4.1) was used, setting the correlation $\rho = 0$ as the two datasets are statistically uncorrelated. This results in

$$\sigma_\Delta = \sqrt{\sigma_1^2 + \sigma_2^2}. \tag{4.3}$$

5 Discussion

This thesis could successfully show that neural networks are an appropriate tool for the task of separating heavy-flavour background from Drell-Yan signal $q\bar{q} \to \mu^+\mu^-$. Feed-forward neural networks can obtain an overall separation accuracy of about 90%, easily more in the high mass bins under the Z^0 peak.
It can be expected that networks perform equally well on other background contributions given enough data and time for calibrating the model parameters. Therefore, neural networks, including more sophisticated models like recurrent and convolutional neural networks, should be investigated to obtain even better results. They provide a good alternative to methods for which a more sophisticated data preparation and phenomenological understanding of the data is necessary. The difficult task is to obtain a good training sample for all background and signal sources, which may be a hard task. However, once they are available in high quality, neural networks are a very efficient way of performing the task of classification, even fast enough to be used in trigger decisions.

6 Outlook

Following the success of the usage of the simple feed forward neural network different neural networks can be employed. Further interesting steps would include

1. the use of more sophisticated neural networks, for example convolutional neural networks after applying some transformation to the data ideally generating meaningful pictures.
2. Combining several different machine learning techniques into one estimator, called ensemble method.
3. classifying all different sources of background or applying the technique to another process.

If one or more of these tasks can be successfully solved they provide a new way to deal with the high statistics at the LHC, which will be even more important after finishing the high luminosity upgrade.

Acknowledgements

First and foremost I want to thank Prof. Nicola Serra for providing me with support and expert knowledge throughout this thesis. I also want to thank Andreas Weiden, who supervised me during the whole process and provided useful insights, whenever a problem arose.
I also want to thank the Physics Department of the University of Zurich providing a great infrastructure of computing power which I could use during my Bachelor thesis.
Lastly I want to express my gratitude towards Jonas Eschle who helped me through useful discussions, guidance and outstanding patience while teaching me the basics of machine learning and statistical analysis.

A Appendix

A.1 Autoencoder

Another different technique, also belonging to the class of neural networks, is an unsupervised learning algorithm called autoencoder. Unsupervised learning means, that no labels are needed in order for the network to learn. The general structure of an autoencoder is shown in fig A.1 [12]. It consists of an encoder part, normally reducing the number of dimensions and a decoder part which maps the low dimensional representation back to the input. This is then considered a regression problem in contrast to the previous classification task.

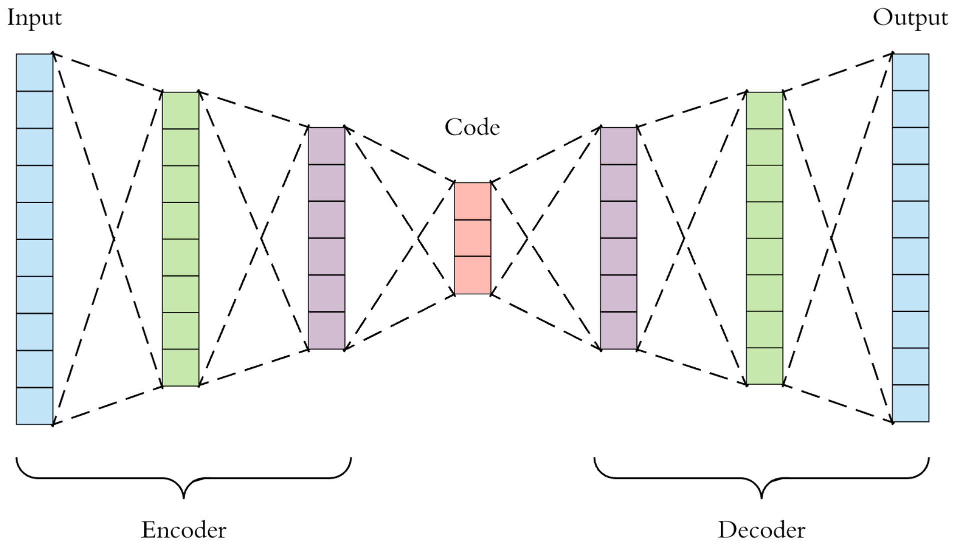


Figure A.1: Schematic view of an autoencoder, trying to map the input onto itself after reducing the number of dimensions [40].

Therefore an appropriate regression loss, like mean squared error (mse) or mean absolute error (mae) has to be used. Additionally, the data was not only standardized, but also normalized such that all its values lie in the interval [-1, 1] with $\tanh(\cdot)$ activation functions.
In an attempt to find new patterns in the data, the number of input dimensions was subsequently reduced to only two dimensions such that the data can easily be visualized. The solution for signal and background is shown in figure A.2.

Some patterns are visible along certain vertical and horizontal lines

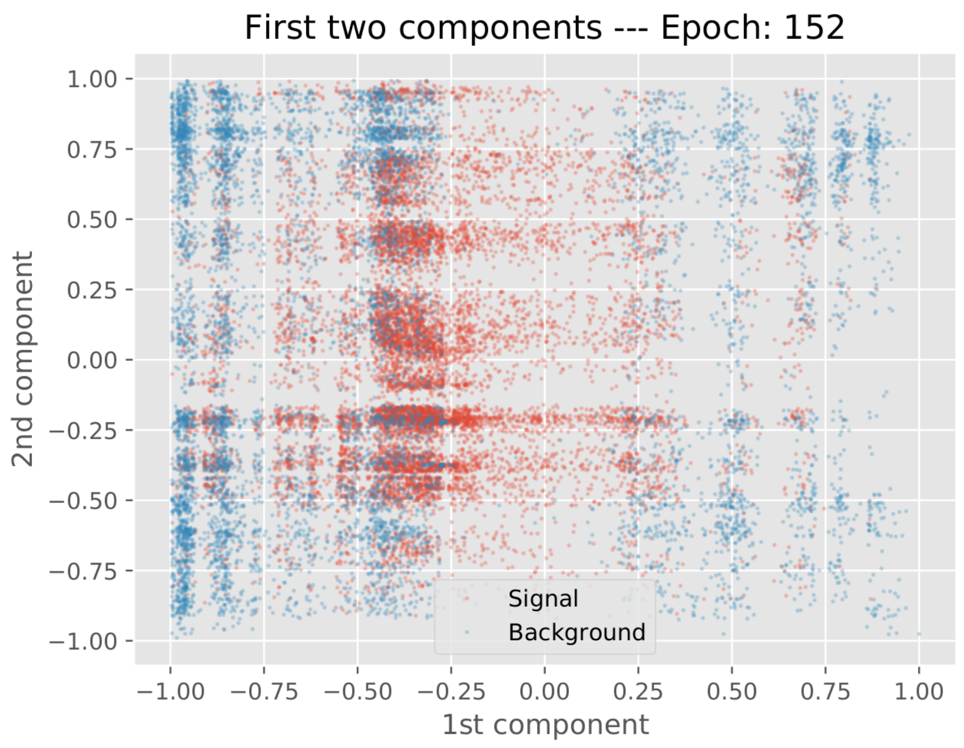


Figure A.2: Two dimensional representation of the validation data of the transverse features after training.

whereas the central region of the plot is populated rather by signal than background. Unfortunately, the interpretation of the lower dimensional data is difficult and there exists no unique method to make sense of these patterns. As the reconstructed mass of the di-muon system was believed to be one of the major function to be reconstructed by a network the patterns were split along those lines and the reconstructed mass plotted in a histogram in an attempt to show that the network groups events with similar mass together. Unfortunately this turned out not to be true and further research would need to be done to make sense of these patterns.

A.2 Applied cuts

Table 7: Performed cuts on simulation and background sample for upward magnet polarization. Very similar results to the low % level are obtained for the downward polarization.

Monte Carlo			
Cut	Instances	Exclusion	%
—	1998901	—	—
Z0_MM <120k	1998901	0	0.00
Z0_MM >10k	1967305	31596	1.58
nSPDHits <600	1958344	40557	2.03
nTrack <150	1958344	40557	2.03
Y <4.5	1958344	40557	2.03
Y >2	1958326	40575	2.03
minus_CHI2 >0.001	1956737	42164	2.11
plus_CHI2 >0.001	1955098	43803	2.19
minus_MINIP <1000	1954955	43946	2.20
plus_MINIP <1000	1954845	44056	2.20

Heavy Flavour background			
Cut	Instances	Exclusion	%
—	2418377	—	—
Z0_MM <120k	2418377	0	0.00
Z0_MM >10k	1677998	740379	30.61
nSPDHits <600	1351201	1067176	44.13
nTrack <150	1351099	1067278	44.13
Y <4.5	1351088	1067289	44.13
Y >2	1336099	1082278	44.75
minus_CHI2 >0.001	1335015	1083362	44.80
plus_CHI2 >0.001	1334144	1084233	44.83
minus_MINIP <1000	1333960	1084417	44.84
plus_MINIP <1000	1333808	1084569	44.85

A.3 Number of instances

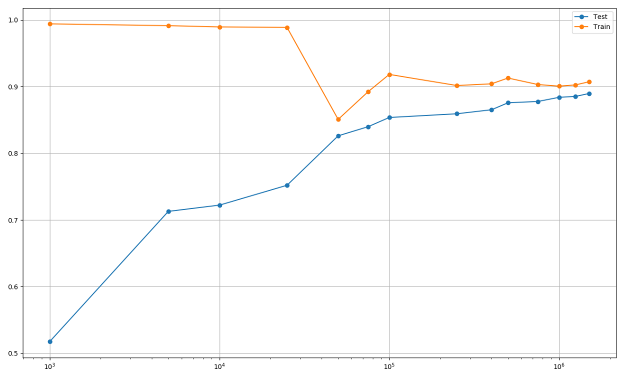


Figure A.3: Accuracy vs. number of instances used during training. The accuracy on the training set is close to 100% for very few instances as the network is complex enough to describe this low number of features (i.e. overfitting). The learning curve saturates at around 100'000 examples. Therefore it was reasonable to assume that 400'000 instances, 200'000 from each background and signal, are enough.

A.4 Absolute results for χ^2 template fit

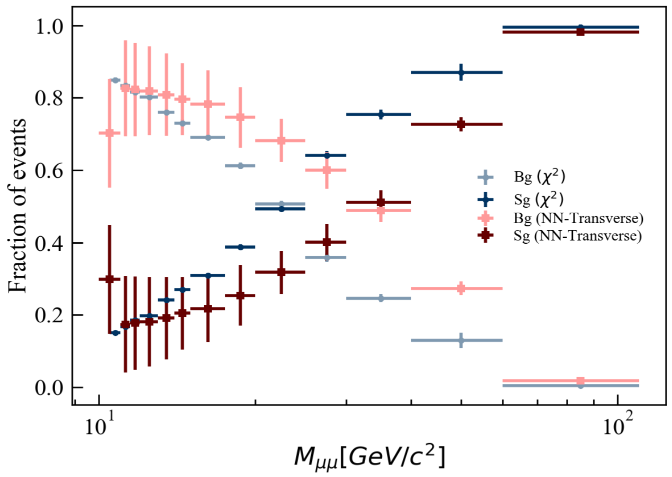


Figure A.4: Comparison in absolute fraction of the χ^2 template fit in the best neural network using the transverse features.

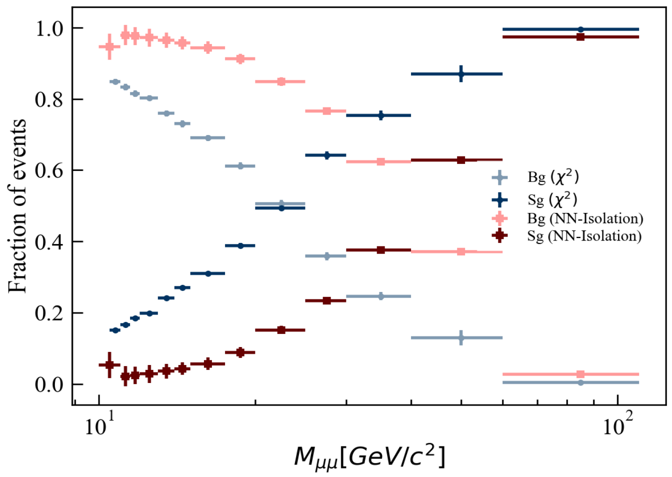


Figure A.5: Comparison in absolute fraction of the χ^2 template fit in the best neural network using the isolation features.

A.5 More complete list of results

#Nodes	Act.	Drop[%]	#Features	PCA	Accuracy[%]
[100, 100, 50]	elu	0	1057	False	90.13
[100, 100, 50]	elu	0	354	True	89.52
[20, 20, 10]	elu	0	354	True	89.38
[100, 100, 50]	elu	0.5	354	True	89.41
[100, 100, 50]	elu	0	354	True	89.5
[750, 500, 500]	elu	0.5	354	True	89.1
[750, 500, 500]	selu	0.5	354	True	89.18
[750, 500, 500]	elu	0	354	True	89.21
[750, 500, 500]	selu	0	354	True	89.19
[750, 500, 500]	elu	0	354	True	89.19
[750, 500, 500]	leaky	0	354	True	89.1
[750, 500, 500]	leaky	0.5	354	True	89.11
[750, 500, 500]	selu	0	1057	False	88.9
[750, 500, 500]	tanh	0	1057	False	88.41
[20, 20, 10]	tanh	0	1057	False	88.46
[20, 20, 10]	tanh	0	354	True	88.42
[750, 500, 500]	tanh	0.5	354	True	88.21
[750, 500, 500]	tanh	0	354	True	88.2
[20, 20, 10]	tanh	0.5	354	True	88.24
[10, 5, 5]	tanh	0	354	True	88.21
[750, 500, 500]	elu	0	1057	False	88.83
[10, 10, 10]	tanh	0	354	True	88.14
[50, 20, 10, 5]	tanh	0	354	True	88.04
[100, 100, 50]	tanh	0	1057	False	87.82
[100, 100, 50]	tanh	0	354	True	87.72
[100, 100, 50]	tanh	0.5	354	True	87.67
[750, 500, 500]	leaky	0	1057	False	87.61
[100, 100, 50]	elu	0	71	False	89.72
[100, 100, 50]	elu	0	54	False	89.7
[100, 100, 50]	elu	0	43	False	89.66
[100, 100, 50]	elu	0	38	False	89.32
[100, 100, 50]	elu	0	33	False	89.19
[100, 100, 50]	elu	0	28	False	89.1
[100, 100, 50]	elu	0	27	False	89.07
[100, 100, 50]	elu	0	24	False	88.9

[100, 100, 50]	elu	0.2	22	False	88.71
[100, 100, 50]	elu	0.3	17	False	88.46
[750, 500, 500]	tanh	0	1057	False	87.69
[750, 500, 500]	tanh	0	1057	False	87.91
[750, 500, 500]	tanh	0.5	354	True	87.21
[750, 500, 500]	tanh	0	1057	False	87.43
[750, 500, 500]	tanh	0	354	True	85.83
[1500, 750, 750, 500, 500]	tanh	0	1057	False	86.75
[750, 500, 500]	tanh	0	354	True	86.26
[750, 500, 500]	tanh	0	1057	False	87.61
[750, 500, 500]	tanh	0	1057	False	87.48
[750, 500, 500]	tanh	0.3	354	True	87.56
[1500, 750, 750, 500, 500]	tanh	0.5	1057	False	87.95
[1500, 750, 750, 500, 500]	relu	0.5	1057	False	79.21
[1500, 750, 750, 500, 500]	tanh	0	1057	False	85.86
[750, 500, 500]	relu	0	1057	False	65.54
[1500, 750, 750, 500, 500]	tanh	0	354	True	85.85
[750, 500, 500]	relu	0	1057	False	85.74
[750, 500, 500]	relu	0	354	True	85.15
[1500, 750, 750, 500, 500]	relu	0.5	1057	False	83.86
[1500, 750, 750, 500, 500]	relu	0	354	True	83.96

A.6 Feature importance

Boosted decision trees are able to give a very intuitive estimate of the feature importance by counting the number of a times a feature has been cut on during training of the algorithm. The better a feature is for separating background and signal the more often it is used to split the data.

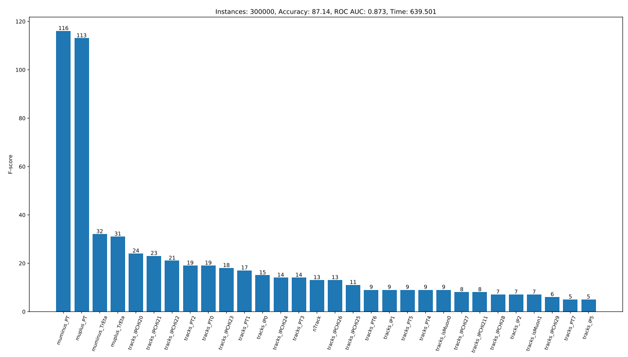


Figure A.6: Feature importance as obtained by XGBoost v0.70.

The most important feature (according to XGBoost) are the transverse momenta of the muons, supposedly because they can be used to partly reconstruct the mass of the di-muon system.

References

[1] S.D. Drell, T.M. Yan (1970), *"Massive Lepton-Pair Production in Hadron-Hadron Collisions at High Energies"*, Phys. Rev. Lett. 25, 316, pp. 316-320

[2] I.R. Kenyon (1982), *"The Drell-Yan process "*, Reports on Progress in Physics, V45, I11, pp. 1261-1315

[3] D.E. Rumelhart, G.E. Hinton, R.J. Williams (1986), *"Learning representations by back-propagating errors"*, Nature volume 323, pp. 533–536

[4] H.N. Mhaskar, Micchelli (1993), *"How to Choose an Activation Function"*, Proceedings of the 6th International Conference on Neural Information Processing Systems, pp. 319-326

[5] R. Tibshirani (1996), *"Regression Shrinkage and Selection via the Lasso"*, Journal of the Royal Statistical Society. Series B (Methodological), Vol. 58, No. 1, pp. 267-288

[6] D. Pyle (1999), *"Data Preparation for Data Mining"*, Morgan Kaufmann Publishers, Inc.

[7] L. I. Smith (2002), *"A tutorial on Principal Components Analysis"*, http://www.iro.umontreal.ca/∼pift6080/H09/documents/papers/pca_tutorial.pdf, retrieved March, 2018

[8] H. Zou, T. Hastie (2004), *"Regularization and variable selection via the elastic net"*, R. Statist. Soc. B, Vol. 67, Part 2, pp. 301–320

[9] B. Widrow (2005), *"Thinking About Thinking: The Discovery of the LMS Algorithm"*, IEEE Signal Processing Magazine Vol. 22, Issue 1, pp. 100-106

[10] J. Shlens (2005), *"A Tutorial on Principal Component Analysis"*, Systems Neurobiology Laboratory, University of California at San Diego, vol. 82

[11] T. Fawcett (2006), *"An introduction to ROC analysis"*, Elsevier, Vol 27, Issue 8, pp. 861-874

[12] G. Hinton, R. Salakhutdinov (2006), *"Reducing the dimensionality of data with neural networks"*, Science, pp. 504-507

[13] Y. Bengio (2009), *"Learning Deep Architectures for AI"*, Foundations and Trends in Machine Learning V2 I1, pp. 1-127

[14] X. Glorot, Y. Bengio (2010), *"Understanding the difficulty of training deep feedforward neural networks"*, Proceedings of Machine Learning Research, vol. 9, pp. 249-256

[15] K. K. Dobbin, R. M. Simon (2011), *"Optimally splitting cases for training and testing high dimensional classifiers"*, BMC Medical Genomics

[16] ATLAS Collaboration (2012), *"Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC"*, arXiv:1207.7214, Phys.Lett. B716, pp. 1-29

[17] CMS Collaboration (2012), *"Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC"*, arXiv:1207.7235, Phys. Lett. B 716, pp. 30

[18] J. Bergstra, Y. Bengio (2012), *"Random Search for Hyper-Parameter Optimization"*, JMLR, pp. 281-305

[19] R. Pascanu, T. Mikolov, Y. Bengio (2013), *"On the difficulty of training Recurrent Neural Networks"*, arXiv:1211.5063

[20] R.E. Shapire (2013), *"Explaining AdaBoost"*, Empirical Inference, pp. 37-52

[21] LHCb Collaboration (2014), *"LHCb detector performance"*, arXiv:1412.6352

[22] N. Chiapolini, U. Straumann, J. Anderson, K. Müller (2014), *"Low-Mass Drell-Yan Cross-Section Measurements with the LHCb Experiment"*

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014), *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"*, JMLR 15

[24] D. Kingma, J. Ba (2014), *"Adam : A method for stochastic optimization"*, arXiv:1412.6980

[25] T. Chen, C. Guestrin (2016), *"XGBoost: A Scalable Tree Boosting System"*, arXiv:1603.02754

[26] P. Ramachandran, B. Zoph, Q. V. Le (2017), *"Searching for Activation Functions"*, arXiv:1710.05941

[27] M. Nielsen (2017), *"Neural Networks and Deep Learning"*, http://neuralnetworksanddeeplearning.com/index.html

[28] K. Janocha, W.M. Czarnecki (2017), *"On Loss Functions for Deep Neural Networks in Classification"*, arXiv:1702.05659

[29] L.E. Melkumova, S.Y. Shatskikh (2017), *"Comparing Ridge and LASSO estimators for data analysis"*, Elsevier, Vol. 201, Pages 746-755

[30] A. Gehrmann-De Ridder (2018), *"The Standard Model of Electroweak Interactions"*, ETH, p. 49

Figures

[31] Z. Wan, J. Conway, A. Anastassov, F. Ratnikov, (2004), *"High Mass Di-Tau Analysis and Search for Z' "*, https://www-cdf.fnal.gov/physics/exotic/r2a/20041014.ditau_zprime/

[32] K. M. Fauske (2006), *"Example: Neural network"*, Texample.net

[33] M. Vretenar, G. Bellodi, R. Garoby, F. Gerigk, K. Hanke, A. M. Lombardi, S. Maury, M. Pasini, C. Rossi, E. Z. Sargsyan (2007), *"LINEAR ACCELERATOR DESIGNS FOR THE UPGRADE OF THE CERN PROTON INJECTOR COMPLEX (LINAC4, SPL)"*, Design Study

[34] D. Fehling (2008), *"The Standard Model of Particle Physics: A Lunchbox's Guide"*, The Johns Hopkins University

[35] B. Schmidt (2009), *"The LHCb detector-Global Status"*, LHCb Collaboration

[36] A. Bhande (2011), *"What is underfitting and overfitting in machine learning and how to deal with it."*, Medium

[37] L. A. Harland-Lang, A. D. Martin, P. Motylinski, R.S. Thorne, (2014), *"Parton distributions in the LHC era: MMHT 2014 PDFs"*, arXiv:1412.3989

[38] S. Farid, (2016), *"Search for New Physics Beyond the Standard Model"*, PhD Thesis

[39] A. Budhiraja (2016), *"Dropout in (Deep) Machine learning"*, Medium

[40] A. Dertat (2017), *"Applied Deep Learning - Part 3: Autoencoders"*, Towards Data Science

[41] S. Varma, S. Das (2018), *"Training Neural Networks"*, DeepLearning