

**Hyperparameter Optimization of a Neural
Network using Lorentz invariant Variables in the
 tZq Channel at 13 TeV with the ATLAS
Experiment**

Richard Baumann

Bachelorarbeit in Physik
angefertigt im Physikalischen Institut

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

Februar 2020

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate kenntlich gemacht habe.

Bonn,
Datum

.....
Unterschrift

1. Gutachter: Prof. Dr. Ian C. Brock
2. Gutachter: Dr. Eckhard von Törne

Acknowledgements

An dieser Stelle möchte mich bei meiner Familie bedanken, die mich während meines bisherigen Studiums unterstützt und mir immer Rückhalt geboten hat. Ich möchte mich außerdem bei meinen Freuden und besonders meiner Freundin bedanken, auf die ich mich während meines Studiums immer verlassen konnte.

Ebenfalls möchte ich an dieser Stelle Ian C. Brock für die Möglichkeit, eine Arbeit in diesem interessanten Gebiet zu verfassen, meinen Dank aussprechen. Auch den Mitgliedern der Arbeitsgruppe gebührt mein Dank, weil sie mich vom ersten Tag an freundlich aufgenommen haben. Die Brettspielabende und Kaffeepausen sowie die allgemeine Diskussions- und Hilfsbereitschaft haben eine entspannte Atmosphäre geschaffen, die zum Gelingen der Arbeit beigetragen hat. Abschließend gehört ein besonderer Dank Christian Kirfel und Federico Capriles für das Korrekturlesen meiner Arbeit.

Contents

1	Introduction	1
2	Particle Physics	3
2.1	Standard Model	3
2.1.1	Bosons	3
2.1.2	Fermions	4
2.2	Four-Vectors	5
2.3	Scalar Products	5
2.4	Lorentz Invariance	6
3	The Atlas Detector	7
3.1	Setup	7
3.1.1	Inner detector	8
3.1.2	Calorimeter	9
3.1.3	Muon spectrometer	10
3.1.4	Detection	10
3.2	Coordinate System	10
3.3	Reconstruction	11
4	Machine learning	13
4.1	Neural Network	13
4.1.1	Function of a singular Node	14
4.1.2	Measurements of Perfomance	15
4.1.3	Backpropagation	17
4.1.4	Improving the Performance	19
5	tZq Process	23
5.1	Signal Regions	23
5.2	Background Processes	24
5.3	Choice of Variables	25
5.3.1	Commonly used Variables	25
5.3.2	Lorentz invariant Variables	26
6	Optimization of the Hyperparameters	29
6.1	Choice of Optimizer	29
6.2	Differences between Signal Regions	32
6.3	Comparison	33

6.4 Outlook	33
7 Conclusion	37
Bibliography	39
List of Figures	41
List of Tables	43
Acronyms	45

Introduction

Processes, that are connected by logic and obey universal laws always fascinated me. While studying physics, many topics sparked my interest, but the most interesting topic was particle physics. The sheer simplicity and beauty of the Standard Model that could explain many phenomena, by only using a set of elementary particles and describing their interactions; it was captivating.

A few years ago, I watched a video that YouTube had recommended to me. In the video, an AI was programmed with the objective to complete the famous first level of Super Mario [1]. I was astonished that the program was able to archive its goal without further input by the user. I already liked programming, therefore the whole concept fascinated me.

Traditionally in particle physics, collision experiments are used in order to create new particles. The four experiments at LHC together produce about 25 Gb/s (gigabyte per second) of data [2]. Due to the large quantity of data produced, a wide range of applications for machine learning are possible and at times necessary. Neural networks are used to distinguish between signal and background events and adversarial neural networks are used to decrease the influence of systematic errors [3]. However, in this work, the impact of different variables on the performance of a Neural Network discriminant is tested. Therefore instead of the commonly used variables, Lorentz invariant variables are used to differentiate between signal and background events in the tZq production process.

The following chapters give brief introductions into the basic topics. In chapter 2, the standard model is explained followed by an overview over the LHC and the ATLAS Detector in chapter 3. Moreover, a closer look is taken on variables and the reconstruction process. Chapter 4 introduces the concept of machine learning, but mainly focuses on neural networks. In chapter 5, the tZq process as well as different sets of variables that can be used to train a neural network are introduced. Finally, the set-up and training process with Lorentz invariant variables of a neural network is described in chapter 6. After optimizing said neural network, a comparison between the training with kinematic variables is made.

Particle Physics

2.1 Standard Model

In the 1950s and 1960s, because of huge improvements of particle accelerators, physics was confronted with a problem. Numerous particles were discovered that seemed to be elementary particles, leading people to call the whole collection a "particle zoo". This contradicted the idea of elementary particles being the smallest components in building matter. An attempt was made by Murray Gell-Mann and Yuval Ne'eman to classify the several hundred resonances and particles by forming multiplets based on their intrinsic properties. This eightfold way is an organizational scheme which classifies mesons and baryons. In addition, it has predicted previously unknown particles such as the Ω baryon. This gave rise to the quark model, which postulated that all baryons and mesons consist of smaller constituents, the quarks.

The Standard Model (SM) is a truly remarkable success of modern physics and the state-of-the-art model used in particle physics. All known elementary particles and most forces are described by it. Even if it is not able to explain some questions, most notable the nature of dark matter, it is the most comprehensive model to date. Most predictions made, based on this model, have been proven correct experimentally in the past decades, most recently the discovery of the Higgs Boson.

The SM describes Bosons and Fermions. Fermions are all matter particles and Bosons are the force carriers that allow Fermions to interact with each other. An overview of the SM is displayed in figure 2.1.

2.1.1 Bosons

The SM describes five different Bosons, which are integer spin particles. Four of them are gauge bosons and the remaining is the Higgs boson. The gauge bosons act as force carrier particles that mediate the different fundamental interactions: electromagnetic, strong and weak forces. The gravitational force is neglected due to its relative strength to the other forces at the energy scale of particle physics.

The electromagnetic force affects all electrically charged particles, and operates on an infinite range. The mediator particle is the photon γ . In contrast to electromagnetism, the weak force is confined to small distances. Most commonly, the weak force is encountered in decay processes such as the β -decay. All fermions of the SM can interact weakly, either via a charged current, using the W^\pm -boson as a mediator; or via a neutral current, using the Z-boson as mediator. The strong force is the strongest

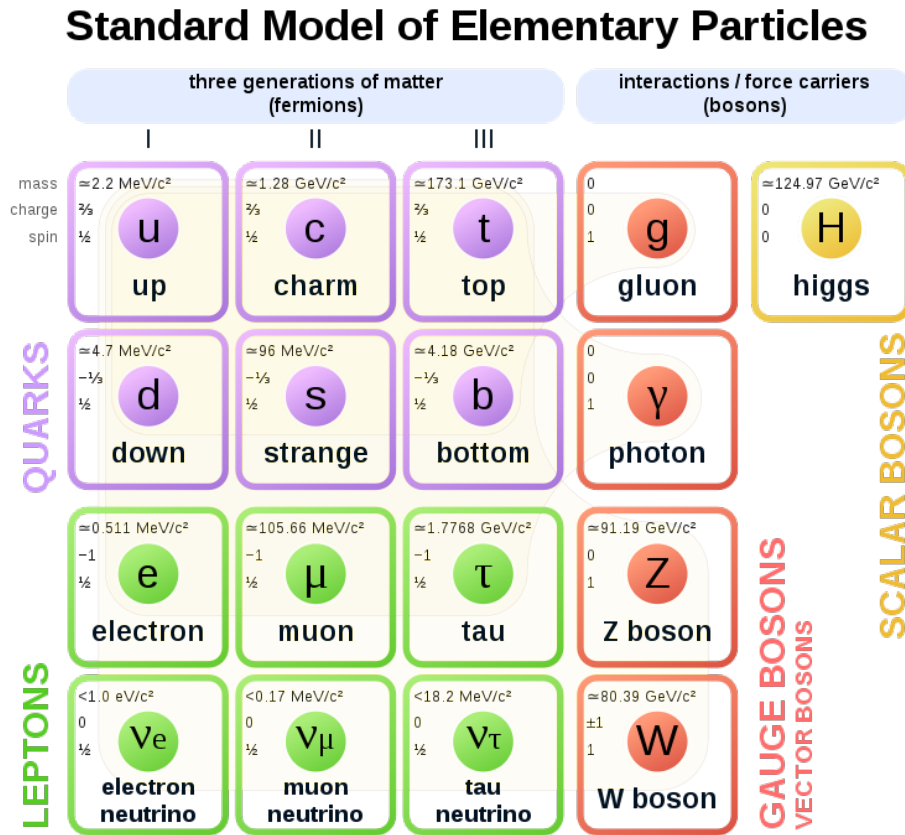


Figure 2.1: Overview of the SM [4]. In the three left columns, the different generations of fermions, are listed in ascending order. On the right of the fermions the bosons are listed.

of the fundamental forces. However, it is also the most confined force with a range of 10^{-15} m. The strong force only applies to color charged particles. The mediator particle of the strong interaction, is the gluon g . It is color charged and therefore able to selfinteract.

2.1.2 Fermions

Fermions, which are half integer spin particles, can be separated into two groups: leptons and quarks. Both are further divided into three generations. The different generations of the leptons consists of the electron e^- and the electron neutrino ν_e (first generation), the muon μ^- and the muon neutrino ν_μ (second generation) and the tau τ^- and tau neutrino ν_τ (third generation). Leptons do not have a color charge and therefore can not interact strongly. Moreover neutrinos do not carry a electric charge in contrast to the other leptones that have a charge of $-1 e$, e being the elementary charge ($e = 1.602176634 \times 10^{-19} \text{ C}$ [5]). Therefore charged leptons can interact via the weak and electromagnetic force. Neutrinos only interact weakly.

Each generation of quarks consists of one up-type quark and one down-type quark. The up-type quarks are the up-quark u (first generation), charm-quarks c (second generation) and top-quark t

(third generation), the down-type quarks are the down-quark d (1st generation), strange-quark s (2nd generation) and bottom-quark b (3rd generation). Up-type quarks have a electric charge of $\frac{2}{3} e$, while down-type quarks have a charge of $-\frac{1}{3} e$. Due to their electric and color charge, quarks can interact either electromagnetically, weakly or strongly.

2.2 Four-Vectors

An important concept for particle physics is the Minkowski space. In addition to the three space coordinates of the Euclidean space, vectors in the Minkowski space have an additional time dimension. Therefore, vectors in the Minkowski space consist of one time-like, which is the zeroth component, and three space-like components, being the first second and third component. Thus, a typical four-vector in covariant form is shown in equation 2.1a. There is also a contravariant form that is displayed in equation 2.1b.

$$a^\mu = \begin{pmatrix} ct \\ x \\ y \\ z \end{pmatrix}, \mu \in [0, 1, 2, 3] \quad (2.1a)$$

$$a_\mu = (ct, x, y, z) \quad (2.1b)$$

An often used 4-vector in particle physics is the four-momentum. The covariant form is presented in equation 2.2, where E is the energy, p_x, p_y, p_z are the momenta in the x-, y- and z-direction and c is the speed of light ($c = 299\,792\,458 \times 10^8$ m/s [5]).

$$p_\mu = \left(\frac{E}{c}, p_x, p_y, p_z \right) \quad (2.2)$$

Natural units are often utilized in particle physics to simplify equations. By defining $c = 1$ the four-momentum from equation 2.2 becomes:

$$p_\mu = (E, p_x, p_y, p_z) \quad (2.3)$$

Another relation, that is often used, is called the energy-momentum relation. Using natural units, the energy-momentum relation is defined as shown in equation 2.4.

$$E^2 = p^2 + m^2 \quad (2.4)$$

2.3 Scalar Products

The scalar product in the Minkowski space can be defined as the matrix shown in equation 2.5a. The scalar product between two arbitrary vectors a and b can be written as is shown in equation 2.5b.

$$\eta(e_\mu, e_\nu) = \eta_{\mu\nu} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad (2.5a)$$

$$\langle a|b \rangle = \eta(a^\mu, b^\nu) = \eta_{\mu\nu} a^\mu b^\nu = a_0 \cdot b_0 - a_1 \cdot b_1 - a_2 \cdot b_2 - a_3 \cdot b_3 = a_0 \cdot b_0 - \vec{a} \cdot \vec{b} \quad (2.5b)$$

2.4 Lorentz Invariance

The ability to describe physical events in different frames of reference is essential for understanding phenomena. In classical physics with reference frames, that are non-relativistic, Galilean transformations can be used. However, these transformations assume an absolute time, defining the identity $t = t'$. In contrast, the theory of relativity shows, that at high speeds time depends on the frame of reference. Therefore, another form of transformation has to be used, which is called Lorentz transformation. variables that describe these transformations are the β and the Lorentz factor γ , which are defined in equation 2.6.

$$\beta = \frac{v}{c} \quad (2.6a)$$

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} = \frac{1}{\sqrt{1 - \beta^2}} \quad (2.6b)$$

An example of a Lorentz transformation is given in equation 2.7a where the observer is boosted along the x-axis with velocity v_x . Equation 2.7b shows the Lorentz transformation in matrix form that can perform the x-boost on a four-vector.

$$\begin{aligned} t' &= \gamma(t - \beta \cdot x) \\ x' &= \gamma(x - v_x \cdot t) \\ y' &= y \\ z' &= z \end{aligned} \quad (2.7a)$$

$$\Lambda_\mu^\nu = \begin{pmatrix} \gamma & -\gamma \cdot \beta & 0 & 0 \\ -\gamma \cdot \beta & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.7b)$$

A quantity that is not changed by a Lorentz transformation is called Lorentz invariant. This quantity has the same attributes in every frame of reference. Examples of Lorentz invariant properties are the speed of light, the mass and the charge.

The Atlas Detector

The Organisation européenne pour la recherche nucléaire (CERN), which translates to the European Organization for Nuclear Research, operates the largest particle physics laboratory in the world [6]. It is located on the French-Swiss border near Geneva, and has been accelerating science for nearly 70 years. Several achievements which are crucial to particle physics have been made [7]. By constantly upgrading the CERN complex, new energy scales are achieved and therefore the understanding of particle physics is broadened.

The Large Hadron Collider (LHC) is the latest particle collider in the CERN complex. With a circumference of nearly 27 km [8], it is the largest particle collider in the world. Protons archive 11 245 revolutions per second [8] when they are accelerated to their maximum velocity which is close to the speed of light. A total of 9 593 magnets make sure that the bunches of protons, which have a total energy of 6.5 TeV, stay on the intended path. With its 13 TeV collision energy, the LHC is designed to test the limits of the standard model. Therefore, 4 detectors at the LHC analyze the manifold particles that emerge from the proton collisions. These experiments are a toroidal LHC apparatus (ATLAS), A Large Ion Collider Experiment (ALICE), Compact Muon Solenoid (CMS) and Large Hadron Collider beauty (LHCb).

Named after the titan that single handedly prevented the sky from crashing into the earth, the ATLAS detector is just as impressive as the LHC. Being one of the two general-purpose particle detectors in the LHC, this gigantic machine is built to detect the smallest constituents of matter. The ATLAS detector weighs about 7 000 tons and measures 46 m in length and 25 m in diameter. This dimensions as well as the constituents of the ATLAS detector are displayed in figure 3.2. The excavation for the construction digged up 300 000 tons of rock [10] and 50 000 tons of concrete [10] were used for the construction.

3.1 Setup

Being an all purpose detector, ATLAS is build to detect a wide range of signals. The ATLAS detector consists of four major components, each crucial to the detection of different particles: the inner detector, the calorimeter and the muon spectrometer. Moreover, a strong magnetic field, which is generated by a system of solenoid magnets, pervades the detector.

CERN's Accelerator Complex

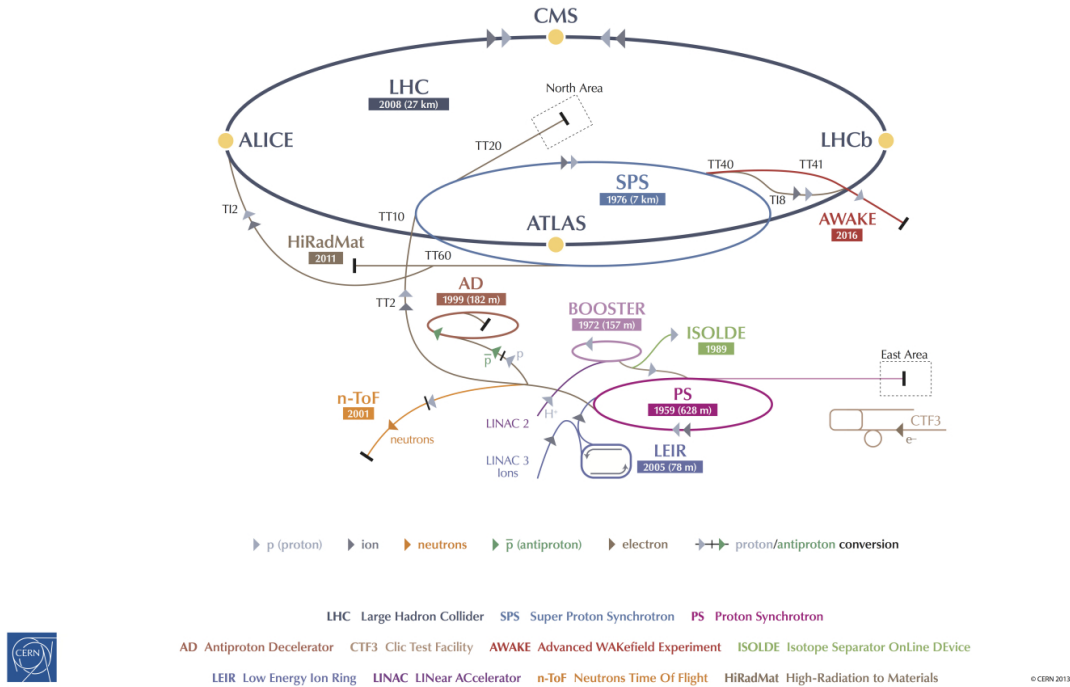


Figure 3.1: Overview of CERNs accelerator complex [9], laying out the general structure of accelerators. Also shown is the LHC and the four experiments, ATLAS, ALICE, CMS and LHCb.

3.1.1 Inner detector

The inner detector in ATLAS consists of three parts, the *pixel detector*, the *Semi Conductor Tracker (SCT)* and the *Transition Radiation Tracker (TRT)*. A general overview is displayed in figure 3.3. The function of this three parts is tracking the trajectory of charged particles. Based on the track and the magnetic field, the charge, direction and momentum of the particle can be determined.

The pixel detector is the component of the ATLAS detector that is the closest to the collision point. It consists of 80 million pixels which are arranged in three concentric layers. The detector’s concept is based on the semiconductivity of silicon which causes ionization of the material when a particle passes through it. The resulting current is picked up by the nearby pixels. The output of multiple pixels in different layers delivers information about the trajectory of the particle.

The SCT is also a silicon based detector, that consists of silicon strip sensors. This sensors are distributed over four cylindrical barrel layers and 18 endcap discs. This setup can is pictured in figure 3.3.

The TRT is an arrangement of straw tube detectors. A charged wire runs though the center of every tube, creating a potential difference between itself and the walls of the tube. Moreover, each tube is filled with xenon and argon, which is ionized when a particle passes through. The resulting ions then either drift towards the charged wire or towards the walls, depending on their charge. By looking at

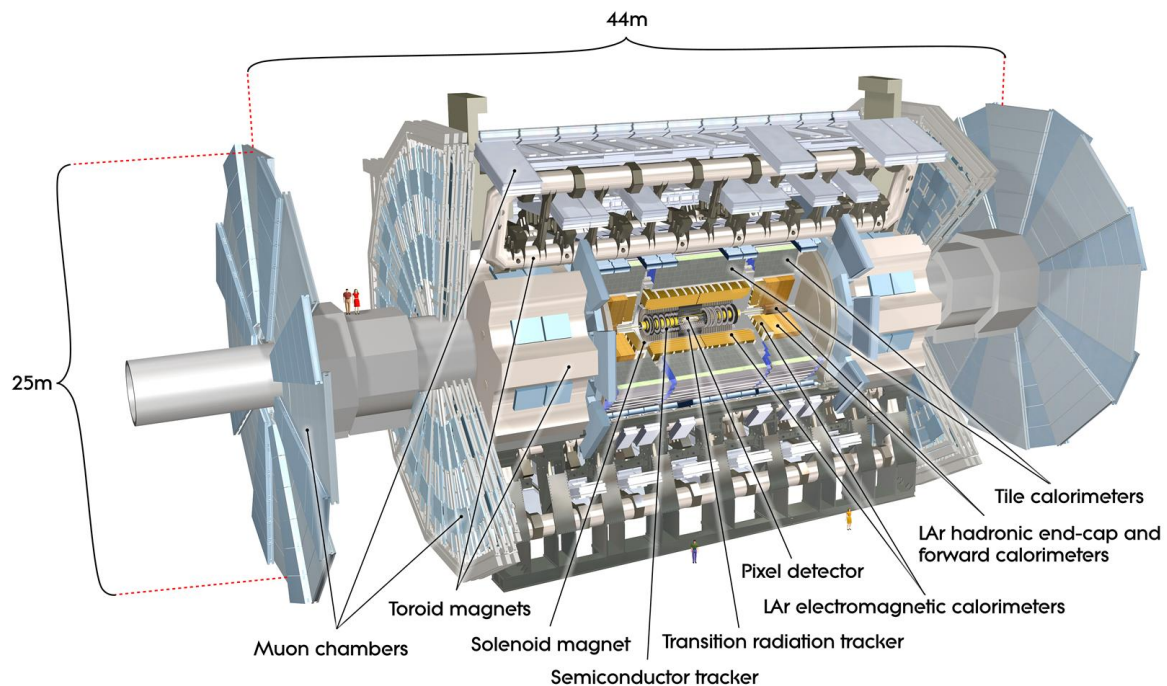


Figure 3.2: Overview of the ATLAS detector, with labels of the components [11].

every straw, that created a signal, the path of a particle can be determined.

3.1.2 Calorimeter

Surrounding the inner detector, the calorimeter measures the energy of particles that are passing through by slowing and eventually absorbing the particle. In general, calorimeters are divided into two parts: the *electromagnetic* and the *hadronic calorimeter*. Both of them are based on the same principle but while the electromagnetic calorimeter uses the electromagnetic force to decrease the energy of a particle, its hadronic counterpart uses the strong force.

Electromagnetic calorimeters detect electromagnetically interacting particles e.g. electrons and photons. These particles lose energy while traveling through matter because of *pair production* and *Bremsstrahlung*. Pair production refers to the process where a photon converts into an electron-positron pair. Electrons and positrons at high energies emit photons, in a process that is called Bremsstrahlung. Through on both of these processes, the energy loss of electrons and photons cause electromagnetic showers. The shower stops when the energy of the particles is insufficient for a further decay.

A Hadronic calorimeter utilizes a similar concept. In this case, hadronic showers are the result of the strong force between the incoming particle and the absorbing material's nuclei as well as ionization. However, a shower only happens if the emerging particles also react with the material's nuclei.

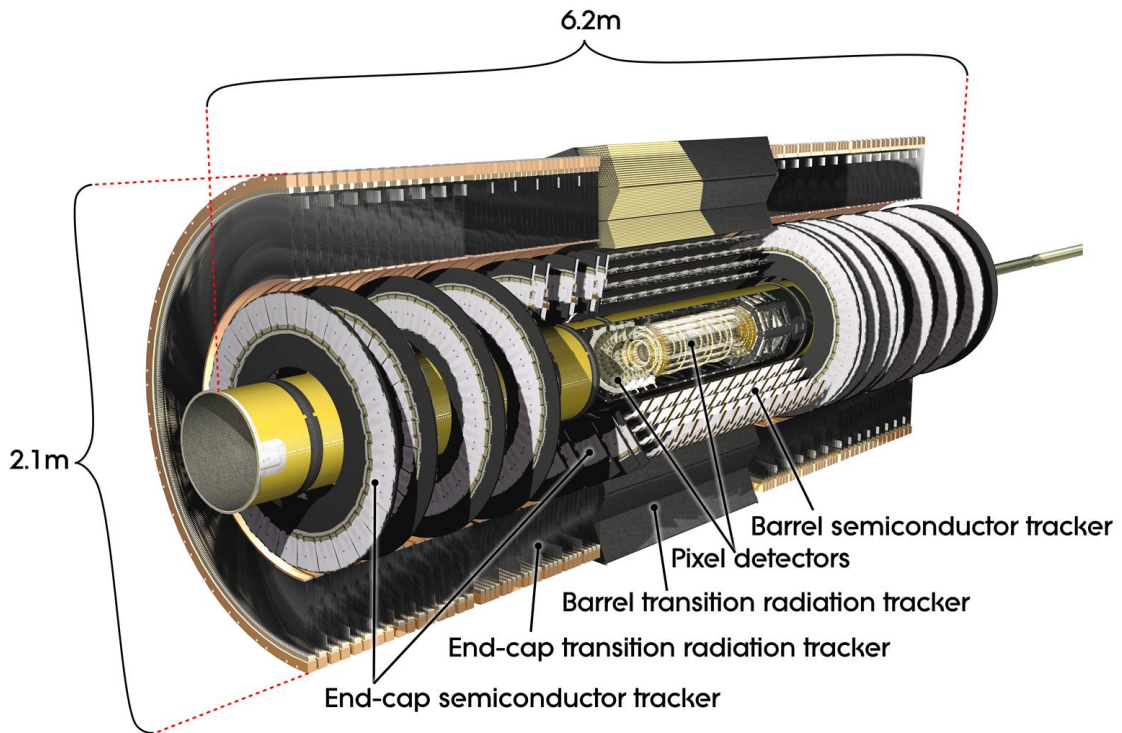


Figure 3.3: Overview of the structure of the inner detector [12].

3.1.3 Muon spectrometer

Muons and neutrinos usually pass through the calorimeters of the ATLAS detector undetected. Muons leave a track in the inner detector, however in order to detect muons, the *muon spectrometer* is necessary. It consists of three toroidal magnets, which generates a strong magnetic field and chambers that measure the track of the particle. The muon spectrometer works similar to the TRT. The toroidal magnets create a magnetic field that deflects incoming muons, which then are tracked in the adjacent chambers. The muon chamber is the largest component of the ATLAS detector.

3.1.4 Detection

Figure 3.4 displays a cross section of the ATLAS detector, as well as tracks left by various particles. As mentioned before, electrons and photons are detected in the electromagnetic calorimeter, while hadrones, exemplified by proton and neutron tracks, are detected in the hadronic calorimeter. Muons are detected in the muon spectrometer, neutrinos do not get detected at all.

3.2 Coordinate System

As displayed in figure 3.2, the ATLAS detector has a cylindrical symmetry. Therefore, an often used set of coordinates are cylindrical coordinates, where the z-axis is along the beam pipe. The x- and

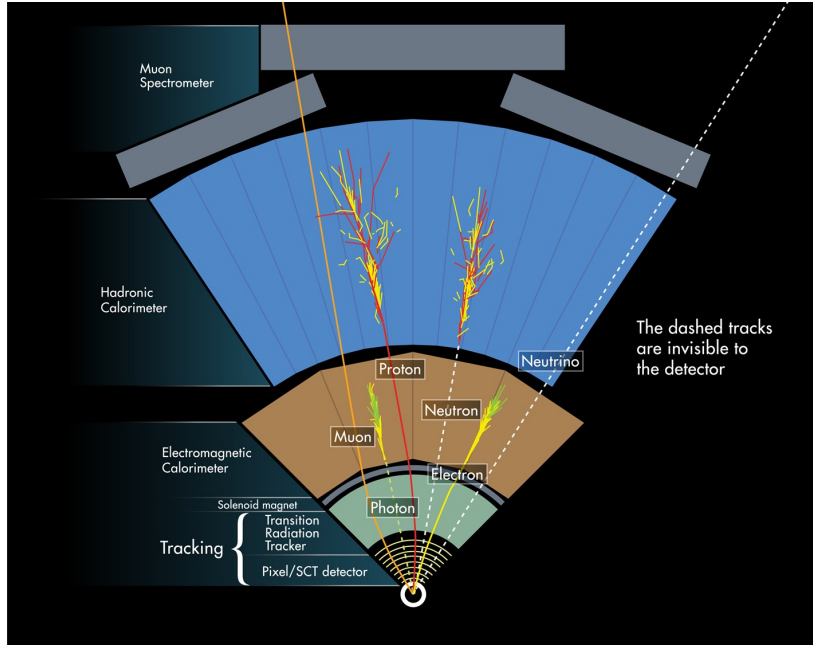


Figure 3.4: Crosssectional overview of the detector, showing typical examples of particles being detected.

y -component of a given coordinate is then expressed through a combination of a radius and the angle Φ between the x -axis, which points towards the center, and the y -axis, which points up. Another variable is the polar angle θ , which describes the angle between the x - y -plane and the z -axis. Additional useful variables are the angular difference (defined in equation 3.1), the pseudorapidity (defined in equation 3.2) and the transverse momentum (defined in equation 3.3). Moreover, the pseudorapidity can be used as a replacement for the polar angle θ .

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\Phi^2} \quad (3.1)$$

$$\eta = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right], \eta \in (-\infty, \infty) \quad (3.2)$$

$$p_T = \sqrt{p_x^2 + p_y^2} \quad (3.3)$$

3.3 Reconstruction

There are some particles that can not be observed by particle detectors. As displayed in figure 3.4, neutrinos do not interact with any of the layers of the ATLAS detector. Also, some particles with a small lifetime decay into other particles, hence only their decay products can be detected. However, based on reconstruction processes, particles that are not observed can be indirectly measured.

The presence of a neutrino is determined by checking for missing momentum [13], which can not be measured directly. However, in a symmetric particle collision, the sum of all measured transverse momenta is equal to zero. Any other result is a strong indication of an undetected neutrino.

A particle that has decayed can be reconstructed using the invariant mass of the daughter particles. If the particle decayed in a two body decay process, equation 2.3 can be used to express the motherparticle (C) and both of the daughter particles (A, B) in the rest frame of the motherparticle:

$$C_\mu = (m_0, 0, 0, 0), \quad (3.4a)$$

$$A_\mu = (E_A, p_{xA}, p_{yA}, p_{zA}), \quad B_\mu = (E_B, p_{xB}, p_{yB}, p_{zB}). \quad (3.4b)$$

The particles are related like this:

$$C_\mu = A_\mu + B_\mu. \quad (3.5)$$

By squaring this equation and using the energy momentum relation that is defined in equation 2.4, this can be simplified to:

$$m_C^2 = m_A^2 + m_B^2 - 2(E_A E_B - \vec{p}_A \vec{p}_B). \quad (3.6)$$

This relation can be used as a condition to check, weather or not two particles are the result of a known particle decay.

Machine learning

Machine learning has advanced significantly in the last decades. Nearly 60 percent of German companies are using at least one application of machine learning [14]. Moreover, there are many cases in everybody's daily life, where machine learning is a vital component. For example, numerous photos taken with a smartphone have been processed by a build-in machine learning [15]. And this are just a few out of many sophisticated cases, where machine learning is used [16].

Traditionally, algorithms are programmed for a certain task with a detailed step-by-step chain of commands. However, this method has some disadvantages, e.g. being written by a human and therefore, limited by the knowledge of the developer about the topic at hand. Moreover, such applications are mostly inflexible and not able to adapt to a change.

Contrary to this, machine learning attempts to fulfill a task with limited prior knowledge and no given solution of said task. Instead of executing an array of commands, machine learning is designed to recognize and approximate patterns. Therefore, applications that utilize machine learning are adaptable to different scenarios, that are similar in their approach. Especially, tasks where the development of a conventional algorithms would be too difficult, for example spam email recognition, can be fulfilled by machine learning applications.

4.1 Neural Network

Unsurprisingly, the the most common machine learning technique is an attempt to replicate the way biological neural systems are set up. This technique is called a neural network or artificial neural network. Similar to the human brain and its neurons, a neural network comprises many basic computation units. These units are commonly referred to as nodes, or fittingly, neurons.

An overview of a neural network is displayed in figure 4.1. In general, a neural network consists of three parts: the input, hidden and output layers. Moreover, a layer is an arrangement of nodes. All nodes from one layer are connected to all nodes in the next layer. Both the input and output layer act as an interface between user and neural network. The input layer receives data, which is forwarded to the hidden layer. Similarly the output layer is passed data by the prior layer and forwards it to the user. The intermediate zone between the input and output layer can not interact with the user, hence the name 'hidden' layer. Despite the name, the hidden layer can consist of one, or multiple layers, making the neural network a shallow or deep one. Moreover, the hidden layer is also solely responsible for the processing of data.

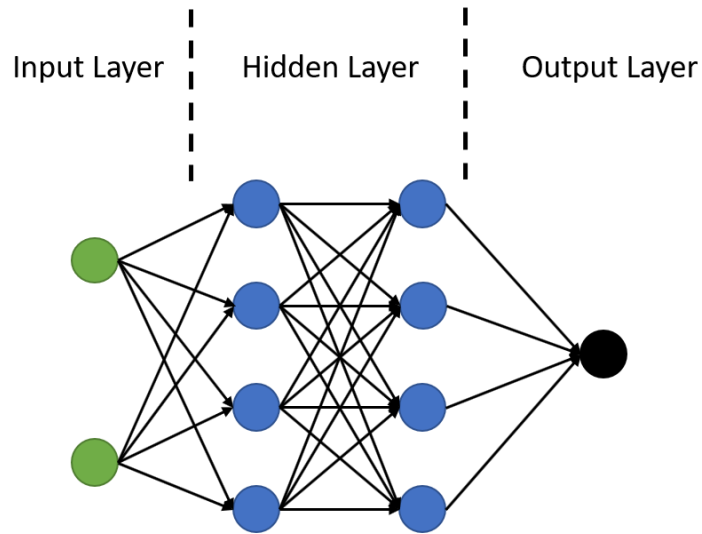


Figure 4.1: Overview of a neural network, comprising an input, hidden and output layer. All nodes from one layer are connected to all nodes of the next one.

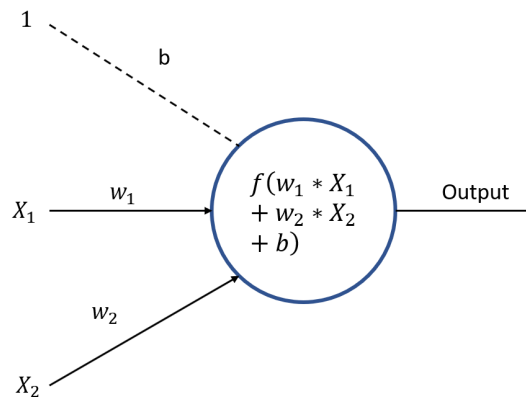


Figure 4.2: Depiction of a basic node. Two inputs (X_i) and weights (w_i), as well as an additional bias term (b) are the input values into the node. The sum of these values is then put through an activation function (f) and output to the next layer.

4.1.1 Function of a singular Node

Figure 4.2 displays a basic node with two input variables. In general, a node obtains input values from all nodes in the previous layer. If the node is an input node, it receives input from an external source. Furthermore, every input has an assigned weight. Therefore, the input of an arbitrary node can be written as:

$$Y_k^n = \sum_m w_{m,k}^n \cdot X_m^{n-1} + b_k, \quad (4.1)$$

where Y is the k -th node located in the n -th layer. In this equation, the node receives an input from m previous nodes, each providing an output X , with a weight w . Also, a bias term b is added.

Like its biological counterpart in the human brain [17], the output of a node changes significantly, if a certain threshold is crossed. For this purpose, activation functions determine the output of nodes. Additionally activation functions introduce non-linearity to tackle more complex problems. Some examples of activation functions are displayed in table 4.1. The binary step function is a prime example for the principle that is inspired by nature. If a certain value is reached, the node puts out 1 otherwise the output is 0. Therefore, the node is either activated or deactivated.

4.1.2 Measurements of Performance

To improve the performance of a neural network, a metric needs to be established to measure the performance of said network. Otherwise neither the user nor the neural network has an understanding of a good model, the network itself would be completely unguided and random.

Supervised learning is a scheme to train a neural network, where each event carries a *label*, which indicates, what the event is. The network then tries to classify the events. Therefore to measure the performance of the neural network, a more sophisticated method has to be established, which compares predictions the network makes and the label, that the event actually has.

In neural networks, the one of the most commonly used method of evaluating likelihood of a true prediction is the loss function. However, there are more quantities that measure the performance in other ways, like the *Receiver Operating Characteristic (ROC)* curve. Therefore, to evaluate the capabilities of a given neural network, multiple metrics are measured during training, in order to look at the performance from different angles.

Loss

The *loss function*, also often referred to as *cost function*, is an expression for the difference between the predictions of the neural network and true label. Therefore, to improve the accuracy of the classification, the loss has to be minimized. A basic loss function would be the absolute value of the difference between the predicted and the real labels. However, a more sophisticated function is the *binary cross entropy* which is defined in equation 4.2. In the formula, $p(\tilde{\tau})$ is the probability the network makes the classification $\tilde{\tau}$. τ is the actual label of the event.

$$L = -\frac{1}{N} \sum_{i=1}^N \tau_i \log(p(\tilde{\tau}_i)) + (1 - \tau_i) \log(1 - p(\tilde{\tau}_i)) \quad (4.2)$$

In this thesis the loss function is the binary crossentropy.

Receiver operating characteristic curve

For a system, that is subject to a binary classifier, the terms *true positive (TP)*, *false positive (FP)*, *false negative (FN)* and *true negative (TN)* are useful. This terms are best understood by taking a look at the confusion matrix displayed in table 4.2.

The ROC curve is a plot of the *true positive rate (TPR)* against the *false positive rate (FPR)*. The

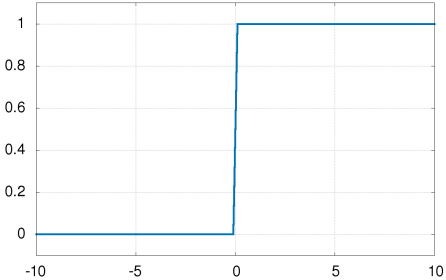
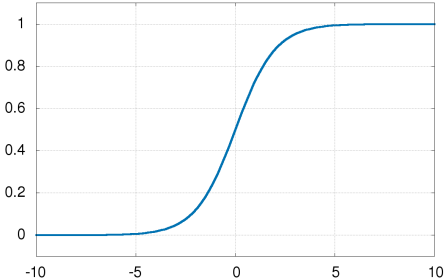
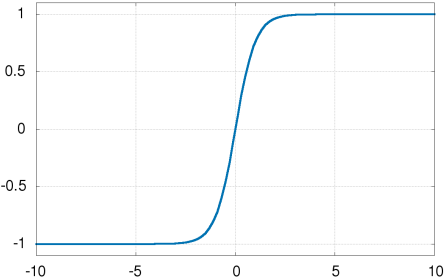
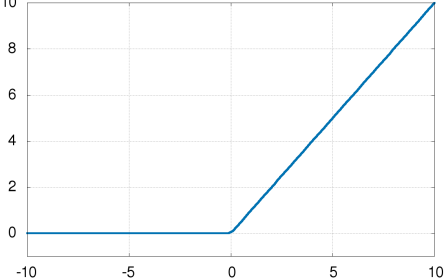
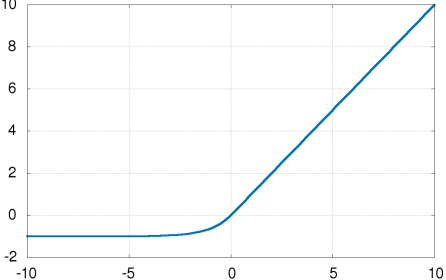
Name	Function	Plot
Binary step	$f(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$	
Sigmoid	$f(x) = \frac{1}{2} \left[1 + \tanh\left(\frac{x}{2}\right) \right]$	
Tanh	$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Relu	$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	
Elu	$f(x, \alpha) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	

Table 4.1: Examples of different activation functions by name, mathematical definition, and graphical representation [18].

		reference variables	
		Positive	Negative
variables called by algorithm	Positive	TP correctly classified as positive	FP falsely classified as negative
	Negative	FN falsely classified as positive	TN correctly classified as positive

Table 4.2: Confusion matrix for better understanding of terms like TP, FP, FN and TN. The labels on the y-axis are algorithm predictions. The labels on the x-axis are the real labels of the variables.

equations for the TPR and the FPR are listed as equation 4.3.

$$TPR = \frac{TP}{TP + TN} \quad (4.3a)$$

$$FPR = \frac{FP}{FP + TN} \quad (4.3b)$$

In case of equality between the TPR and FPR, the ROC curve becomes a diagonal. This kind of curve indicates a random selection and therefore, the system makes as many right as wrong predictions. A ROC curve above this diagonal line is desirable because that demonstrates that more positive than negative predictions are made. The *area under curve (AUC)* value is another good classifier to determine the performance of the network. The AUC value is calculated by integrating the ROC curve from 0 to 1. The result is a number between 0 and 1, where the random process has an AUC value of 0.5.

4.1.3 Backpropagation

Backpropagation is the backbone of a neural network. It allows a network to systematically improve based on its previous model. This is achieved by adjusting parameters after data has passed through the neural network. The cycle of all data passing through the network is called an *epoch*. In general, not all input data is passed through the neural network at once, rather it is divided into small packages, called batches. Instead of after each epoch, backpropagation happens after each batch has been processed by the network.

In general, a backpropagation algorithm takes a look at the gradient of the loss function with respect

to the weight and bias factor for each individual node. By calculating the impact the weights have on the gradient of the loss function after each epoch, beneficial adjustment to the weights can be predicted. These adjustments are made until ideally a minimum of the loss function is reached. The gradient is calculated by using the formulas from equation 4.4, where L is the loss function defined in equation 4.2, and Y was defined in equation 4.1.

$$\frac{\partial L}{\partial w_{m,k}^n} = \frac{\partial L}{\partial Y_k^n} \frac{\partial Y_k^n}{\partial w_{m,k}^n} \quad (4.4a)$$

$$\frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial Y_k^n} \frac{\partial Y_k^n}{\partial b_k} \quad (4.4b)$$

Using equation 4.1, equation 4.4 can be simplified:

$$\frac{\partial Y_k^n}{\partial w_{m,k}^n} = \sum_m X_m^{k-1} \implies \frac{\partial L}{\partial w_{m,k}^n} = \frac{\partial L}{\partial Y_k^n} \sum_m X_m^{k-1}, \quad (4.5a)$$

$$\frac{\partial Y_k^n}{\partial b_k} = 1 \implies \frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial Y_k^n}. \quad (4.5b)$$

The gradient can also be written as in equation 4.6. θ are the optimized parameters during the training, for example weights or bias terms. $\tilde{\tau}_i$ is the prediction made by the network, and τ_i is the label of the event. This more simplified and general form will be used in further calculations.

$$g(\theta) = \nabla_{\theta} \sum_i L(\tilde{\tau}_i, \tau_i) \quad (4.6)$$

A basic backpropagation algorithm iterates equation 4.7 until a stopping criteria, like minimization of the loss function, is reached. η is the step size for the updates, also called *learning rate*.

$$\theta_{i+1} = \theta_i - \eta g(\theta_i) \quad (4.7)$$

Optimizers are functions that determine how the parameters are adjusted based on the loss gradient. They take multiple parameters, and can lead to a quicker minimization of the loss function, that are calculated faster.

One of the most commonly used optimizer is the *Stochastic Gradient Descent (SGD)*, which is based on equation 4.7. As a further improvement, a parameter called momentum can be introduced. This momentum is used to take gradients of the previous steps into account. In general, multiple small steps using very aligned gradients can be replaced by fewer steps. However, if the gradients are less aligned, a low momentum is beneficial, because it prevents the overshooting of a minimum. With the user defined scaling parameter γ and the momentum, which is the sum over the past gradients, the equation, that calculates the change in the parameter θ becomes:

$$\theta_{i+1} = \theta_i - \eta g(\theta_i) + \gamma \sum_j \eta g(\theta_j). \quad (4.8)$$

Additionally, a decay parameter can be introduced, which decreases the effect of less recent gradients, making them less impactful over the course of a training.

Adaptive Optimizer

In contrast to SGD, Adam is an adaptive optimizer, which is influenced by the past gradients. It updates the weights, by taking an average of past gradients and squared past gradients. An update of parameters in Adam looks like equation 4.9. m , defined in 4.9a and v , defined in 4.9b are biased estimates of the two momenta, which Adam uses. \tilde{m} , defined in 4.9c and \tilde{v} , defined in 4.9d are the bias corrected forms estimates. With both momenta, parameters like weights, can be updated as shown in 4.9e.

$$m_i = \beta_1 m_{i-1} + (1 - \beta_1) g(\theta_{i-1}) \quad (4.9a)$$

$$v_i = \beta_2 v_{i-1} + (1 - \beta_2) g(\theta_{i-1})^2 \quad (4.9b)$$

$$\tilde{m}_i = \frac{m_i}{1 - \beta_1} \quad (4.9c)$$

$$\tilde{v}_i = \frac{v_i}{1 - \beta_2} \quad (4.9d)$$

$$\theta_{i+1} = \theta_i - \frac{\eta}{\sqrt{\tilde{v}_i} + \epsilon} \tilde{m}_i \quad (4.9e)$$

β_1 , β_2 and ϵ are chosen by the user, based on the understanding of the problem. According to [19] $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10 \times 10^{-8}$ are a good set of default parameters. The decay parameter, which was mentioned in section 4.1.3, can also be introduced to Adam.

4.1.4 Improving the Performance

Overfitting

One of the most common problems when optimizing a neural network is overfitting. During the training, a neural network picks up many features of which not all are actual features of the data, but random fluctuations and noise. An example for the concept of overfitting in general can be seen in figure 4.3. While the orange function represents the training data perfectly, it diverges significantly from testing data. The blue function is obviously a better fit which describes both training and testing data. This problem can be countered in many ways, some examples are data splitting and dropout.

In a feedforward network that uses supervised learning, the dataset used for training can be divided into two sets. One is specifically designated for training the network, while the other is for validating the model that the network has developed during training. Hence, a difference in performance, especially a growing divergence between training and testing data indicates overfitting.

Another behavior, that a neural network can develop is a dependency between nodes of different layers on each other. Therefore, certain nodes could form a dominant "path" through the network. This can cause the nodes to become interdependent, hence leading to a less general network. To counteract this behavior, a method called dropout can be used. By applying dropout, every epoch a certain amount of nodes in every layer is deactivated. This concept is sketched in figure 4.4. However, dropout decreases the speed at which the network finds a minimum.

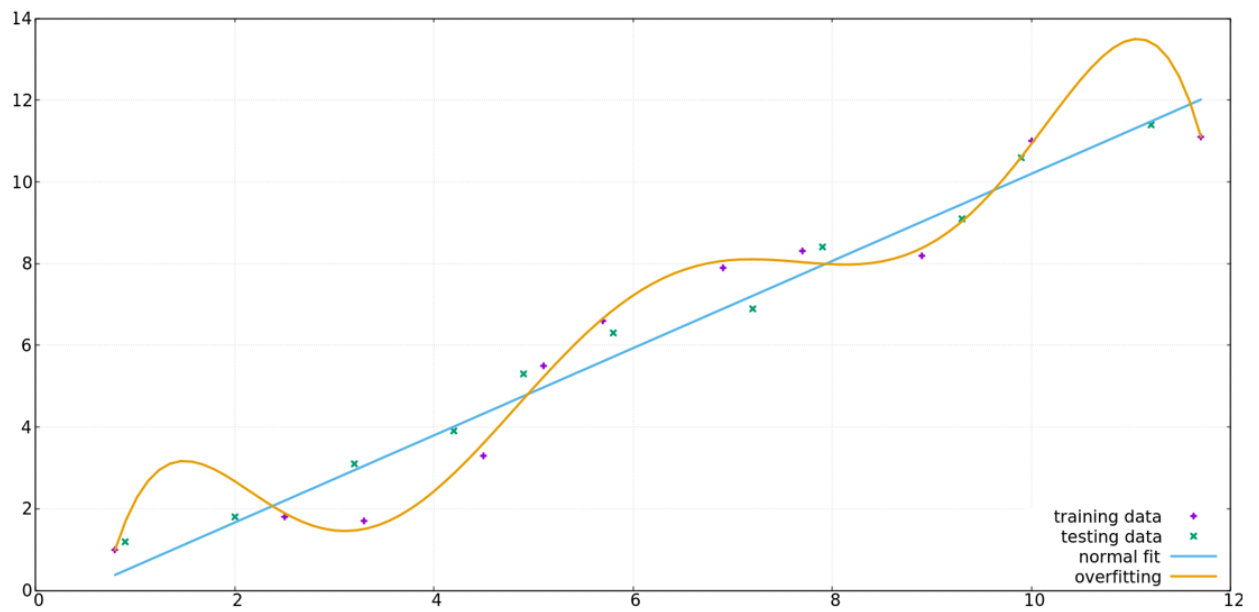


Figure 4.3: Theoretical example of the concept of overfitting, displaying one set of training and one set of testing data. One function is an adequate fit, while the other is an overfit.

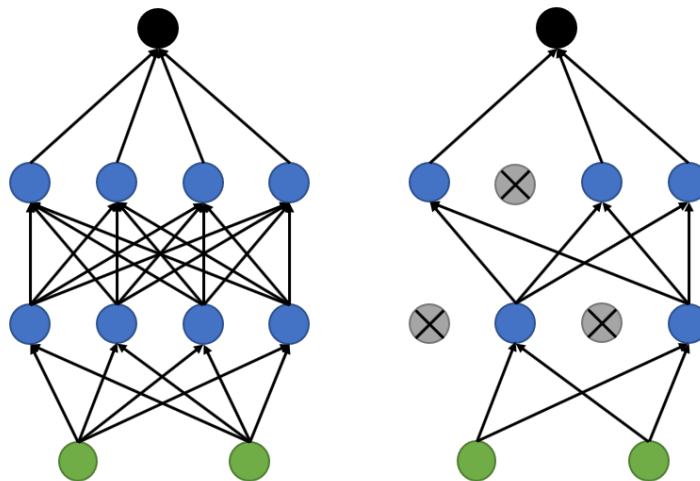


Figure 4.4: Sketch of the network shown in figure 4.1, before and after dropout is applied. On the left side, all the nodes within the range of one layer are connected. On the right side, some nodes have been deactivated by dropout and therefore are not connected to the other nodes.

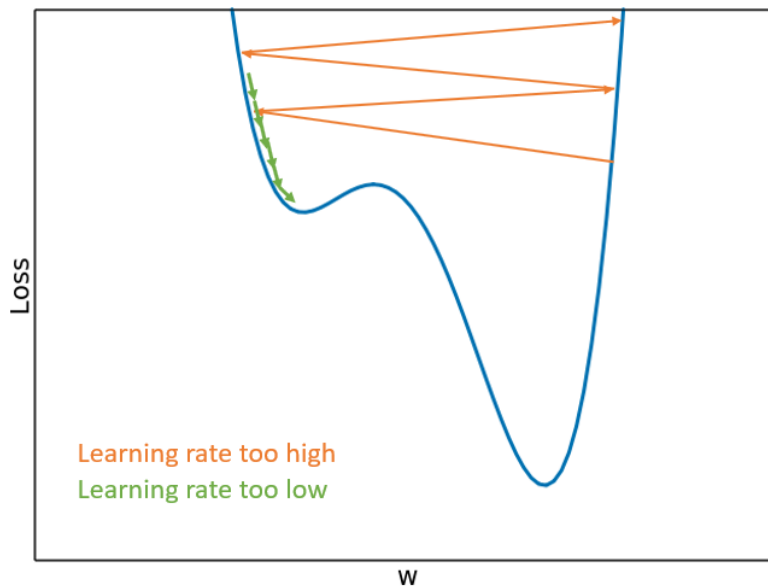


Figure 4.5: Visualization of the influence that different learning rates have on the learning process.

Optimizing Hyperparameters

Hyperparameters are parameters, that are used to control the training process of a neural network. In contrast to other parameters, the values of hyperparameters are set before and not changed during the training process of the network. An optimization of this parameters can increase the performance of a neural network significantly.

Some hyperparameters change the setup of the neural network, examples of this being the number of nodes per layer, the amount of layers and the number of epochs. The choice of hyperparameters need to match the complexity of the problem. If one chooses a simple network for a complicated problem, then the network will not perform well or be mostly unable to classify. On the other hand, if a network is more complicated than the problem, it can reach overfitting much faster and be just as useless in application.

As mentioned before, optimizers also use hyperparameters. Both SGD and Adam have learning rates and decay hyperparameters. In both cases, one of the most impactful hyperparameters is the learning rate. Using a large learning rate can cause the network to skip and never find the minimum. A small learning rate, however, can cause the network to get stuck in a local minimum and therefore, not find the minimum at all. This behavior is visualized in figure 4.5.

The batch size is also a hyperparameter that has an influence on training. If the batch is too small, some features could be recognized as dominant because they are dominant in the current batch. One way to encounter this problem is batch normalization. In applying batch normalization, the output between each layer becomes normalized, which decreases the effect of overrepresented features.

The optimization of a neural network is not a trivial task. One obvious way to search for the right combination of hyperparameters is a grid search. Selecting an area of reasonable hyperparameters and looking at their results can reveal an optimized set of hyperparameters. However, the more

hyperparameters a neural network has, the larger the grid becomes that has to be searched.

A more sophisticated approach is the evolutionary search. For this method, sets of hyperparameters are randomly generated. After all sets of hyperparameters have finished training, the loss function is used to rank the sets based on their performance. The worst performing sets are replaced by new sets of hyperparameters that have been created based on the better performing ones. This process then is repeated until performance no longer increases.

tZq Process

The top-quark is the most massive elementary particle measured. It has a lifetime of 0.5×10^{-24} s, which is shorter than the timescale at which the strong force acts, making it unable to form hadrons. Because of this, one can probe the top-quark's decay in order to get information on a seemingly solitary quark. Information, like polarization, that is typically lost when quarks form bound states. Hence, by observing the top-quark, the consistency of the SM can be tested. Typically, the top quark is produced in $t\bar{t}$ pairs via the strong interaction. However, there are single-top-quark production channels. One of the more rare single-top-quark production processes is the focus of this thesis, the tZq .

Feynman diagrams are a two dimensional graphical representations of mathematical expressions which describe complex interactions between sub-atomic particles in physics. One of the two axes represents space while the other represents time. Straight lines are used to express fermions. An arrow on the straight line points either in the direction of time if the depicted fermion is a particle, or against time if it is an antiparticle. Bosons are either portrayed as a dashed or wavy line, depending on the convention. The exception of this rule is the gluon, which is illustrated using a line that looks like a spring. Lowest order Feynman diagrams of the tZq process are displayed in figure 5.1.

5.1 Signal Regions

For this process, two signal regions are defined: 2j1b and 3j1b. The number in front of the j refer to the number of jets that are detected in the respective signal region. A jet is a cone of collimated particles, that originates from the hadronization of quarks or gluons. Jets which originate from a b-quark are called b-jets. To identify b-jets, an algorithm scores all jets based on how likely they are to originate from a b-hadron decay. Therefore jets that are identified as b-jets are called b-tagged. The number in front of the b expresses how many b-jets are required to form the signal region.

As displayed in figure 5.2 a possible final state for a tZq production process contains three leptones, one neutrino, one b-quark, and one quark. The quarks are detected as jets or as b-jets if tagged as such. The untagged jet is typically found in the forward part of the detector and so it is called a forward jet. The additional untagged jet in the 3j1b signal region originates from a gluon-jet, which is referred to as radiation.

To differentiate the leptons they are called l_1 , l_2 and l_3 , where l_1 and l_2 are the leptons originating from the decay of the Z-boson. Therefore l_3 is the lepton, that is produced by the decay of the W^\pm -boson. In order to determine which leptons are l_1 , l_2 and l_3 , equation 3.6 is used. The two leptons

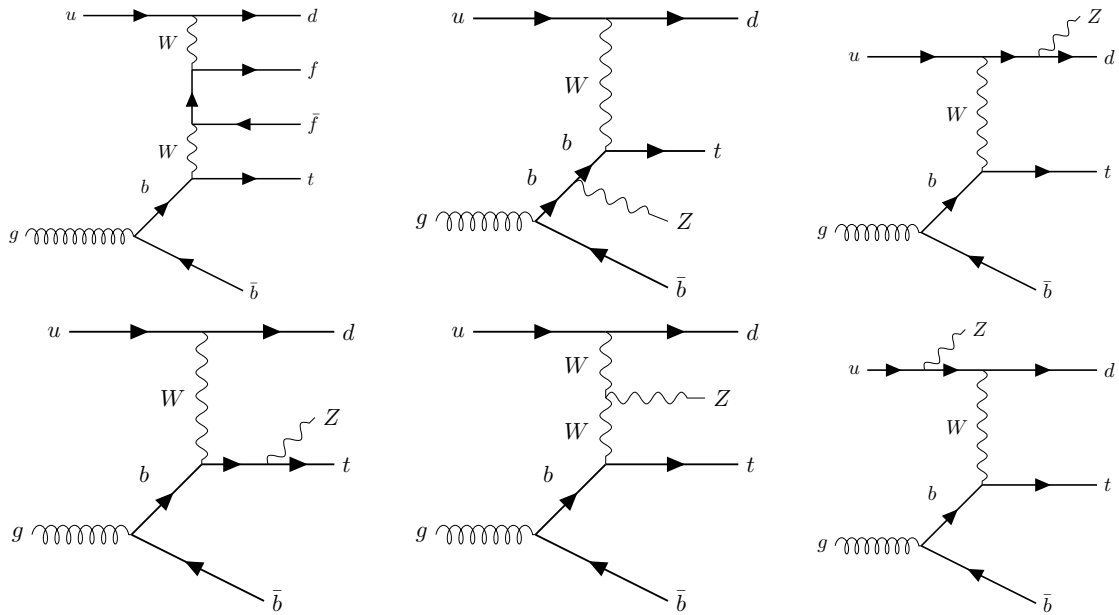


Figure 5.1: Lowest order Feynman diagrams of the tZq process.

which combine to make a Z -boson with mass closest to the known Z -boson mass are assigned as l_1 and l_2 ; the remaining one is l_3 .

5.2 Background Processes

Background processes can be falsely identified as signal for various reasons. They can either have final states which appear similar to the signal region or objects that have been misidentified by the reconstruction process.

Background processes of the tZq process are the productions of $t\bar{t}V$ (displayed in figure 5.3(a)), diboson (displayed in figure 5.3(b)), single top-quark (displayed in figure 5.3(c)), $t\bar{t}$ (displayed in figure 5.3(d)), Z -jets (displayed in figure 5.3(e)) and $tW^\pm Z$ (displayed in figure 5.3(f)). The single top-quark production channel has a low contribution however, it is considered as a background process to tZq . Moreover only the tW^\pm sample is evaluated because the s- and t-channel are not overlapping with the tZq process. Therefore only the tW^\pm is displayed in figure 5.3(c). The diboson background channel has three different event sections: $W^\pm W^\pm$, $W^\pm Z$ and ZZ events, figure 5.3(b) shows the $W^\pm Z$ channel.

All of these processes can be falsely identified as the 2j1b or 3j1b signal region. In the $tW^\pm Z$ process for example both gluons as well as the b-quark are detected as jets. Furthermore, the Z - and W^\pm -boson decay in three leptones and one neutrino, which is the same amount and configuration as in the tZq process. Therefore, the requirements of the 3j1b region are met. For the same reasons, the $t\bar{t}V$ and $t\bar{t}$ production channel also appear as 3j1b signal region.

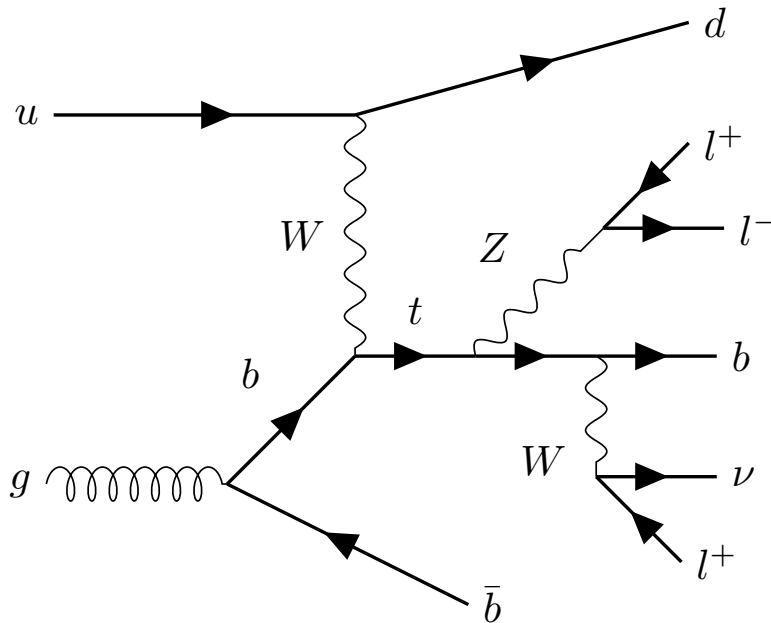


Figure 5.2: Feynman diagram of the full decay chain of the tZq process in the trilepton final state.

5.3 Choice of Variables

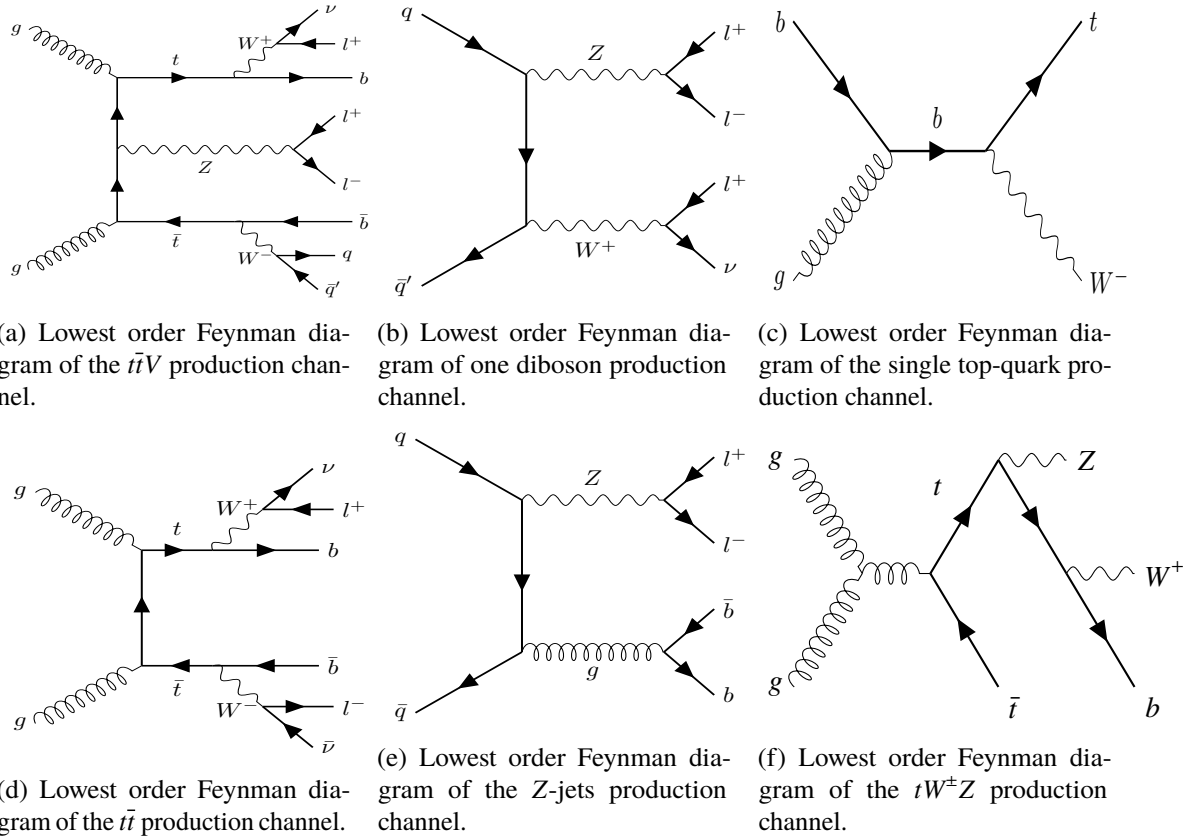
The core of every machine learning application is to identify features of the dataset at hand. However, a machine learning application does not have a sensory system and is therefore highly depended on input data. Thus, the input data has to be carefully chosen by a human with an understanding of the topic at hand, so that useful information is passed to the network.

As an example, one could imagine a machine learning application that is tasked with distinguishing between cats and dogs based on given input variables. An example for less viable variables would be whether or not the animal has fur, claws or teeth; while the height, length and the sound it makes would be good variables.

The number of background processes is large in comparison to the number of the tZq events. Therefore, a simple analysis would be difficult and unreliable. However, a more sophisticated method like a neural network can be used to increase the discrimination between signal and background events.

5.3.1 Commonly used Variables

Signal and background events are similar in the configuration of their final state particles. However, other properties of the final state particles can be analyzed. Commonly, variables which describe characteristic features based on which particles can be identified, are chosen to differentiate between the signal and background. These variables can be simple, or complex reconstructed variables. Listed in table 5.1 are some variables that are chosen to train a neural network to distinguish tZq process from background events.


 Figure 5.3: Lowest order Feynman diagrams of background processes of the tZq process.

5.3.2 Lorentz invariant Variables

As defined in section 2.4, Lorentz invariance refers to quantities that are not changed by a Lorentz transformation. Lorentz invariant variables, do not depend on boosts or orientation of their system. Therefore, a lot of uncertainties become reduced. Lorentz invariant variables also do not correlate with each other.

Squaring the equation, that was defined in 3.5, shows the daughter particle's kinematics to the invariant mass of the mother particle:

$$\begin{aligned} C_\mu C^\mu &= (A_\mu + B_\mu)(A^\mu + B^\mu) \\ &= A_\mu A^\mu + 2A_\mu B^\mu + B_\mu B^\mu, \end{aligned} \quad (5.1a)$$

$$\begin{aligned} A_\mu B^\mu &= \frac{1}{2} (C_\mu C^\mu - A_\mu A^\mu - B_\mu B^\mu) \\ &= \frac{1}{2} (m_C^2 - m_A^2 - m_B^2). \end{aligned} \quad (5.1b)$$

Variable	Rank		Description
	2j1b	3j1b	
m_{bjf}	1	1	(Largest) invariant mass of the b-jet and the untagged jet(s)
m_{top}	2	2	Reconstructed top-quark mass
$ \eta(j_f) $	3	3	Absolute value of the η of the forward-jet
$m_T(l, E_T^{\text{miss}})$	4	4	Transverse mass of the W^\pm -boson
b tagging score	5	11	b-tagging score of the b-jet
H_T	6	-	Scalar sum of the p_T of the leptons and jets in the event
$q(l_3)$	7	8	Electric charge of the lepton from the W^\pm -boson decay
$ \eta(l_3) $	8	12	Absolute value of the η of the lepton from the W-boson decay
$p_T(W)$	9	15	p_T of the reconstructed W^\pm -boson
$p_T(l_3)$	10	14	p_T of the lepton from the W^\pm -boson decay
$m(l_1 l_2)$	11	-	Mass of the reconstructed Z-boson
$ \eta(Z) $	12	13	Absolute value of the η of the reconstructed Z-boson
$\Delta R(j_f, Z)$	13	7	ΔR between the forward-jet and the reconstructed Z-boson
E_T^{miss}	14	-	Missing transverse momentum
$p_T(j_f)$	15	10	p_T of the forward-jet
$ \eta(j_r) $	-	5	Absolute value of the η of the radiation-jet
$p_T(Z)$	-	6	p_T of the reconstructed Z-boson
$p_T(j_r)$	-	9	p_T of the radiation-jet

Table 5.1: Variables, which are used for a neural network in the 2j1b and 3j1b signal regions. The variables are ranked for both regions [20].

Moreover, using p_T , Φ and η a four-vector can be written as:

$$a^\mu = \begin{pmatrix} p_T \cosh(\eta) \\ p_T \cos(\Phi) \\ p_T \sin(\Phi) \\ p_T \sinh(\eta) \end{pmatrix}. \quad (5.2)$$

Therefore a scalar product between two vectors, which do not have a common ancestor, in this notation is:

$$\begin{aligned} \langle A|B \rangle &= A_\mu B^\mu = p_{T,A} p_{T,B} [\cosh(\eta_A) \cosh(\eta_B) - \cos(\Phi_A) \cos(\Phi_B) - \sin(\Phi_A) \sin(\Phi_B) - \sinh(\eta_A) \sinh(\eta_B)] \\ &= p_{T,A} p_{T,B} [\cosh(\eta_A - \eta_B) - \cos(\Phi_A - \Phi_B)]. \end{aligned} \quad (5.3)$$

The only pair of particles that originate from a common ancestor are l_1 and l_2 . Therefore, their scalar product is described by 5.1b. All other particles do not share a common ancestor, and can therefore be described using 5.3. Even though, p_T , Φ and η are not Lorentz invariant properties, the scalar product in equation 5.3 is Lorentz invariant[21].

Therefore full set of Lorentz invariant parameters can be determined, which comprises scalar products as well as Lorentz invariant variables from table 5.1. For both the 2j1b and 3j1b regions these variables are listed in table 5.2.

Variable	2j1b	3j1b	Description
m_Z	x	x	Mass of the reconstructed Z-boson
$m_{b_j f}$	x	x	(Largest) invariant mass of the b-jet and the untagged jet(s)
m_{top}	x	x	Reconstructed top-quark mass
$q(l_W)$	x	x	Electric charge of the lepton from the W^\pm -boson decay
$\langle l_1 l_3 \rangle$	x	x	Scalar product between the first and second lepton
$\langle l_2 l_3 \rangle$	x	x	Scalar product between the second and third lepton
$\langle b_j l_1 \rangle$	x	x	Scalar product between the b-jet and the first lepton
$\langle b_j l_2 \rangle$	x	x	Scalar product between the b-jet and the second lepton
$\langle b_j l_3 \rangle$	x	x	Scalar product between the b-jet and the third lepton
$\langle f_j l_1 \rangle$	x	x	Scalar product between the forward-jet and the first lepton
$\langle f_j l_2 \rangle$	x	x	Scalar product between the forward-jet and the second lepton
$\langle f_j l_3 \rangle$	x	x	Scalar product between the forward-jet and the third lepton
$\langle f_j b_j \rangle$	x	x	Scalar product between the forward-jet and the b-jet
$\langle r_j l_1 \rangle$		x	Scalar product between the radiation jet and the first lepton
$\langle r_j l_2 \rangle$		x	Scalar product between the radiation jet and the second lepton
$\langle r_j l_3 \rangle$		x	Scalar product between the radiation jet and the third lepton
$\langle r_j b_j \rangle$		x	Scalar product between the radiation jet and the b-jet
$\langle r_j f_j \rangle$		x	Scalar product between the radiation jet and the forward-jet

Table 5.2: Lorentz invariant variables, which are used to train a neural network in the 2j1b and 3j1b signal regions. The x marks whether or not the variable appears in the respective signal region. The term scalar product refers to the definition given in equation 5.3.

Optimization of the Hyperparameters

The primary focus of this chapter is the comparison between networks, that have been trained using Lorentz invariant and commonly used variables. Therefore, neural networks are set up and trained using the Lorentz invariant variables for both signal regions. The performance of the networks trained with Lorentz invariant variables are compared to the networks that were based on commonly used variables.

In order to create neural networks, the open-source library Keras [22] is used. Keras is written in python and able to run on TensorFlow, CNTK, or Theano. Keras comprises almost all tools, which are essential in order to build up a neural network, e.g. activation functions and optimizers.

6.1 Choice of Optimizer

After setting up the network, it is tested with a set of initial hyperparameters and the Lorentz invariant variables from table 5.2. Using supervised training, the network is trained to differentiate between signal and background events. Initially the optimizer SGD was used, however it did not seem to be able to fulfill the previously mentioned objective. The initial chosen learning rate was causing the loss function to stagnate at a certain value or to alternate and therefore the loss never converged reliably to low values. However, as mentioned before in section 4.1.4 alternating loss values over the course of the training can be the result of a large learning rate. Nevertheless, the impression that SGD was not able to separate signal and background events was backed up by the ROC curve. While the ROC curves often seemed to follow a linear gradient, the AUC values of these ROC curves were close to 0.6. Furthermore, a grid search scanning pairs of learning rates and momenta was conducted, but it did not reveal a promising combination either.

Therefore Adam, is tested as optimizer. In figure 6.1 the difference between Adam and SGD is displayed. While SGD does not seem to learn well, Adam is able to separate the events reliably, even though it is clearly underfitted. Because of this, the search for a good combination of a learning rate and momentum for SGD is abandoned and all further networks are trained with Adam as optimizer.

In order to increase the performance of the neural network, the set of hyperparameters is optimized as well. However, the large number of hyperparameters of the system forces a large grid that has to be searched. This behavior is often referred to as the curse of dimensionality. Because of this the general approach is to optimize one or two hyperparameters at once based on the loss-value. A combination of hyperparameters that resulted in the lowest loss value is assumed to be an optimized set.

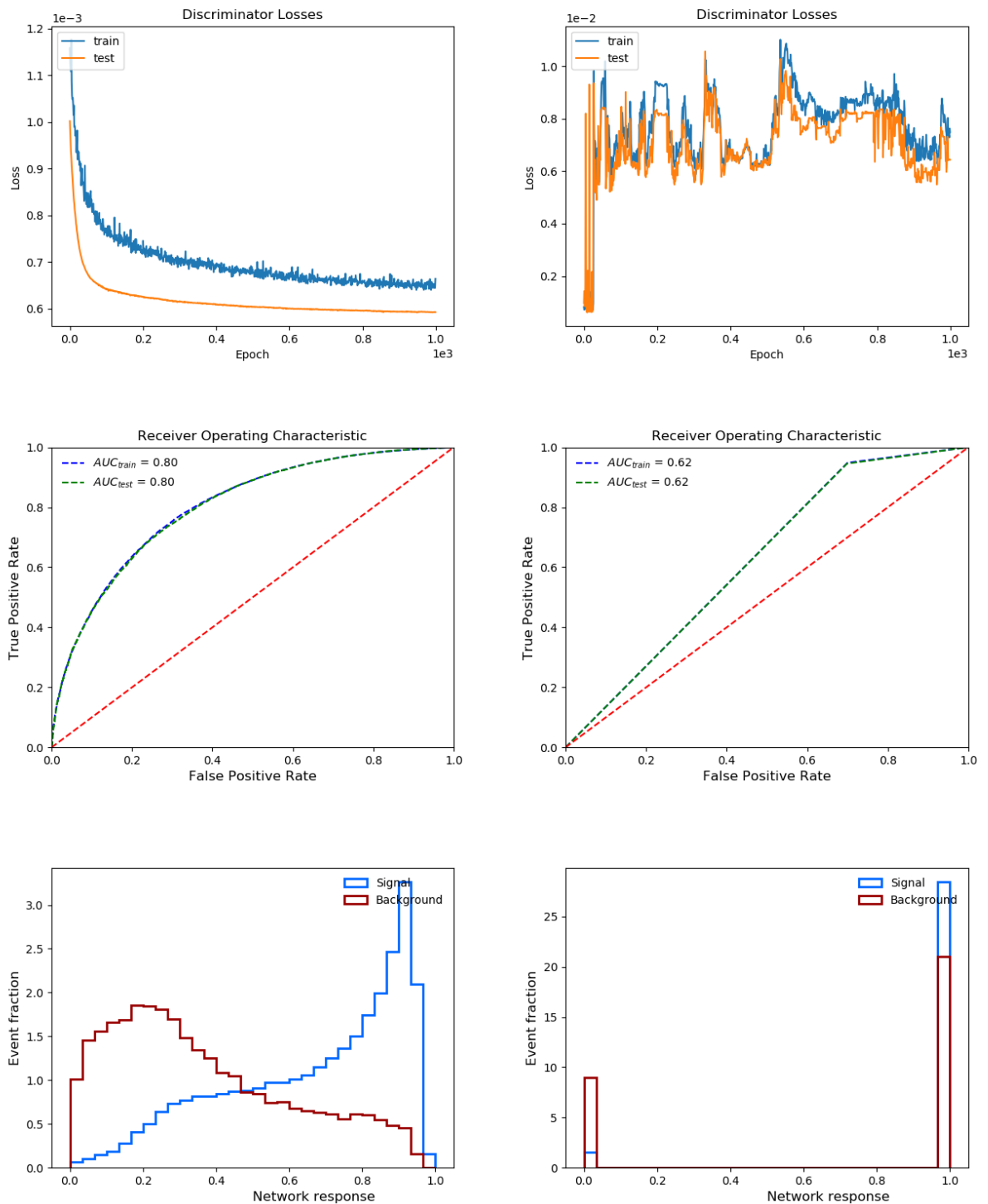
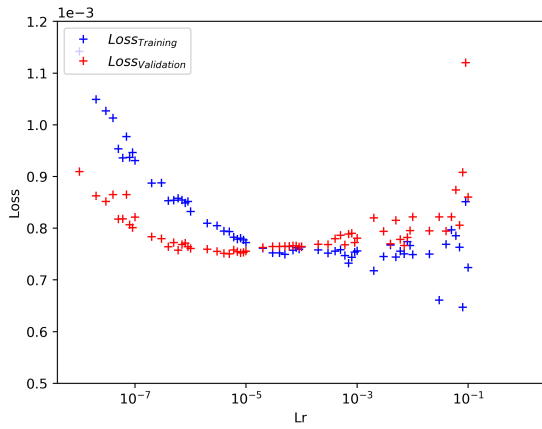


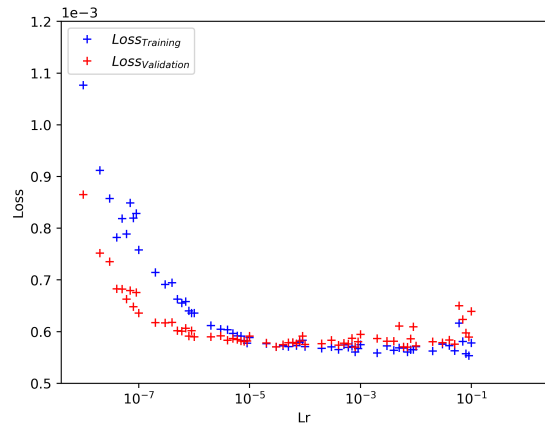
Figure 6.1: Comparison between Adam and SGD, based on different metrics. From top to bottom the metrics are loss, ROC with AUC, and the response of the network. Diagrams on the left side display the results achieved using Adam. The right column are the metrics of the network using SGD.

6.1 Choice of Optimizer

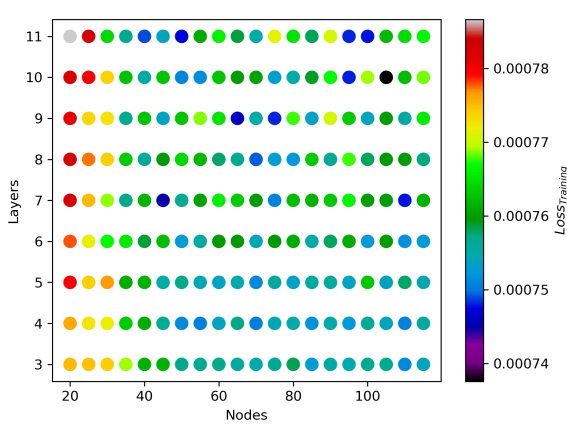
Nodes=80, Layers=7, Dropout=0.1, Validation=0.2, Batchsize=1024, Decay=1e-07



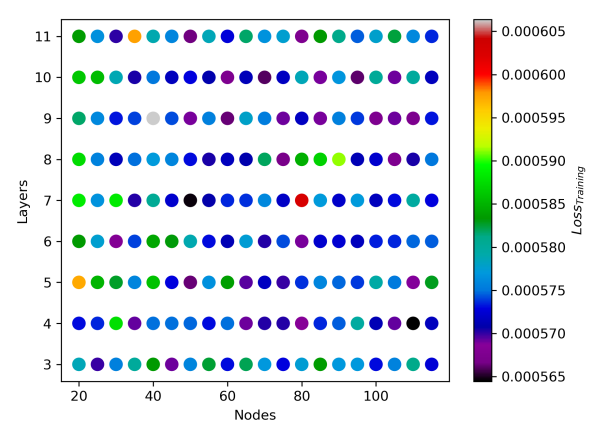
Nodes=80, Layers=7, Dropout=0.1, Validation=0.2, Batchsize=1024, Decay=1e-07



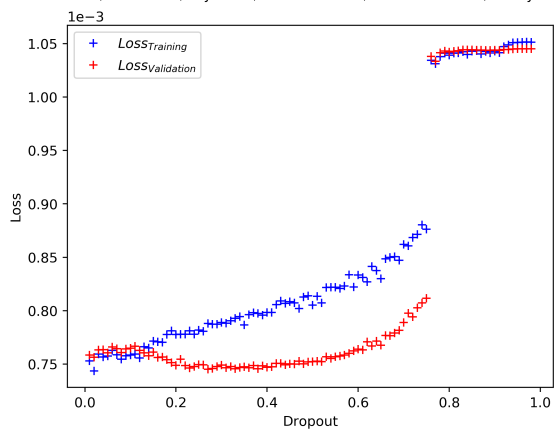
Lr=5e-05, Dropout=0.1, Validation=0.2, Batchsize=1024, Decay=1e-07



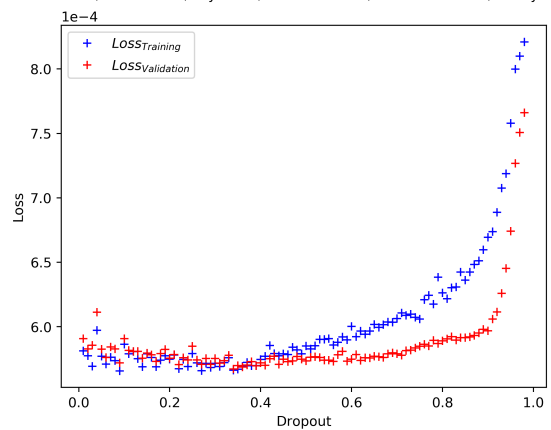
Lr=6.5e-05, Dropout=0.1, Validation=0.2, Batchsize=1024, Decay=1e-07



Lr=5e-05, Nodes=45, Layers=7, Validation=0.2, Batchsize=1024, Decay=1e-07



Lr=6.5e-05, Nodes=110, Layers=4, Validation=0.2, Batchsize=1024, Decay=1e-07



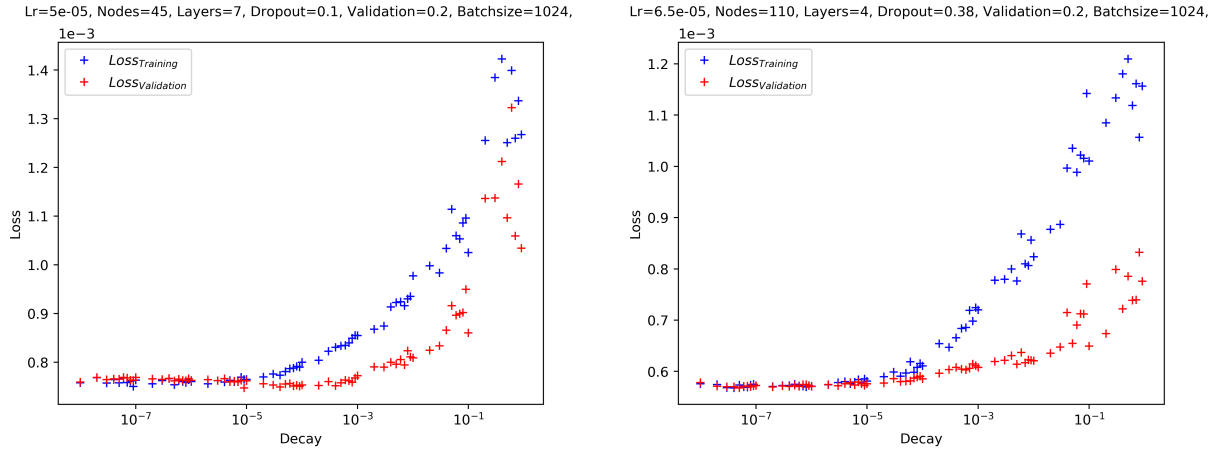


Figure 6.2: Loss values of different neural networks, that were trained using different hyperparameters. From top to bottom the changing hyperparameters were the learning rate, the number of nodes per layer and number of layers, the dropout and the decay. The 2j1b signal region is on the left side and the 3j1b region on the right.

	2j1b	3j1b	range for grid search
Learning rate	5×10^{-5}	6.5×10^{-5}	$[1 \times 10^{-7}, 1]$
Layers in the hidden layer	7	4	[3, 11]
Nodes per hidden layer	45	110	[20, 120]
Dropout	0.1	0.38	[0.01, 0.99]
Decay	4×10^{-6}	7×10^{-8}	$[1 \times 10^{-8}, 0.1]$

Table 6.1: Optimized values of hyperparameters used to train neural networks for both the 2j1b and 3j1b signal regions. The ranges for the grid search are also listed for each parameter.

This process is iterated until all hyperparameters are optimized. Figure 6.2 displays this iterative process. First the learning rate was optimized, second the number of layers and nodes, third is the dropout and lastly the decay hyperparameter.

The range for the grid searches are displayed in table 6.1, as well as the resulting optimized parameters. For this training processes, the number of epochs was set to 1000. The loss of performance, that can occur during training over too many epochs was countered by early stopping. This method stops the training process when indications overfitting is detected.

6.2 Differences between Signal Regions

During the optimization process, some differences between the 2j1b and 3j1b signal region were observed. A good example of these differences are displayed in figures 6.2 and 6.3. The values in the figures displaying the 3j1b region seem to alternate more, while the figures of the 2j1b signal region appear to follow a function. A comparison of the performance of neural networks using different signal regions, which is displayed in figure 6.4, also shows a difference. Networks trained using the 2j1b and 3j1b signal regions result in similar ROC curves and AUC values. However, the optimized

network, which was trained using the 3j1b signal region archived a lower loss value than the one using the 2j1b signal region.

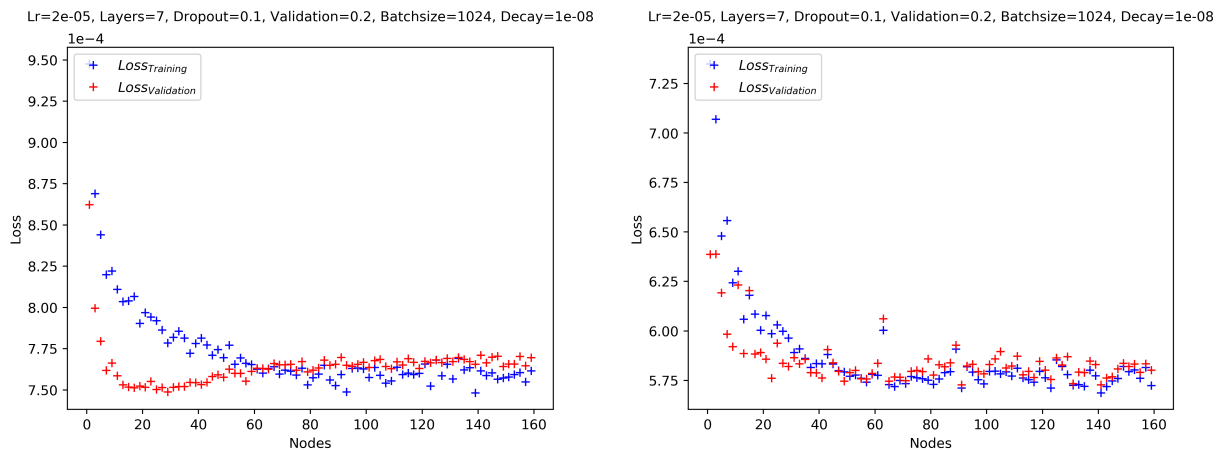


Figure 6.3: Loss values of different neural networks, that were trained using different amounts of nodes per layer. The 2j1b signal region is on the left side, the 3j1b region on the right.

6.3 Comparison

Figure 6.5 shows the performance of a neural network which was trained using the variables from table 5.1. Comparing this figure to figure 6.4, shows mixed results. On one side, the ROC curve of the network that was trained with commonly used shows a higher AUC value than both networks that used Lorentz invariant variables. However the loss value of both of these networks are smaller than the loss value of the network that was trained using the variables from table 5.1.

Using Lorentz invariant variables to train the networks seem to be a viable alternative to commonly used variables. The results achieved by using Lorentz invariant variables seem to be as good as results achieved by using common variables. Furthermore the Lorentz invariant variables could even prove to be a better choice, because of advantages over commonly used variables, e.g. that they are not correlated and independent of boost and orientation of the system.

6.4 Outlook

This behavior could be the result of the attempt to optimize the hyperparameters based on the minimization of the loss value. Therefore another grid search could be conducted, which is focused on the maximization of the AUC value. Moreover, by using the previously mentioned method of optimizing one or two hyperparameter at a time, it is possible that the found minimum of the loss function is not the global minimum but rather a local one. Also before starting this iterative process, a initial set of hyperparameters was chosen, which can also limit the success of the search of optimal hyperparameters.

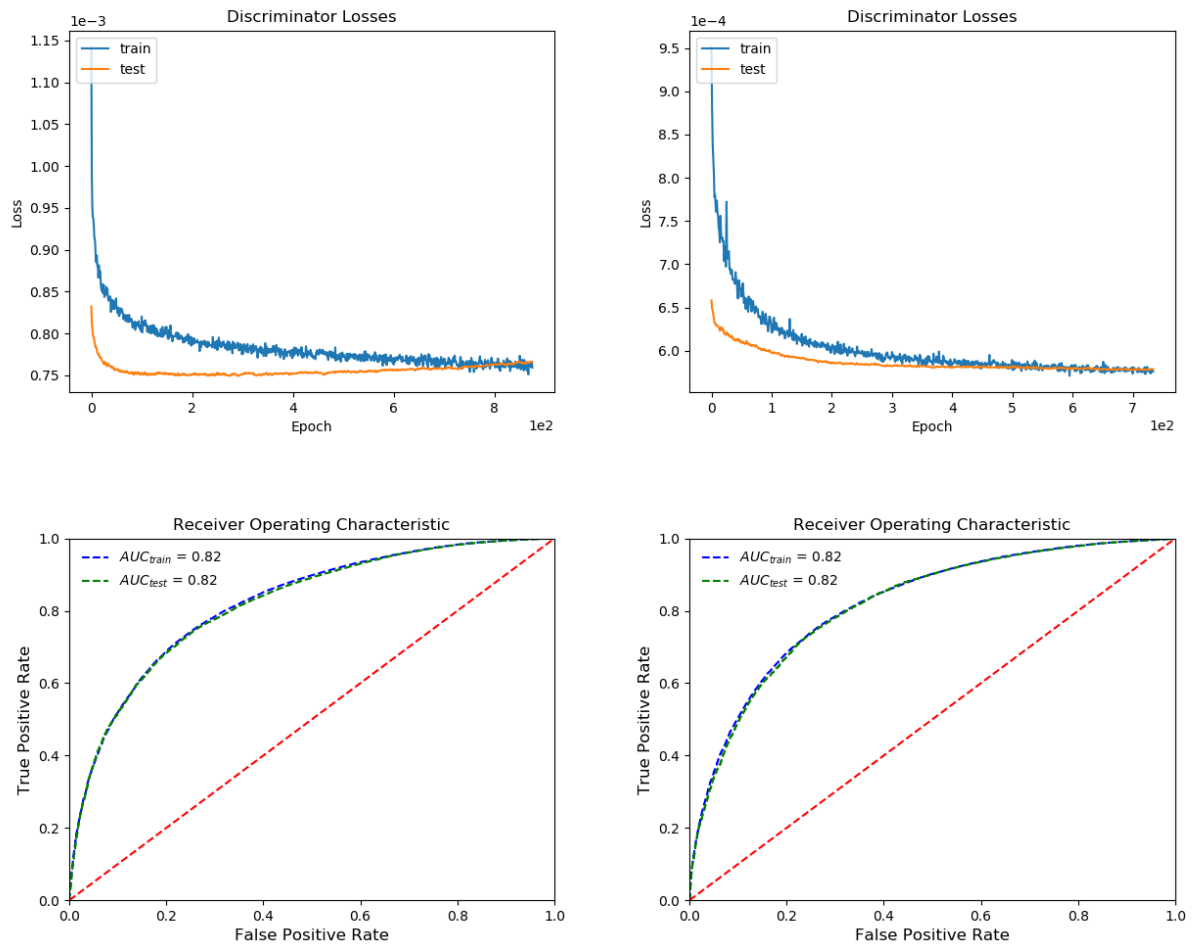


Figure 6.4: Performance of two optimized neural networks, which were trained using Lorentz invariant variables, measured by different metrics. The the graphs on the left side show metrics of the network, that was trained using the 2j1b region. The graphs on the right show metrics of the 3j1b region.

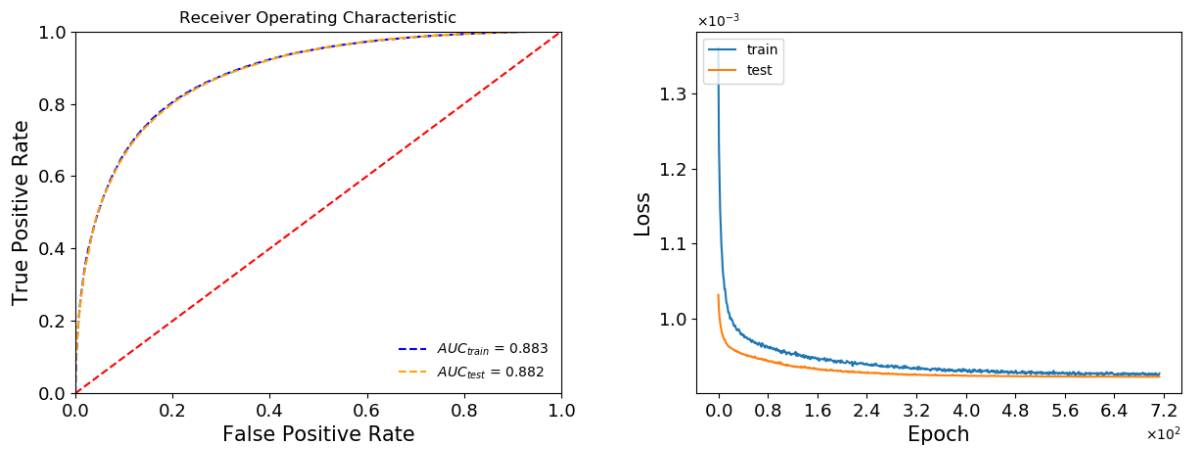


Figure 6.5: Performance of a neural network, which was trained the variables from table 5.1, measured by different metrics [23].

To counteract the curse of dimensionality that prevented a proper grid search, an evolutionary search could be implemented. Additionally, other optimization schemes that were not mentioned in this thesis could be introduced, such as K-Fold [24] and the L1 and L2 norm [25].

Conclusion

The main focus of this work was to compare the performance of neural networks, which were trained with different sets of variables with respect to their ability to discriminate signal and background events in the tZq process. This process is of interest because of the inability of the top quark to hadronize, making it describable as a free quark. Frequently, kinematic variables are used as input variables for the neural network.

The concept of machine learning, as well as the concept of Lorentz invariance, was introduced. A set of Lorentz invariant variables was defined, which are not depended on the unpredictable boost and orientation of the system. Furthermore Lorentz invariant variables also do not correlate with each other. Neural networks, which were operating in two different signal regions, were optimized by using a grid search. The performance of these networks was then compared to a network which was trained with commonly used variables.

While training neural networks with Lorentz invariant variables two patterns have been observed. The first pattern was that while one signal region had a lower loss value, the other signal region was alternating less. The second pattern was that networks in both signal regions had a lower AUC value than a network which was trained using common variables. However, the loss values of the networks that used Lorentz invariant variables were lower. This can be the result of the optimization of the hyperparameters, which had the objective to minimize the loss.

Nonetheless networks which were trained using Lorentz invariant variables performed as well as networks which used common variables. Therefore Lorentz invariant variables are a considerable option for training neural networks in both signal regions and can even be a better choice, because of the advantages that they offer.

Bibliography

- [1] SethBling, *MarI/O - Machine Learning for Video Games*, [Online; accessed 27-January-2020], Youtube, 2015, URL: <https://www.youtube.com/watch?v=qv6UVOQ0F44> (cit. on p. 1).
- [2] CERN, *Processing: What to record?*, [Online; accessed 27-January-2020], 2020, URL: <https://home.cern/science/computing/processing-what-record> (cit. on p. 1).
- [3] T. Alef, *Incorporation of Systematic Uncertainties in the Training of Multivariate Methods*, tech. rep. BONN-IB-2019-04, Physikalisches Institut, 2019 (cit. on p. 1).
- [4] Wikipedia contributors, *Standard Model of Elementary Particles*, [Online; accessed 05-January-2020], 2019, URL: https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg (cit. on p. 4).
- [5] M. Tanabashi et al., *Review of Particle Physics*, Phys. Rev. D **98** (3 2018) 030001, URL: <https://link.aps.org/doi/10.1103/PhysRevD.98.030001> (cit. on pp. 4, 5).
- [6] Wikipedia contributors, *Large Hadron Collider*, [Online; accessed 10-February-2020], 2019, URL: https://en.wikipedia.org/wiki/Large_Hadron_Collider (cit. on p. 7).
- [7] CERN, *key achievements*, [Online; accessed 10-January-2020], 2020, URL: <https://home.cern/about/key-achievements> (cit. on p. 7).
- [8] Education, Communications and Outreach Group CERN, *LHC faq the guide*, Brochure, 2017, URL: <https://home.cern/sites/home.web.cern.ch/files/2018-07/CERN-Brochure-2017-002-Eng.pdf> (cit. on p. 7).
- [9] CERN, *Facts and figures about the LHC*, ed. by CERN, 2020, URL: <https://home.cern/resources/faqs/facts-and-figures-about-lhc> (cit. on p. 8).
- [10] ATLAS Outreach, “ATLAS Fact Sheet : To raise awareness of the ATLAS detector and collaboration on the LHC”, 2010, URL: <http://cds.cern.ch/record/1457044> (cit. on p. 7).
- [11] J. Pequeno, *Computer generated image of the whole ATLAS detector*, [Online; accessed 21-January-2020], 2008, URL: <https://cds.cern.ch/images/CERN-GE-0803012-01> (cit. on p. 9).
- [12] J. Pequeno, *Computer generated image of the ATLAS inner detector*, [Online; accessed 21-January-2020], 2008, URL: <https://cds.cern.ch/images/CERN-GE-0803014-01> (cit. on p. 10).
- [13] M. Thomson, *Modern Particle Physics*, Cambridge University Press, 2013 (cit. on p. 11).

- [14] B. Reder, *Studie Machine Learning Deep Learning 2019*, Study, 2019, URL: https://blog.techdata.de/app/uploads/2019/04/Tech-Data_IDG_ML-DL-Studie2019.pdf (cit. on p. 13).
- [15] S. Byford, *How AI is changing photography*, [Online; accessed 03-February-2019], 2019, URL: <https://www.theverge.com/2019/1/31/18203363/ai-artificial-intelligence-photography-google-photos-apple-huawei> (cit. on p. 13).
- [16] Wikipedia contributors, *Machine learning*, [Online; accessed 15-January-2020], 2019, URL: https://en.wikipedia.org/wiki/Machine_learning#Applications (cit. on p. 13).
- [17] The University of Queensland, *How do neurons work?*, ed. by The University of Queensland, 2020, URL: <https://qbi.uq.edu.au/brain-basics/brain/brain-physiology/how-do-neurons-work> (cit. on p. 15).
- [18] Wikipedia contributors, *Activation function*, [Online; accessed 08-January-2020], 2019, URL: https://en.wikipedia.org/wiki/Activation_function (cit. on p. 16).
- [19] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 2014, arXiv: 1412.6980 [cs.LG] (cit. on p. 19).
- [20] *Observation of the associated production of a top quark and a Z boson in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, tech. rep. ATLAS-CONF-2019-043, CERN, 2019, URL: <https://cds.cern.ch/record/2690716> (cit. on p. 27).
- [21] J. Conrad, *Lorentz Invariance and the 4-vector Dot Product*, [Online; accessed 03-February-2019], 2010, URL: https://uspas.fnal.gov/materials/12MSU/Conrad_4vector.pdf (cit. on p. 27).
- [22] *Keras: The Python Deep Learning library*, [Online; accessed 27-January-2020], Keras, 2020, URL: <https://keras.io/> (cit. on p. 29).
- [23] P. Nogga, *Multivariate Analysis of a Top Quark Production with an associated Z Boson at the ATLAS Experiment corresponding to $\sqrt{s} = 13$ TeV*, Bachelorthesis: Rheinische Friedrich-Wilhelms-Universität Bonn, 2020 (cit. on p. 35).
- [24] J. Brownlee, *A Gentle Introduction to k-fold Cross-Validation*, [Online; accessed 03-February-2019], 2019, URL: <https://machinelearningmastery.com/k-fold-cross-validation/> (cit. on p. 35).
- [25] R. Karim, *Intuitions on L1 and L2 Regularisation*, [Online; accessed 03-February-2019], 2018, URL: <https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261> (cit. on p. 35).

List of Figures

2.1	Summary of the standard model of particle physics	4
3.1	Overview of CERNs accelerator complex	8
3.2	Overview of the ATLAS detector	9
3.3	Structure of the inner detector	10
3.4	Scheme of typical particle detection	11
4.1	Overview of a neural network	14
4.2	Depiction of a basic node	14
4.3	Example of overfitting	20
4.4	General sketch for dropout	20
4.5	Behavior of different learning rates	21
5.1	tZq Feynman diagrams	24
5.2	Full tZq decay chain	25
5.3	Feynman diagrams of tZq background	26
6.1	Comparison of Adam and SGD	30
6.2	Grid searches for different hyperparameters	32
6.3	Differences between the 2j1b and 3j1b signal region	33
6.4	Differences of the performance between the 2j1b and 3j1b signal region	34
6.5	Performance of the variables from table 5.1	35

List of Tables

4.1	Examples of different activation functions	16
4.2	Confusion matrix	17
5.1	Variables for tZq training	27
5.2	Lorentz invariant variables for tZq training	28
6.1	Hyperparameters for different signal regions	32

Acronyms

ALICE A Large Ion Collider Experiment. 7, 8

ATLAS a toroidal LHC apparatus. 7–11, 33

AUC area under curve. 17

CERN Organisation européenne pour la recherche nucléaire. 7, 8

CMS Compact Muon Solenoid. 7, 8

FN false negative. 15, 17

FP false positive. 15, 17

FPR false positive rate. 15, 17

LHC Large Hadron Collider. 7, 8

LHCb Large Hadron Collider beauty. 7, 8

ROC Receiver Operating Characteristic. 15, 17

SCT Semi Conductor Tracker. 8

SGD Stochastic Gradient Descent. 18, 19, 21, 29

SM Standard Model. 3, 4, 23

TN true negative. 15, 17

TP true positive. 15, 17

TPR true positive rate. 15, 17

TRT Transition Radiation Tracker. 8, 10