TECHNISCHE
UNIVERSITÄT
DRESDEN

Study of the production of two same-charge polarised W bosons via vector boson scattering with neural networks

Bachelor-Arbeit
zur Erlangung des Hochschulgrades
Bachelor of Science
im Bachelor-Studiengang Physik

vorgelegt von

Lisa Lehmann

geboren am 20.08.1999 in Hoyerswerda

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden

2021

Eingereicht am 07. Juni 2021

1. Gutachter:	Prof. Dr. Michael Kobel
2. Gutachter:	Dr. Karolos Potamianos
Betreuer:	Dr. Karolos Potamianos

Summary

Abstract:
The Standard Model of particle physics (SM) predicts in what way elementary particles can interact. One such interaction is vector boson scattering (VBS) of $W^{\pm}W^{\pm}jj$. In studying this VBS process we are able to probe electroweak symmetry breaking (EWSB) as a key feature of the SM directly. EWSB is how the W boson acquires mass and a longitudinal polarisation component. In the SM fine-tuned interferences prevent VBS processes from violating unitarity. This study uses Monte Carlo (MC) simulations of proton-proton collisions with VBS signature as input data for neural networks (NNs). The goal is to discriminate between VBS events with two longitudinally-polarised W bosons, $W_L^{\pm}W_L^{\pm}$, from $W^{\pm}W^{\pm}jj$ events where at least one of the W bosons is transversely polarised. While validating the MC data, tests on different observable distributions indicate inconsistencies towards the expected behaviour, which need to be further investigated. This work presents different NN-topologies which are able to distinguish well between the signal and background processes. In further research the idealistic dataset of only $W^{\pm}W^{\pm}jj$ events needs to be extended by more background processes and detector simulation for the NNs to work with experimental data.

Zusammenfassung:
Das Standardmodell der Teilchenphysik (SM) sagt vorher wie Elementarteilchen miteinander wechselwirken können. Eine dieser Wechselwirkungen ist die Vektor Boson Streuung (VBS) von $W^{\pm}W^{\pm}jj$. Mithilfe dieses Prozesses ist es möglich die elektroschwache Symmetriebrechung (EWSB) genauer zu untersuchen. Im SM sorgt EWSB dafür, dass W Bosonen eine Masse sowie eine longitudinal polarisierte Komponente erhalten. Fein abgestimmte Interferenzen halten VBS Prozesse des SM davon ab die Unitarität zu verletzen. In dieser Arbeit werden Monte Carlo (MC) Simulationen von Proton-Proton Kollisionen mit einer Signatur von VBS Prozessen als Eingabedaten für Neuronale Netze (NN) verwendet. Es sollen VBS Ereignisse mit zwei longitudinal polarisierten W Bosonen, $W_L^{\pm}W_L^{\pm}$, von $W^{\pm}W^{\pm}jj$ Ereignissen mit mindestens einem transversal polarisierten W Boson voneinander getrennt werden. Bei der Validierung der MC Daten mithilfe von Tests an verschiedenen Observablen zeigen sich leichte Abweichungen vom erwarteten Verhalten gefunden, welche in weiteren Untersuchungen genauer verstanden werden müssen. In dieser Arbeit werden verschiedene Topologien von NN vorgestellt, die gut zwischen den Signal- und Hintergrundprozessen unterscheiden können. Zukünftige Forschung wird diese idealisierte, auf $W^{\pm}W^{\pm}jj$ Prozesse beschränke Studie für NN mit weiteren Hintergrundprozessen und Detektorsimulationen erweitern.

Contents

1 Introduction **1**

2 Introduction to Particle Physics **3**

2.1 The Standard Model of particle physics 3

2.2 Vector-Boson Scattering and Polarisation 5

2.3 The ATLAS-Detector at the Large Hadron Collider 6

3 Machine Learning **9**

3.1 Basic Introduction and terminology 9

3.2 Neural Networks 10

3.3 Performance measures and figures of merits 14

3.3.1 Learning curves 14

3.3.2 ROC curve and the area under the curve 15

3.3.3 Significance as figure of merits 16

4 Event simulation and analysis **19**

4.1 Event Generation with Monte Carlo Generators in the Athena-Framework . . 19

4.2 Data Analysis and Closure tests 20

4.2.1 Rivet Analysis 21

4.2.2 Cross section closure 21

4.2.3 Decay Angle Distributions 21

4.2.4 Pairflavor Variable 22

4.3 Selection of polarisation sensitive variables 24

5 Performance of Neural Networks **27**

5.1 Figures of merits 27

5.2 Implementation of neural networks in Python 27

5.3 Promising models 28

6 Summary and Outlook **33**

A Appendix **35**

A.1 Activation Functions 35

A.2 Backpropagation 37

A.3 Polarisation sensitive variables - plots 39
A.4 Implementation of Neural Networks with Keras 44

B Bibliography **47**

1 Introduction

Physics is a broad field of astonishing research. Some physicists look at the electronic and magnetic properties of solid and condensed matter [1], others search ways to describe biological processes, like the walk of a kinesin molecule [2],[3]; some even try (and succeed!) to build computers based on basic quantum mechanic principles [4].

On scales as small as 10^{-18} m physicists try to understand and describe the most fundamental constituents of our universe. Over the last hundreds of years this scale had to be lowered repeatedly after big revelations on what the smallest, non-dividable, also called *elementary*, particle was at that time. During this process different particles were discovered, such as the electron [5] or the proton [6]. Further research showed that even protons and neutrons, which form the atomic nucleus, are bound particles consisting of quarks. This whole zoo of particles and their possible interactions is described within the best general theory we have so far: the Standard Model of particle physics (SM).

The SM originated in the 1960s and 1970s [7] and a lot of its predictions were confirmed since then. The last predicted particle, the Higgs boson, was discovered in 2012 by the ATLAS and CMS experiments at the Large Hadron Collider (LHC) [8]. Since then, a lot of effort was put into measuring predictions of different kinds of particle interactions. The SM is a very successful model of nature, yet it doesn't explain several cosmological observations which can only be explained using concepts like dark matter and dark energy, which have not yet been confirmed. This prompts physicists to propose (and search for) new theories that go beyond the SM and explain these observations. For these purposes very exact measurements, like the g-2 experiments at Fermilab [9], have to be done, to either confirm predictions of the SM or hint towards physcics beyond.

A key concept in the SM is Electroweak Symmetry Breaking (EWSB). EWSB is implemented in the SM through the Brout-Englert-Higgs (BEH) field going from a symmetric state to an unsymmetric state, triggering mechanisms that give particles, especially vector bosons, their masses. The vector bosons make interactions between particles possible and are predicted to interact with each other (self-coupling). Vector boson scattering (VBS) is sensitive to EWSB. An intersting VBS process, because of its low background, is the scattering of two W bosons $W^{\pm}W^{\pm}jj$ which was first experimentally observed in 2017 [10],[11].

This thesis focus is to study ways to discriminate between events where two W bosons with longitudinal polarisation (LL) scatter with each other $\left(W_L^{\pm}W_L^{\pm} \longrightarrow W_L^{\pm}W_L^{\pm}\right)$ from events where

one (TL), or two (TT) of the bosons are transversely polarised. Polarisation, especially LL, is of interest because it is a direct consequence of EWSB. The study is done with the help of Machine Learning tools such as Neural Networks (NNs). NNs are very powerful at classifying labelled events and therefore are suited for the problem of separating LL from TT/TL events. The work on NNs for discrimination between polarised scattering in [12] served as main inspiration for this thesis' topic.

The thesis is structured as follows: In **Chapter 2** an introduction to particle physics is given. The SM is presented, as well as the process of VBS of these peculiar polarised W bosons. After that the ATLAS detector at the Large Hadron Collider (LHC) is introduced as an example of how to measure these scattering processes. **Chapter 3** explains the basic terminology of machine learning, what NNs are and how they work. We also discuss several performance metrics such as a simplyfied significance and the area under a classification performance curve. A methodic overview of the tools and observables used in this thesis is given in **Chapter 4**. The results of this work are shown in **Chapter 5**. Finally **Chapter 6** provides a summary of this thesis' work and gives an outlook.

2 Introduction to Particle Physics

In particle physics the leading question, asked on the smallest scales, is "How does matter interact and what is it made of?". In search of an answer, physicists are concerned with the fundamental constituents of matter in the universe, the elementary particles, and the interactions between them, the forces. Section 2.1 introduces the Standard Model of particle physics that best describes the experimental data since the 1970s. In section 2.2 the process of vector bosons scattering is introduced. A good theory needs to be tested as far as possible. This is done, among other places, at the ATLAS experiment at CERN. In section 2.3 it is introduced as a main tool for experimentalists.

2.1 The Standard Model of particle physics

A theory describing interactions between particles emerged in the 1960s and 1970s. It was called the *Standard Model of particle physics* (SM). Unfortunately, gravity is not yet included, neither describes the SM everything we know and have observed yet. Still, it is the best description we have so far.
Matter can be divided into two kinds of particles. Particles of integer spin are called *bosons* and particles with a spin multiple to $\frac{1}{2}$ are called *fermions*. The SM distinguishes fermions into *quarks* and *leptons*. Both carry the *electromagnetic charge* and a *weak charge*, but only quarks carry a *color charge*. There exist three generations of leptons and quarks, where each generation is a heavier copy of the one before. Figure 2.1 shows these particles, the quark flavors in purple and the lepton flavors in green, each column represents a generation. Every particle has its anti-particle, differing only in the charge it is carrying, which is opposite. If we count the leptons, quarks and their respective anti-particles, the SM contains 48 fermions.

In the universe exist, as far as we know, four fundamental forces: the *electromagnetic* force, the *strong* force, the *weak force* and the *gravitational* force. The first three are described within the SM. A quantum theory that includes the very weak gravitational force is yet to be found. Each of these forces is described by its own theory, a so called *Quantum Field Theory* (QFT), thus the SM is a collection of these. QFTs combine quantum mechanics with special relativity, which is important for the very small and usually very fast moving elementary particles. These theories describe forces in terms of particle exchange, which means that the interaction be-

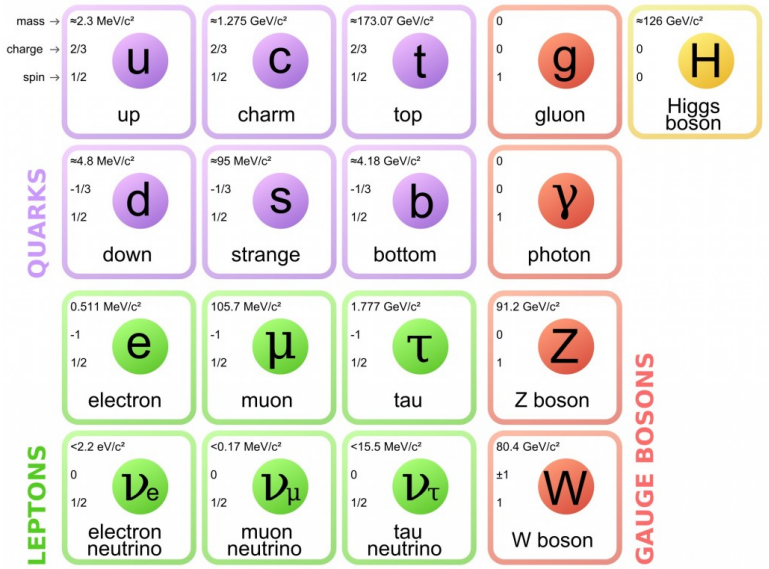


Figure 2.1: Overview of all particles in the Standard Model. The charge is only referring to the electromagnetic charge. Graphic taken from [13].

tween different particles is mediated by a "force-carrying" particle: the mediator, called *gauge boson* or *vector boson*. These bosons are spin-1 particles.

The theory describing electromagnetic interactions is called *Quantum Electrodynamics* (QED) and is mediated by the massless *photon*. The strong force is described by the theory of *Quantum Chromodynamics* (QCD) and is mediated by the massless *gluons*. It is responsible for the cohesion of protons and neutrons that form the atomic nuclei. The weak interactions are mediated by three vector bosons, an electrically neutral *Z boson* and the two electrically-charged *W bosons*. The W and Z bosons are massive with masses of $m_W = 80.379 \pm 0.012\,\text{GeV}$ and $m_Z = 91.1876 \pm 0.0021\,\text{GeV}$ [14]. The *Glasgow-Weinberg-Salam* (GWS) theory describes the weak interaction and combines it with the electromagnetic interaction into a single electroweak force. The weak force is responsible for the decay of particles like the pion or the nuclear beta decay. It is the only force that allows a change in flavor.

Particles can only participate in a specific interaction if they carry the respective charge. That means that for example leptons cannot interact via the strong interaction but all fundamental particles can participate in the weak interaction.

An important part of the SM is the *Brout-Englert-Higgs (BEH) mechanism*. It describes a BEH-field that is shaped like a sombrero, which causes *spontaneous symmetry breaking* below a critical temperature. Through coupling to the BEH-field the bosons acquire mass via the Higgs-mechanism. A key feature of this mechanism is the in the 1960s postulated *Higgs boson*, an excitation of the BEH-field. With its discovery in 2012 by the ATLAS and CMS experiments the SM was validated further.

A more complete introduction, also to the mathematical concepts, can be found in [15],[16].

A great didactically approach is given in [17].

2.2 Vector-Boson Scattering and Polarisation

Two particles scatter with each other when they interact via the exchange of a mediator. According to the SM, some of the mediators even scatter with themselves in so called self-couplings. The scattering of vector bosons is thus called *vector bosons scattering* (VBS). For W bosons it was first observed by the ATLAS and CMS experiment in 2017 [10],[11]. This was a significant challenge since the process is extremely rare: in over 10^{14} collisions only 5 have a signature sensitive to VBS.
At the LHC, as a proton-proton collider, the W bosons are radiated off the protons' quarks. The VBS process signature commonly used for observations is an electroweak process with the W bosons interacting with each other and then decaying fully leptonically into an electron, muon or taoun and the respective neutrino. Figure 2.2 shows a schematic diagram of this process. The hashed circle represents the different possibilities for the W bosons to interact. The diagrams that could be inserted into the blob are shown in Figure 2.3.

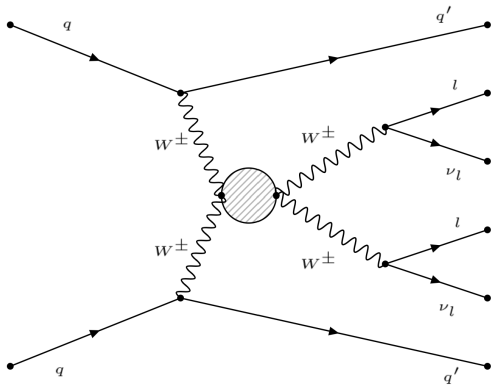


Figure 2.2: Schematic diagram of the vector boson scattering process at the LHC. The dashed circle stands for the possible interactions between the vector bosons shown in Figure 2.3.

Since VBS processes are sensitive to several predictions of the SM and models beyond the SM, it is an ongoing field of research to measure specific quantities of the scattering process.

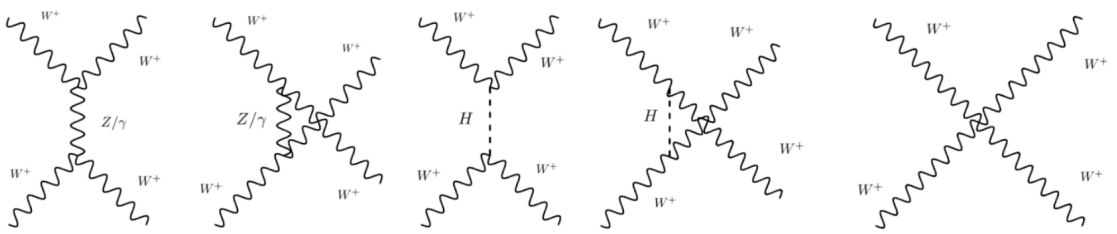


Figure 2.3: Possible ways the W bosons can interact with each other.

One such quantity that allows to probe the SM directly is the *polarisation* of the individual bosons. It is defined as the normalised projection of the spin onto the direction of motion of the particle, also called *helicity*.

$$h = \frac{\mathbf{S} \cdot \mathbf{p}}{|\mathbf{p}|} \tag{2.1}$$

If the spin is perpendicular to the direction of motion, h is equal to 0, and the state is called *longitudinally polarised*. For the cases where the spin is parallel or anti parallel to the direction of motion, the state is called *transversely polarised*, $h = \pm 1$.

It can be shown that if there were no couplings of the Higgs boson to the other bosons, the cross section of the scattering process of only longitudinal polarised Bosons $W_L^\pm W_L^\pm \to W_L^\pm W_L^\pm$ would diverge at large energies. In a physical theory this should not happen, since it is violating unitarity, an important condition in QFTs.
Being able to measure this process is a motivation in the study of VBS processes. To differentiate between the longitudinally polarised bosons and the transversely polarised bosons, it is important to know the fraction of events of each polarisation in $W^\pm W^\pm jj$-VBS where both bosons will be longitudinally polarised. The theory predicts, that the fraction of longitudinally polarised bosons should be about 5-10% of all $W^\pm W^\pm jj$ events.

2.3 The ATLAS-Detector at the Large Hadron Collider

Although this work does not use experimental data, it is important to understand the actual experiment, since the goal is to continue this study on simulated data and apply it to observed data from ATLAS.
The *Large Hadron Collider* (LHC) is a hadron accelerator and collider [18] and installed in a 26.7 km long tunnel about 100 m beneath Geneva in Switzerland.
The unit $barn^{-1}$ is a measure for amount of data called *integrated luminosity* L. It is the time integral over the luminosity, which describes the number of particle collisions per area and time and depends on different beam properties. Together with the *cross section* of an event, the probability that a process will occur, we calculate the event rate via $N_{event} = L \cdot \sigma_{event}$. That way we can predict how many events of a given process with a given cross section will occur per unit of time or have occurred in total. Until today 147 fb^{-1} of data were recorded which corresponds to approximately 200 $W^\pm W^\pm jj$ events.
The aim of the LHC is to test the SM further, e.g. with the detection of the Higgs Boson in 2012 [8], and to reveal new physics. Four different experiments are installed at the four collision points of the LHC: the general-purpose detectors ATLAS and CMS as well as the more specialised ALICE and LHCb experiments. ATLAS and CMS are located on opposite sides of the ring. The idea behind that is, what one experiment detects, the other one should find as well.

The acronym *ATLAS* stands for "A Toroidal LHC ApparatuS" which describes the detector structure already a bit. In its cylindric shape different types of detectors, each able to measure specific properties of particles, are aranged in layers around each other. The overall structure of the ATLAS detector is shown in Figure 2.4. This introduction follows [19].
In the ATLAS detector a particular coordinate system is used. The interaction point is set as the origin. The z-axis is defined as the beam direction, the positive x-axis points to the center of the ring and the positive y-axis points upwards. The angles for the cylindrical coordinate system are defined accordingly, so that the azimuthal angle ϕ is measured around the beam axis and the polar angle θ is measured from the beam axis. Other coordinates allow for easier orientation and separation of events: The pseudorapidity is defined as $\eta = -\ln\tan(\frac{\theta}{2})$ and the rapidity, used for massive objects like jets[1], is defined as $y = \frac{1}{2} \cdot \ln((E + p_z)/(E - p_z))$. The distance in the ϕ- and y-space is defined as $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$.
It is worth to mention the challenges in the data acquisition. Proton-proton collisions happen with a frequency of up to 1.7 GHz while the data recording system is only capable of handling about 1 kHz. To deal with the amount of events that have to be rejected per second a complex *trigger* system was created [20].
After a collision at the origin of the coordinate sysem, the resulting particles will propagate through the different parts of the detector, which are the inner detector, the calorimeters and the muon system.

The inner detector is embedded in a 2 T axial magnetic field. It is designed to measure the tracks of charged particles which is necessary for the momentum measurement. The innermost layers consist of different precision tracking detectors, the *Insertable B-Layer* (IBL), *Pixels* and the *Semiconductor Tracker* (SCT). They are arranged on concentric cylinders around the beam axis and cover a region of $|\eta| < 2.5$. The outer part is build of 4 mm straw tubes that enable track following up to $|\eta| = 2.0$, called *Transition Radiation Trackers* (TRT).

The Electromagnetic and Hadronic Calorimeters cover the range of $|\eta| < 4.9$. Over the η region that matches the inner detector the fine granularity of the electromagnetic (EM) calorimeter is suited for precision measurements of electrons and photons. The coarser granularity of the rest of the calorimeter system satisfies the requirements for jet reconstruction and missing transverse energy measurements. In general calorimeters provide a good containment for EM and hadronic showers. The EM calorimeter in the inner part covers in the barrel region $|\eta| < 2.5$ and on the end-caps a region of $|\eta| < 3.2$ in total. It consists of accordion-shaped kapton electrodes of Liquid Argon and lead absorber plates. In a next layer, the *Tile Calorimeter* and the *LAr hadronic end-cap Calorimeter* (HEC), which are sampling calorimeters, enclose

[1] particles produced in the hadronisation process of quarks that propagate in similar directions

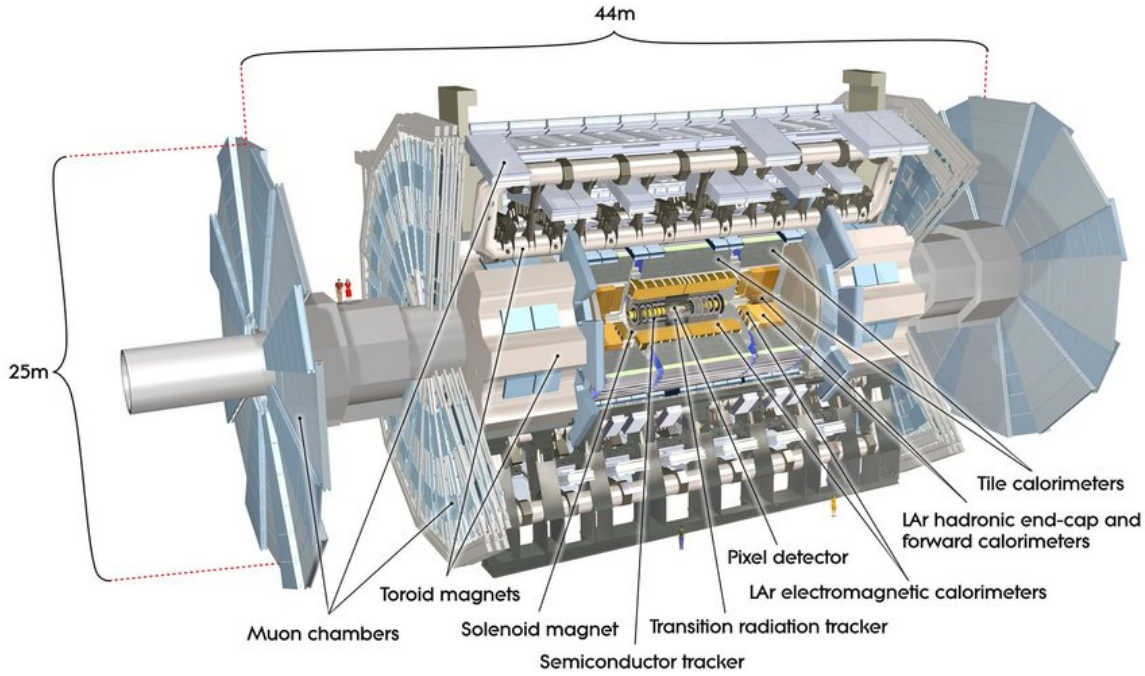


Figure 2.4: Cut-away view of the ATLAS detector. The dimensions of the detector are 25 m in height and 44 m in length. The overall weight of the detector is approximately 7000 t. Picture taken from [19].

the inner part together with the *LAr forward Calorimeter*. It consits of three modules, where one is optimised as EM calorimeter, the other two are perfect for hadron energy measurements.

The Muon Spectrometer is the outer layer of the ATLAS detector. It's task is to detect the mouns that leave only a track in the inner detector. It is surrounded by large superconducting air-core toroid magnets so that the muon tracks are bending in the resulting magnetic field. Since a high level of particle flux is expected the system is instrumented with a seperate trigger system and high precision tracking chambers.

3 Machine Learning

In section 3.1 we introduce the concept of machine learning (ML) as well as the basic terminology to be prepared to understand what Neural Networks (NNs) are and how they work in section 3.2. Most explanations have been inspired by [21, 22, 23].

3.1 Basic Introduction and terminology

Machine learning (ML) is the science (or even art) of programming computers so they can learn from experience [21]. Take as an example the tasks of a spam filter: It has to decide on a daily basis whether an incoming mail is spam or ham. It needs to keep in mind how a spam mail is structured. The filter also needs to adapt to new kinds of mails (e.g. new syntax, another prince offering you a lot of money, ...). In other words it has to improve automatically through experience.
ML receives so much attention because it is able to tackle difficult problems. There are problems where the existing solutions require a lot of fine tuning and rule-obeying or complex problems where a traditional approach does not yield a good solution or no solution at all. In those (and many more) cases some algorithm that can learn these rules or is able to detect these patterns will perform way better than a human [24]. Other use-cases are fluctuating environments where the algorithm has to adapt to new data very quickly.

The goal of many ML algorithms is to find a relation between the inputs x and outputs, here also called labels, y. They approximate an underlying function $f(x) = y$ with which it can make assumptions on new data. If the algorithm knows the actual labels of the training data y_{label} it is called *supervised learning*, if it doesn't, it is called *unsupervised learning*.
Generalisation on new data can happen in two ways: Either the system learns examples by heart and generalises to new data by using a similarity measure to compare them to the learned examples, called *instance based learning*, or it builds some kind of model from the examples and uses that model to make predictions, called *model based learning*.
If the function $f(x)$ is discrete, the algorithm performs a *classification task*; Remember our spam filter that classifies mails into spam and ham (or 0 and 1). Predicting the shape of a continuous distribution is called a *regression task*.

In this thesis supervised model-based learning will be used to perform a classification task. As sketched in the introduction to VBS (section 2.2), we want to be able to separate longitudinally polarised W bosons from the rest of the data. The algorithm will have to distinguish between the signal process (the longitudinally polarised W bosons) and the background (every other possible process, e.g. transversely polarised W bosons).

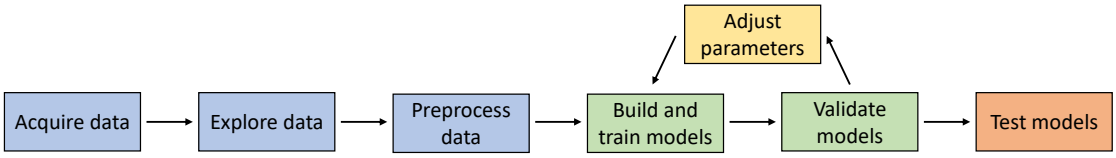


Figure 3.1: Typical structure of a machine learning project. This picture was taken from the lecture series of M. Schwarzenberger.

A typical **structure of a ML project** as shown in Figure 3.1 consists of the following steps:

1. Acquire data
2. Explore data: Do the variables we can acces look as expected? Can we find some strong correlations between them ourselves?
3. Preprocess data: Normalise the data, fix NaN values, throw away not useful data, etc. Split the data into a sample with training data and a sample of test data that our model will get to see only in the end to measure how good it generalises. Split training data into "real" training data and validation data, with which we can measure the ability to generalise during training.
4. Build and train different models: judge them according to their learning curves
5. Validate the models: via the validation data according to their learning curves - if they perform too bad, reject the models and start again at 4.
6. Test models: on test data. If it performs badly, reject it, else it's good to go.

The process described in point 4 and 5 is called *training*. The goal during training is to minimise the so called error of the ML algorithm, i.e. to minimise the amount of wrong predictions. This is done using the gradient descent algorithm, which is explained in detail for NNs in section 3.2.

3.2 Neural Networks

A *Neural Network* (NN) is a model inspired by the networks that biological neurons build in the human brain. NNs consist of different nodes which are connected to each other and

ordered in layers. The nodes are called neurons, following the biological interpretation, that have to reach a potential z in order to be activated.

A basic structure is shown in Figure 3.2. The circles on the left represent the input neurons, all together they build the *input layer*. The circle on the right denotes the output neuron[1] building the *output layer*. The layers in between are called *hidden layers*. The *weighted connections* between the layers show how strongly the neurons of the previous layer influence the neurons in the next layer. Additionally each neuron has a specific *bias*, that gives a threshold on when the neuron should become activated. A NN is called *fully connected* or *dense*, if all neurons in each layer are connected to all neurons in the next layer, as shown in Figure 3.2.

The output of a neuron is the output of an *activation function* Ψ where its input is the weighted sum of the input vectors, like it is shown in Figure 3.3. Which function is used depends on the neuron being in the output or hidden layer. Most common activation functions Ψ are: the sigmoid function σ, the ReLU function, the tanh and the softmax function. Depending on how fast a NN shall be trained and what kind of task the output neuron(s) have to do, activation functions Ψ perform quite differently. Usually a ReLU function is used in the hidden layers and a sigmoid or softmax function in the output layer. A more detailed summary including the definitions and a graphical visualisation of activation functions can be found in the appendix A.1.

A NN is trained in two main steps: First we compute the output of the network, via computing the activation of each neuron as shown in figure 3.3, then we use gradient descent with back-propagation to update the parameters, which are the weights and biases. This explanation follows the main steps outlined in [22].

To compute the output or *activation* a of a NN we have to keep track in which layer l the neuron is:

[1]There can be more than one output neuron.

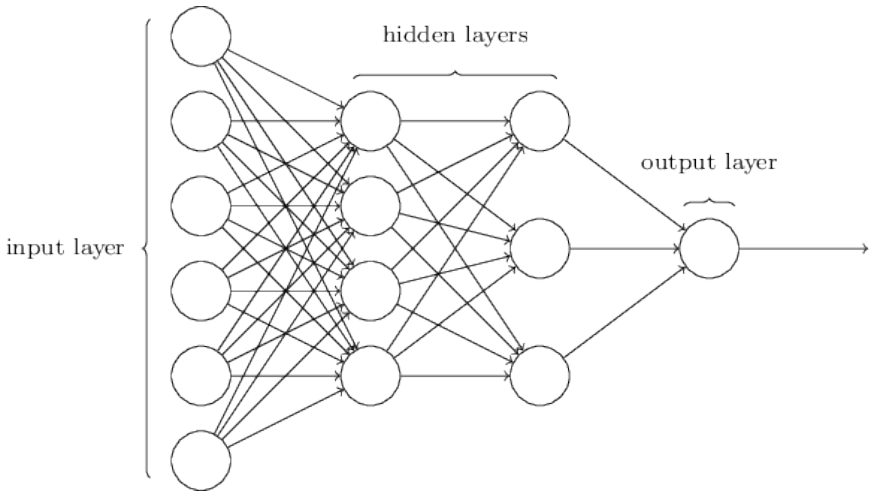


Figure 3.2: Terminology and structure of a Neural Network. Graphic taken from [22].

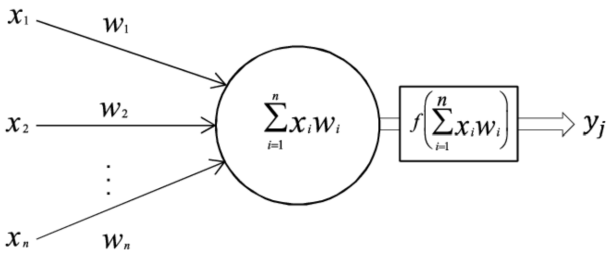


Figure 3.3: Calculation of the output of one neuron. Graphic taken from [25].

$$a_j^l = \Psi(\sum_k w_{jk}^l \cdot a_k^{l-1} + b_j^l) \tag{3.1}$$

Here a_j^l denotes the activation of the j^{th}-neuron in the l^{th}-layer, b_j^l similarly the bias of the j^{th}-neuron in the l^{th}-layer and w_{jk}^l the weight of the connection from the k^{th}-neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th}layer. Therefore a_j^l depends on all activations of the k-neurons in the previous layer.

We will denote x as the training input, for example a vector of 10 different variables, and $y(x) = y$ as desired output, in our case if x belongs to a longitudinal polarised W boson event. The goal of training is to find weights and biases in such a way that the output of our network is close to the desired output. To see how well the NN already performs we use a *loss function* (cost function) $L(w, b)$, depending on the weights and biases. Our loss is small, $L(w, b) \approx 0$, if the prediction is close to the actual label and larger the further away it is. We achieve the goal of minimising $L(w, b)$ by using an algorithm called *gradient descent,* since finding a minimum analytically is a very difficult problem. Keep in mind that there are $j \cdot k$ weights between the layers - so there are many parameters that contribute to the loss functions value.

A way to understand the gradient descent algorithm is to imagine a loss function depending on two variables, v_1 and v_2. The function describes a surface in three dimensions. If we imagine a ball starting to roll down from a random point, it will eventually stop at the bottom. This rolling down can be described in terms of "steps" the ball is making on its way, which will always be in direction of the steepest descent, so to speak in direction of the negative gradient. After a "step" the loss changes. For our ball we could define the "law of motion" as follows

$$\mathbf{v} \to \mathbf{v}' = \mathbf{v} - \eta \nabla L \tag{3.2}$$

where $\mathbf{v}$ is the position vector and η is the step size, called *learning rate.* If we update our position over and over again like described in Equation 3.2 we will reach a - hopefully global - minimum. Basically the same works when we need to find the minimum in our high-dimensional parameter space (depending on w, b), only that the image of a ball on a for example 13,000 dimensional surface isn't the most intuitive.

To apply gradient descent in a NN we need to rewrite Eq. 3.2 in a way that updates the weights w_k and biases b_l, which now represent the position of the ball. This means that the gradient vector ∇L has now components $\partial L/\partial w_k$ and $\partial L/\partial b_l$. In practice, to compute the gradient ∇L we need to compute the gradients ∇L_x separately for each training input x and then average them $\nabla L = \frac{1}{N}\sum_x \nabla L_x$. If the number of training inputs is large this will take a lot of computational resources. That is why often the *Stochastic Gradient Descent*(SGD) algorithm is used, which takes random samples of size m out of our input data and estimates the gradient on these. The small number of random samples is called a *mini-batch*. Provided the sample size is large enough, the averaged value of the gradient of the mini-batch will roughly equal the actual gradient $\nabla L \approx \frac{1}{m}\sum_{j=1}^{m} \nabla L_{x_j}$. With that said we can rewrite Eq. 3.2 for our NN:

$$w_k \to w_k' = w_k - \frac{\eta}{m}\sum_j \frac{\partial L_{x_j}}{\partial w_k} \tag{3.3}$$

$$b_l \to b_l' = b_l - \frac{\eta}{m}\sum_j \frac{\partial L_{x_j}}{\partial b_l} \tag{3.4}$$

Here w_k and b_l represent all the weights and biases in our NN and the sum goes over all training examples in the current mini-batch.

After one such step, i.e. compute the activation of each neuron, determine the loss and update the weights and biases for one mini-batchin a SGD step, the algorithm takes the next random chosen mini-batch and trains with those. Once the NN was trained with all training data, it is called an *epoch*. Then it will start all over again with a new training epoch.
As already mentioned, SGD is not always the fastest or most expedient algorithm. For example, it can get stuck on a saddle point of the loss function L or oscillate in a deep valley. Here the image of the ball with a given momentum and air resistance comes in handy again. There are algorithms that optimise the behaviour of SGD, called *optimisers*. The one used in this thesis is called Adam optimiser [26] and leads in training usually to a fast convergence towards a minimum. An introduction to these optimisers is given in [27].

A very important step in the SGD algorithm was not mentioned yet: How is the gradient computed?

This is done via a fast algorithm known as *backpropagation*. It became famous with a paper by Rumelhart, Hinton and Williams in 1986 [28]. The goal of backpropagation is to compute the partial derivates of L with respect to the weights and biases. The algorithm relies on the chain rule. It calculates errors for all neurons in the NN by knowing the analytical derivations of the loss and activation functions. This error then connects the loss function to the respective weights. A detailed explanation of backpropagation is given in the Appendix A.2.

For classification tasks the *cross-entropy* is an appropriate loss function. It is defined as

$$L = -\frac{1}{N}\sum_x \sum_j \left[y_j \cdot \log(a_j^L) + (1 - y_j) \cdot \log(1 - a_j^L)\right] \tag{3.5}$$

where N is the number of training data, the first sum goes over all training inputs and the second sum over all output neurons, y_j is the desired output and a_j^L, the activation of a given output neuron j, is the predicted output. It is the common choice for classification tasks in NNs, because of the properties of its first derivative. When computing the gradient of L it only depends on the input vector x, the desired output y and the activation $a = \Psi(z)$ of the neuron. There are no other derivatives involved, which makes training fast. This loss function is used in this thesis.
There is the possibility that the NN learns the input features only by heart and performs badly on never seen data, called *overfitting*. To prevent models from doing so we use *regularisation* techniques, that restrain the models parameters while training. A first way to prevent overfitting is to compute the loss of the validation data set after each epoch and monitor its development. If it starts to saturate or diverge from the training loss, we stop the training. This strategy is called *early stopping*. Another strategy is to modify the NN, so that in each iteration over a mini-batch the connections of some randomly chosen neurons get temporarily deleted. This technique is called *Dropout*. There are other regularisation techniques where the loss function itself gets modified through a penalty term, called L1 and L2 regularisation, but they are not used in this work.

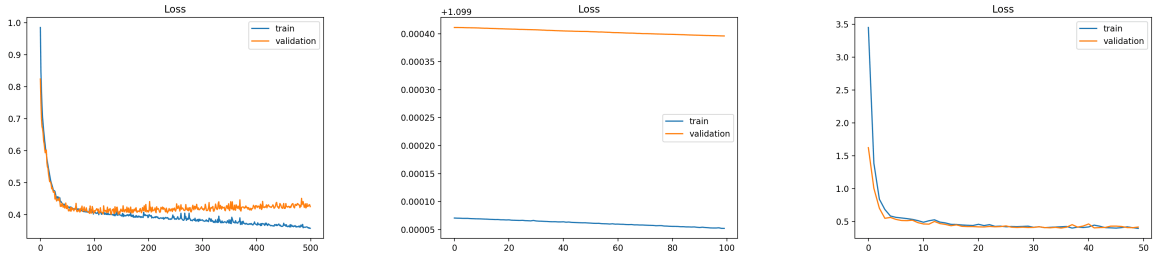
3.3 Performance measures and figures of merits

In the following sections we introduce different performance measures like the learning curves in section 3.3.1, the area under the curve in section 3.3.2 and the significance in section 3.3.3. We will use them to rank different NN in their performance, so that we are able to decide which ones should be preferred.

3.3.1 Learning curves

During training, the performance is monitored with the help of the validation dataset. After each epoch the model makes predictions on the validation data whith the loss calculated as well. That way we can plot the training and validation loss over the evolution of the training, the epochs. These so called *learning curves* give a good measure on the general performance of the model itself. If the validation loss diverges from the training loss our model might be *overfitting* (learning the noise). If the training loss remains flat regardless of training or is not converged yet it might be *underfitting* (not finding all correlations). Examples of such curves are given in figure 3.4. If this happens we need to go back and change parameters of our model

because an over- or underfitting model will not generalise well on new data. If the two losses are close together and converge to a (small) value our model is good to get evaluated on the test dataset, so we get to see how well it performs on actual new (i.e. unseen) data.
Often the learning curves contain another metric as well: the *accuracy*. It is defined as the ratio of the correct predictions to the total number of predictions.



a) b) c)

Figure 3.4: Examples of Learning curves and their interpretation, taken from [23]. a) shows an overfitting model, b) shows an underfitting model and c) shows a good model.

3.3.2 ROC curve and the area under the curve

The *receiver operating characteristic* (ROC) is a measure explicitly for binary classification tasks. It tells us how much of our signal process we can keep under the caveat of how much background we have to accept. See Figure 3.5 for a visualisation.
A *true positive* (TP) is an event of our signal that got correctly identified as such. A *false positive* (FP) is an event of our background that got incorrectly classified as signal. *False Negatives* (FN) and *True Negatives* (TN) are defined accordingly. *Sensitivity* or true positive rate is the proportion of correctly identified signal, $TPR = \frac{TP}{TP+FN}$. *Specificity* or *true negative rate* is the proportion of correctly identified background, $TNR = \frac{TN}{TN+FP}$. For the ROC these quantities are computed with the classifiers output. Therefore a threshold is set successively in each bin to determine the TPR and TNR, see Figure 3.5. The values for each threshold mark an entry in the characteristic. In the end the sensitivity is plotted over the specificity. In other words: on the x-axis the proportion of obtained signal in combination with the rejected background (i.e. correctly classified background) on the y-axis is shown.
The more signal we want to keep the more background we have to keep (sensitivity-specificity-trade-off), thus a perfect classifiers ROC curve would be as much as possible located towards the top right corner.
A good figure of merits is the *area under the ROC curve* (AUC). The closer to 1 the better, if it's close to 0.5 one might as well choose a random classifier.

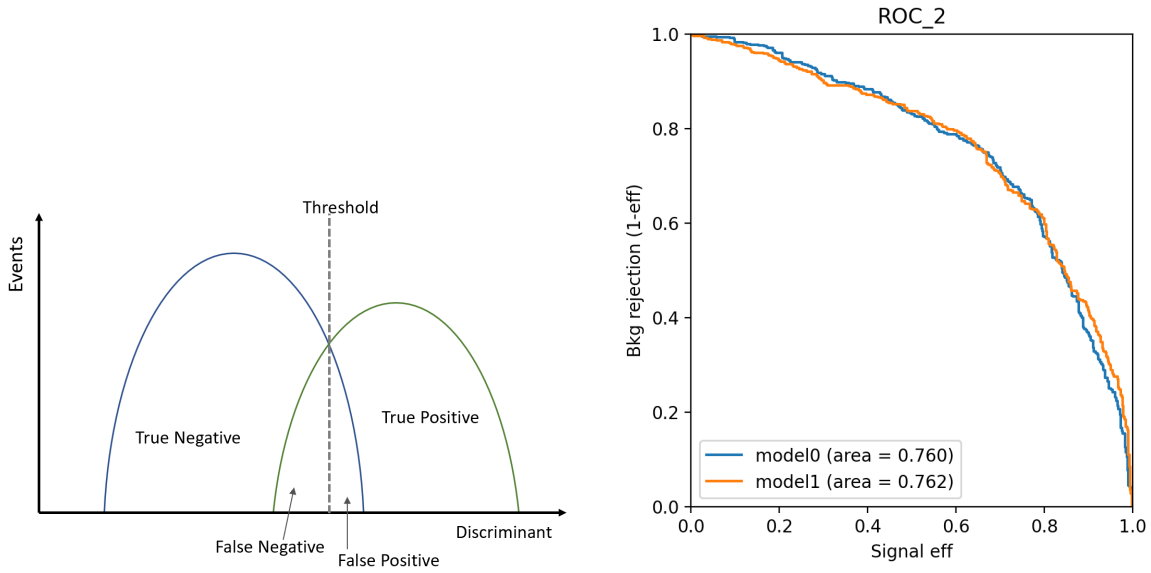


Figure 3.5: On the left is a visualisation of a specific threshold, so that every point under the green curve which is on the left side of the threshold gets classified as the wrong class and vice versa. On the right is a visualisation of the ROC curve, resulting of the threshold setting from the left plot.

3.3.3 Significance as figure of merits

In particle physics there is another widely used statistical measure: the *significance*. It gives us probabilistic insight whether our measured hints for a signal process are only statistical fluctuations.

To explicitly define statistical significance of a measurement a statistical model is needed. This is given by the *Likelihood function* L, depending on parameters of the models hypothesis. In a simplified way it is the product of Poissonian probabilities for all bins in a histogram

$$L(n|\mu,\gamma) = \prod_{b_r \in bins} Pois(n_{b_r}|\gamma_{b_r} \cdot (\mu S_{b_r} + B_{b_r})) \cdot Pois(m_{b_r}|\gamma_{b_r} \cdot \tau_{b_r}) \tag{3.6}$$

where n denotes the number of measured events, γ models uncertainties in each bin, μ is the signal strength, $m_{b_r} = \tau_{b_r}$ number of raw MC events in bin b_r.

This Likelihood function could describe for example a given theory that there is some particle at $m = 20\,\text{TeV}$. To test this assumption a hypothesis is defined. The *null hypothesis* H_0 is that there is no sign of such a new particle. The signal strength has to have a specific value μ, in our case $\mu = 1$. The *alternative hypothesis* is that the signal strength μ is any value $\hat{\mu}$, meaning there might be such a particle.

The *p-value*, a measure on if H_0 should be rejected given a specific threshold of p, is given as

$$p_{value} = \alpha = Pr(t > t_0; H_0) = Pr(|x| > Z; Gaus(x|0,1) \tag{3.7}$$

where t is the test statistic value, t_0 is the measured value of t and Z is the significance. This test statistic should be constructed such, that the null and alt hypothesis are best disentangled. The Neyman-Pearson lemma, Wilks theorem and the asymptotic formula paper [29] give us as an approximated best test statistic the significance in the form of Equation 3.8, with $\mu = 0$ for observation and $\mu = 1$ for limit setting and measurements.

$$Z(n|\mu,\sigma) = \sqrt{-2\ln\frac{L(n|\mu,\hat{\hat{\gamma}})}{L(n|\hat{\mu},\hat{\gamma})}} \tag{3.8}$$

We can split the whole likelihood function L into single bins $\widetilde{L}$ and calculate Z for each bin. To compute Z over all bins one can obtain that it is equal to the squared sum of the single bin significance $Z_{bin,i}$:.

$$Z(n|\mu,\sigma) = \sqrt{-2\ln\prod_i \frac{\widetilde{L}(n_i|\mu,\hat{\hat{\gamma}}_i)}{\widetilde{L}(n_i|\hat{\mu},\hat{\gamma}_i)}} = \sqrt{\sum_i Z_{bin,i}(\mu,\sigma)^2} \tag{3.9}$$

The formula for a single bin can be simplified if systematic uncertainties can be neglected ($\sigma = 0$) and the obtained number of events n is fairly close to the expected number $\nu(\mu) = \mu S + B$, where S is the number of events of the signal process and B includes all background processes:

$$Z_{bin,approc}(n|\mu,0) = \frac{|n-\nu(\mu)|}{\sqrt{\nu(\mu)}} \tag{3.10}$$

If we would now assume to have a signal strength of $\mu = 1$ (this is in our case an overall value, the same for each bin), we would expect to measure $\nu(1) = S + B$ but measure only $n = B$ events. This results in a total Significance of

$$Z = \sqrt{\sum_i (\frac{S}{\sqrt{S+B}})^2} \tag{3.11}$$

The quantity $\frac{S}{\sqrt{S+B}}$ is the expected significance with which one rejects S assuming the signal is absent, and thus can be used to optimize the expected upper limit on s. In other words, the significance we want to use as figure of merits is resulting of a chi-squared distribution with 1 degree of freedom.

4 Event simulation and analysis

In this Chapter we explain how particle physicists acquire simulated data in section 4.1 and how they check if it's behaving appropriately in section 4.2. Finally we will present the variables that were chosen as inputs for the NNs in section 4.3.

4.1 Event Generation with Monte Carlo Generators in the Athena-Framework

In particle physics, if the process of interest is rare, such as VBS, physicists have to wait some years - or even decades - until there is enough experimental data to look at. Even then it might not be useful to look directly at data, since it is without any classification into the different included processes. That is why in high energy physics *Monte Carlo (MC) Generators* play a big role. In short, they use random numbers to simulate collision events, that are fairly similar to experimental data. One *event* is the simulation of one proton-proton collision and the resulting sub-processes. The production of events is based on a theory like the SM. To study the distinguishing of different processes with NNs, MC generators provide labeled samples. Another great thing about MC event generation is that they provide a way to compare the experimental data to the MC samples predicted by theory, thus we can check the theory as well.

A MC generator has to obtain the cross section as probability measure. It is done via numerically approximating the integral of the matrix element for the hard scattering processes in fixed order perturbation theory. The matrix element describes the transition from the initial to final state. Then MC generators have to simulate parton showers with QCD effects, by approximating higher order contributions in perturbation theory of further radiation of particles. For this step the Markov-Chain-Approach is used [30]. With the help of these MC methods we can solve problems without laboriously or non existent analytical solutions.

In this thesis MadGraph5_aMC@NLO [31], a fully automated public event generator, concerned with calculating the matrix element, was complemented with Pythia [32], another commonly used event generator, that is focusing on the simulation of parton showers and soft QCD effects. Information on events that passed the steps of event generation in a generator like MG5 + Pythia8 are referred to as *truth level* or *particle level*. For comparison with

experimental data the simulation of detector effects and the subsequent reconstruction chain is needed, this information then is called *detector-level* or *reconstruction-level*. Within the ATLAS experiment, the Athena Framework is used for large scale data analyses and data generation. It provides the ATLAS simulation and reconstruction Framework [33] to account for detector-level simulation. Since this process is expensive (in computing time and time in general) it was not used in this thesis.

For the generation of polarised particles, MG5 provides an easy syntax since its version 2.7 to specify the polarisation of particles [34]. The project AthGeneration within the Athena Framework is used for event generation. In this thesis AthGeneration 21.6.64 was used to handle the event generation with MG version 2.9.2 and Pythia version 8, since it makes it fairly easy to distribute the computations on different CPUs/Nodes in a Batch system like TU Dresden's own Taurus.

The used syntax for event generation is

```
1 import model sm-no_b_mass
2 generate p p > w+{X} w+{X} j j QCD=0, w+ > l+ vl, w+ > l+ vl @0
3 add process p p > w-{X} w-{X} j j QCD=0, w- > l- vl~, w- > l- vl~ @0
```

where the X in curly brackets can be set to $X = 0$ for longitudinal polarisation and $X = T$ for transversal polarisation. We will refer to samples with longitudinally polarised bosons as LL, samples with transversal polarisation as TT and samples with one boson transversally polarised and one longitudinally polarised as TL or LT (depending on which is the first boson). We will only refer to TL and LT as TL.
In total we generated 230.000 events for LL, 140.000 events for TT and 80.000 events for TL. The different sample sizes result from problems in the event generation. The helicity was mostly defined in the parton parton rest frame, setting me_frame=1,2 but for the closure test in section 4.2.3 samples in the rest frame of the two W bosons were generated as well.

4.2 Data Analysis and Closure tests

The Rivet Framework allowing us to access MC data easily is introduced in section 4.2.1. Its important to check different properties that tell us whether the MC data is behaving in the expected ways. Those checks are called closure tests. For polarisation that means that the cross sections of the different processes (LL,TL,TT) should add up to the one of the process without any restrictions/specifications. We examine this in section 4.2.2. Another test is to look at the distributions of the decay angle of the W bosons' leptons in section 4.2.3. At last we analyse many different variables obtained from our MC data. Since the distribution of the Pairflavor Variable seems a bit off, we examine this further in section 4.2.4.

4.2.1 Rivet Analysis

The Rivet-Framework [35] provides a way to easily access the MC data and project it onto observables. In this thesis an already existing analysis from the ssWW group of the ATLAS experiment was used [36] and expanded with other interesting variables. Parts of the expansion were taken from the analysis made in [37].
Such an analysis initialises different histograms, loops over every event in the MC data and fills the histograms. The loop over the events made it possible to access all high and low level variables that will be presented in section 4.3. The variables for each event were written into a CSV file which is an easy to work with format for machine learning. After the analysis 37.782 LL events, 24797 TT events and 13733 TL events were available.

4.2.2 Cross section closure

The W bosons are Spin-1 particles. That means they will always have a specific polarisation. When we take a look at the different possible polarisation pairs LL, TT and TL, we have to keep in mind that they should behave like the process where no polarisation is specified, when combined. A good test is to check if the cross sections of LL, TT and TL add up to the cross section of the $W^{\pm}W^{\pm}jj$ process. In 4.1 the cross sections and the respective uncertainties are shown. There is a difference of about 2 fb between the sum and the full sample. A slight deviation is expected, since the splitting of the polarised processes makes them gauge invariant because interferences between LL,TT and TL are not accounted for. We conclude that this is still a good closure.

	σ [fb]	$\Delta\sigma$ [fb]
LL	1.889	0.006
TL	17.738	0.206
TT	17.387	0.050
Σpol	37.014	0.262
ssWWjj	35.128	0.099
$\Sigma\sigma(pol) - \sigma(WW^{\pm}jj)$	1.886	0.163

Figure 4.1: Cross sections of the different polarised $W^{\pm}W^{\pm}jj$ processes, the polarised $W^{\pm}W^{\pm}jj$ process and their differences.

4.2.3 Decay Angle Distributions

To properly quantify the polarisation of the bosons by measured quantities of their decay products, the *decay angle* θ^*_W is introduced. It describes the angle between the lepton's momentum in the boson's rest frame and the direction of the boson's momentum boosted into the boson's rest frame or the rest frame of the two protons [38].

If the bosons momentum is considered in the same frame of reference the helicity was defined in, the angular distribution of the decay products should follow an analytically expected distribution [39]

$$\frac{1}{\sigma}\frac{d\sigma}{d\cos\theta^*} = \frac{3}{8}(1-\cos\theta^*)^2 f_L + \frac{3}{8}(1-\cos\theta^*)^2 f_R + \frac{3}{4}\sin^2\theta^* f_0 \tag{4.1}$$

Here f_L and f_R are the polarisation fractions of the transversely polarised bosons and f_0 is the fraction of the longitudinally polarised bosons. This formula can only be used if no lepton cuts are applied.

Equation 4.1 gives us a measure on how good the decay angle distributions of the MC samples match the expected distribution, i.e. how trustworthy the polarised samples are. Therefore the decay angle θ^*_W was included in the rivet analysis. The helicity of the majority of samples was defined in the protons' rest frame. Since it was not possible in the time span of this thesis to obtain the protons' original momentum, a set of test samples with helicity defined in the W bosons rest frame was generated. Figure 4.2 shows the angular distributions as well as a function that fits the expected distribution onto the data. For this plot's generation the analysis provide by [37] is used.

In another bachelor thesis that was using samples generated with the new MadGraph polarisation syntax [37], a good closure couldn't be found.

In Figure 4.2, the distributions of the TT and TL sample seem to fit quite well onto the prediction. The distribution of the LL sample however, does not bring a good closure.

Nevertheless, there was no hint on what problem might cause this, therefore this needs to be investigated further. Since the distributions for TT and TL seem ok, we accept this as an okay closure.

4.2.4 Pairflavor Variable

An interesting variable is the *Pairflavor*. It states into which combination of leptons the W bosons decayed. The possible categories in our case are: ee, μe, $e\mu$, $\mu\mu$. There are only these four categories since we are excluding tau events in the analysis, which lowers the complexity of the analysis. The histogram in Figure 4.3 shows these categories as bins on the x-axis and the fraction of events on the y-axis.

This variable should not be sensitive to the polarisation of the W bosons since the decay probabilities of the bosons into the leptons are not dependent on that either. We would expect this distribution to be constant over all bins for all processes. The observed TL process does not behave like that. It seems like mixed-flavor decays (μe, $e\mu$) are preferred over same-flavor decays.

One assumption on why that is was that although the taus are neglected for simplicity in the

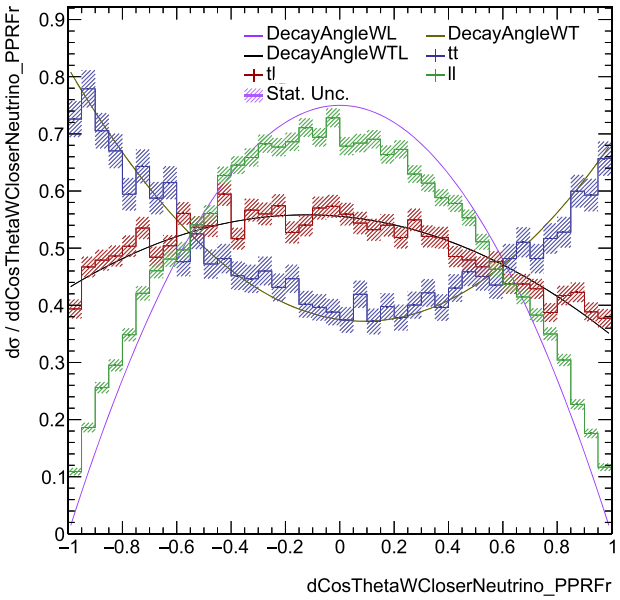


Figure 4.2: Plot showing the normalised cross section fractions of the angular Distributions of the W bosons' decay products and the according Fitfunction after Equation 4.1.

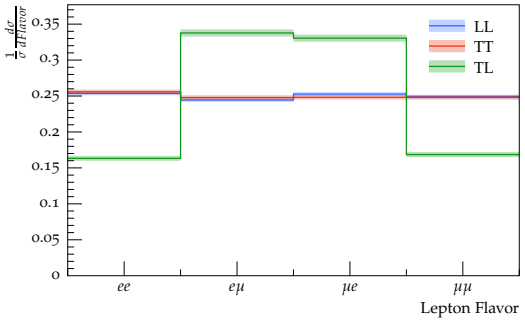


Figure 4.3: Plot showing cross section fractions the Pairflavor Variable, representing the lepton flavors into which the W bosons decay in this works studied VBS signature.

analysis they might have an influence on the decay modes, since the event could be a tau that decayed into a muon and therefore we have more counts on the mixed-channels. It is still unclear why this should only happen for the TL process.
Additionally the TL process is the one where most of the MC samples failed, i.e. they did not reach by far the targeted number of events[1]. So another assumption is that this generator-like problem might affect the pairflavor.
Since the other variables do not show this effect that one polarised process behaves significantly different than the others, we accepted the MC samples under the caveat that this problem should be investigated further.

4.3 Selection of polarisation sensitive variables

The selection of variables sensitive to polarisation that give the NN a chance to discriminate between the processes was done with the histogram output of the rivet analysis. Many different variables were looked at, where the choice was inspired by [12].
There are variables depending on a third jet, like the Zeppenfield variable of the third jet, that are quite sensitive to polarisation. Since there are a lot of events without a third jet in our dataset, it was tested in the NN training if a fusion is possible in a way, that for the only 2 jet events the third jet variables get set to zero. Unfortunately this let to no good generalisation on new data with only 2 or 3 jet events. Therefore only variables that depend on one or two jets are used.
The used variables are the p_T, ϕ and η coordinate of the two leptons and the two jets. We will refer to the rapidity as η so that there is no confusion with the cartesian y-coordinate. The transversal momentum p_T of a particle is the absolute value of the components of the momentum vector that are transversal to the beam direction, i.e. the x- and y-component.
As high level variables were used the x- and y-component of the missing transverse energy (MET), the difference in η and ϕ of the jets, the mass of both jets and both leptons. The MET, better to speak of missing transverse momentum, is defined as

$$E_T^{miss} = -\sum_i \overrightarrow{p}_{T,i} \tag{4.2}$$

where the sum goes over all particles participating and resulting of a collision. Since the initial transverse momentum is zero, because movement is only allowed in z-direction, MET gives an estimate on how many particles did not get detected.
Also the Zeppenfield variable, as defined in Eq. 4.3, for the two leptons,

$$z_X = \frac{|\eta_X - |\eta_{j_1} + \eta_{j_2}|/2|}{\Delta\eta_{jj}} \tag{4.3}$$

[1]The MadGraph authors were contacted because of this issue [40]

the distance $\Delta R_{ll,jj}$ in the ϕ and η space of the two leptons and jets, and the R_{p_T} variable

$$R_{p_T} = \frac{p_T^{l_1} \cdot p_T^{l_2}}{p_T^{j_1} \cdot p_T^{j_2}} \tag{4.4}$$

were used.

To get an idea on how sensitive to polarisation these variables are, the distributions of the m_{jj} and m_{ll} observable are shown in Figure 4.4. These variables have a score of 3.3 sigmas on the significance metric, introduced in section 3.3.3. The distributions of the other variables that were included in the input CSV for the NNs are shown in A.3.

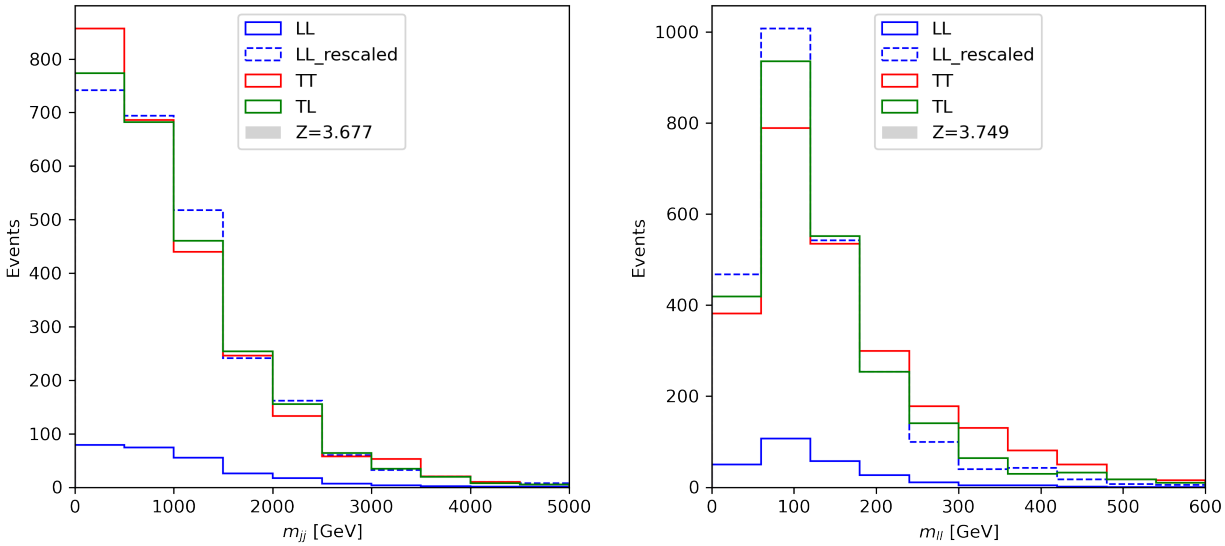


Figure 4.4: Distributions of m_{jj}(left) and m_{ll}(right) as example for polarisation sensitive variables. The dashed blue line shows only a qualitative distribution of LL rescaled to the cross sections of TT and TL, to get an idea of the actual shape difference. The solid lines represent the distributions scaled with their respective cross sections. Significance Z as figure of merits is introduced in section 3.3.3.

5 Performance of Neural Networks

In this chapter we finally look at the trained NNs and rank them according to the figures of merits.

5.1 Figures of merits

In this thesis both, the AUC and the significance are used to distinguish and rate different models of NNs.
The AUC is determined from the NN models predictions on the test dataset. The significance is determined using a holdout dataset of 1000 events for each process, completely unknown to the models. This dataset is chosen because it turned out to be laborious to split the test dataset again into the different polarisation processes. The holdout dataset gets divided into a signal dataset (only LL) and a background dataset (TT,TL). The NN model predicts on the datasets, we put it into a histogram and scale it to the expected number of events with the help of the crosssection and a toy luminosity of $140\,\mathrm{fb}^{-1}$. Then the significance is calculated as given in Equation 3.11.
In reality the background data set would additionally consist of other processes like Z+j or WZ, which are not included in this study. Furthermore, experimentally observed data inherits a lot of detector effects like charge flip events or fake lepton classification [11], which are as well not included because we did not perform detector simulations on the dataset. If we were to account for all these background contributions we would have to slightly change the definition of the significance. In this work we will neglect all background except for TT and TL polarised $W^{\pm}W^{\pm}jj$ scattering. Nevertheless we can already get an idea of the improvent to the discrimination by the m_{ll} and m_{jj} observable, as shown in Figure 4.4.

5.2 Implementation of neural networks in Python

The implementation in python is be done using the high level deep learning API Keras[41]. Keras relies on a computation back-end, where in this thesis the TensorFlow[42] platform was chosen, as a powerful machine learning library. The Scikit-Learn Library[43] is used for its simple implementation on the splitting of the data into a train, validation and test dataset.

A detailed example of a typical network training is shown in appendix A.4 as well as in the Keras documentation [41].

5.3 Promising models

For this study many different NN topologies were trained and tested. The main difference between the topologies is the number of input vectors, i.e. input branches. Models with one input vector, also called Sequential Models, and with two, three, four and five input vectors were build. The branches of the models with more than one input vector were merged together in different ways, inspired by [12]. The ReLU function is used for activation in the hidden layers and the sigmoid function is used for the output neuron. For each of these "main topologies" different combinations of input parameters were tested as well as different numbers of hidden layers, hidden units, dropout rate and learning rate. This search was done again with standardised input variables. For that scikit-learns Standard Scaler was used, which scales each variable distribution to have a mean of 0 and a standard deviation of 1 (assuming we have Gaussian data, which is approximately true for large sample sizes).
After the search three models for each topology were chosen to be evaluated in a final round on the test and holdout dataset, according to their learning loss and accuracy. That were in total 42 models on which the following observations were made.

Ranking of input variables. For each topology a different set of input vectors was made, since a network trained on one input vector cannot be evaluated with a different input size. The following interpretation is only a qualitative one whether input vector combinations made it to the final round or not.
For the sequential models these were basically tests on which variables are really needed for the network to understand the pattern, i.e. tests were variables of section 4.3 were left out. Amongst the best performing NNs were the ones that had left out either the Zeppenfield variables, the ϕ coordinate of the leptons and jets respectively or the η coordinate of the jets. This makes sense because for the η and ϕ variables also the difference between them was given, so the information would have been included twice. Although the Zeppenfield variables seem to give a good discrimination between the polarised processes, they don't seem to help the NNs do their jobs better.
The best choices for models with two input branches are three:

(a) as first input p_T, ϕ and η of all leptons and jets and as second input all high level variables,

(b) as first input all variables that have to do with the leptons and the MET and as second input all variables that have to do with the jets and the MET,

(c) all variables as first input and as second input the p_T of the two leptons.

Options a) and b) seem to be a nice seperation between the amount of input variables, so that each branch can learn to discriminate on specific features of the particles. Option c) is merely a fancier sequential model where the second branch really examines the features of the p_T.
Models with three input vectors performed best when these were split into

(a) as first input all variables in connection to the p_T, as second input all variables in connection to ϕ and η coordinates and as third input the MET components and R_{p_T}, or

(b) as first input p_T, η and ϕ of the leptons and jets, as second input the ΔR and Zeppenfield variables, as third input the MET components, R_{p_T} and m_{ll}, m_{jj}..

For models with four input branches only one input set made it to the best models. Here the variables in connection with p_T were on the first branch, variables in connection to η on the second branch, variables in connection to ϕ on the third branch and at last the Zeppenfield variables, the ΔR variables and MET on the fourth branch.
There is also only one top candidate for models with five input vectors, where the first contains the $p_T s$, R_{p_T} and m_{ll}, m_{jj}, the second contains all η variables, the third all ϕ variables, the fourth the Zeppenfield variables and the fifth ΔR variables and MET.
For a full overview of the specific input vectors see Table A.1.

Influence of scaling. Models with standardised input variables got on average higher ratings on all scores (loss/accuracy, AUC, significance). This effect was most prominent for the sequential models. The sequential models with unscaled inputs ranked on the AUC score at around 0.77 whereas models with scaled inputs achieved about 0.8, which is an improvement of about 5%.
It seems reasonable that the scaling has such an impact on the NNs. Think of the ϕ coordinate of one lepton and the p_T of one lepton. The ϕ variable has only a range of 0 to 2π whereas the p_T variable can have values in a range of "0 GeV" to approximately 700 GeV. This is a huge difference in scale. That way it is more likely for the p_T variable to become dominant because of its high values. To eliminate it's best to scale all variables to a similar range.
Nevertheless there were models were scaling didn't make such a big difference as in the case of the sequential models. This happened mostly for NNs with four or five input branches. Here variables of similar range were grouped together, so scaling does not change the range of the values as much.

Best Topologies. To find the best topologies for the classification task of discriminating between the LL process and the background processes, we take a look at the models scoring best on our figures of merits.

At first we look at the AUC. Because of how difficult the task is to differentiate between the polarised processes, we did not expect a huge variance in the score values.
The best sequential model achieved an AUC of 0.804. It has three hidden layers, a dropout rate of 20% after each layer and 150 nodes per layer.
The highest scoring models on the AUC score are of topologies with four and five branches. The best model achieved an AUC of 0.809. It has four input branches, where each branch consists of four hidden layers with no dropout and 100 nodes each. These four branches are then combined to a single branch with again two hidden layers of 100 nodes each. The second best score was achieved by a model with five input branches, with the same topology as the four branch model added with an extra branch. Figure 5.1 shows the topology of the NN with four branches.

The significance score yields a similar result. Here the models of topologies with four and five input vectors performed the best as well. The above described topology of the four branch network, now with 50 instead of 100 nodes, performs best with a significance of 5.656 sigma. The best sequential model, the same as for the AUC score, reaches a significance of 5.499 sigma. If we compare these results to the discriminative power of typically used variables like m_{ll} and m_{jj} as shown in Figure 4.4, we see that the NN discriminant provides an improvement of about 2 sigma.

The above described model with four input branches performs best on all scores. To visualise this performance Figure 5.2 shows the discriminant output of the best model in comparison to the best sequential model. Figure 5.3 shows in conclusion the ROC curves with the respective AUC in the label for the best models. All top models perform quite similar, nevertheless further tests on input vectors and more MC data would bring improvement. For future studies it seems reasonable to either work with the more difficult to implement four branch model or with the sequential models, both with about three layers per branch and between 50 and 150 nodes. Dropout does not seem to be necessary.

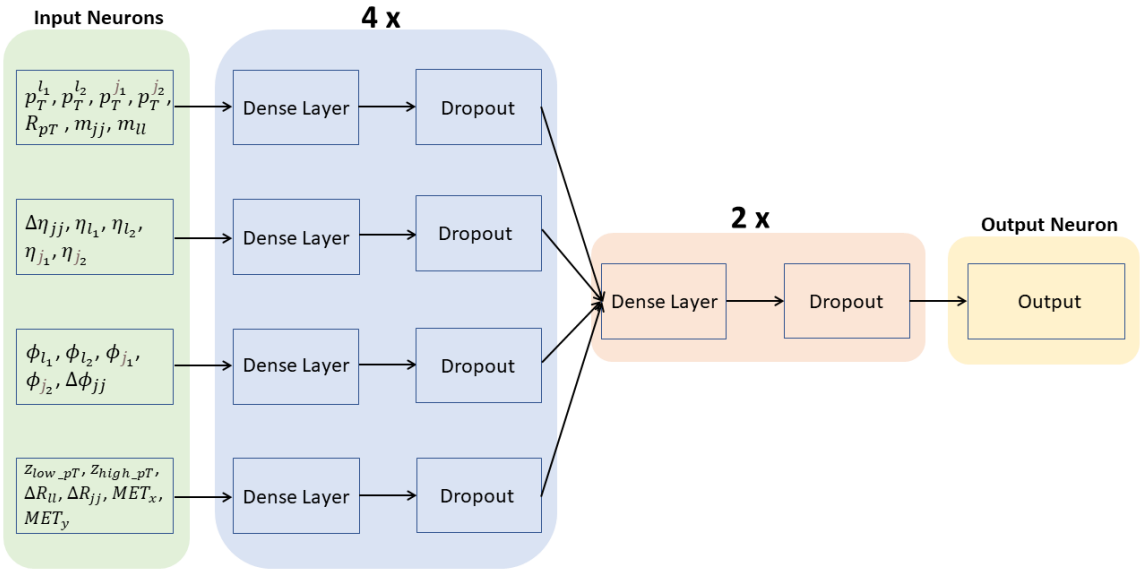


Figure 5.1: Sketch of the best topology with four input branches. In the green boxes all input variables are shown. Layers in the blue and orange boxes exist as many times as shown in the graphic.

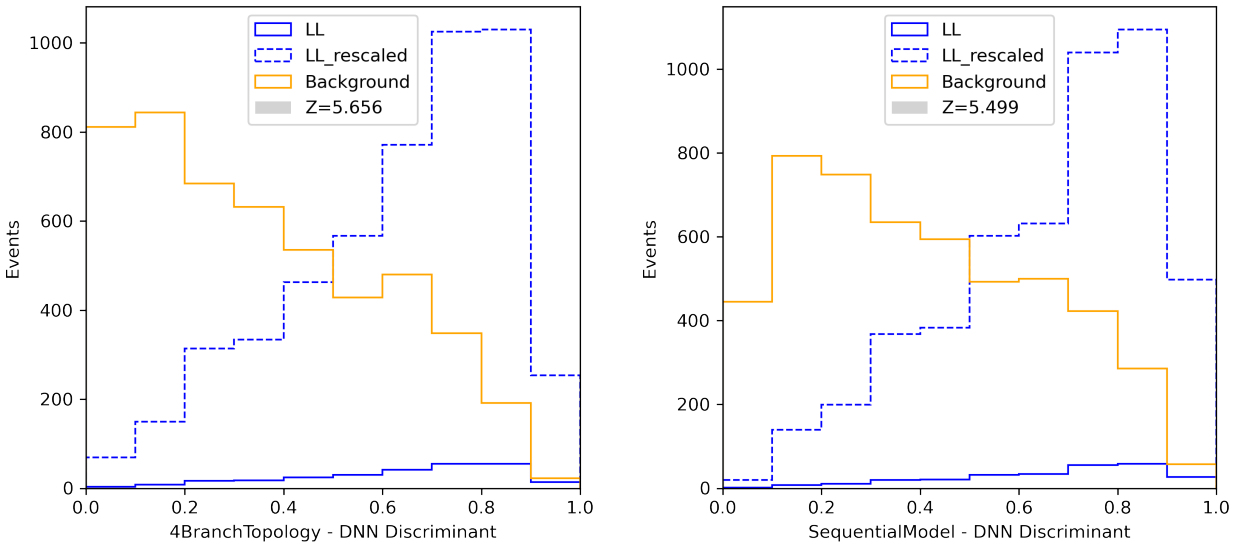


Figure 5.2: NN discriminant of the model performing best on the significance score, see section 3.3.3, on the left side and the best sequential model on the significance score on the right side. Here background means the TT+TL process. The dashed line is a qualitative distribution of LL, scaled to the cross section of the background, for a better shape comparison. The solid lines represent the distributions scaled with their respective cross sections.

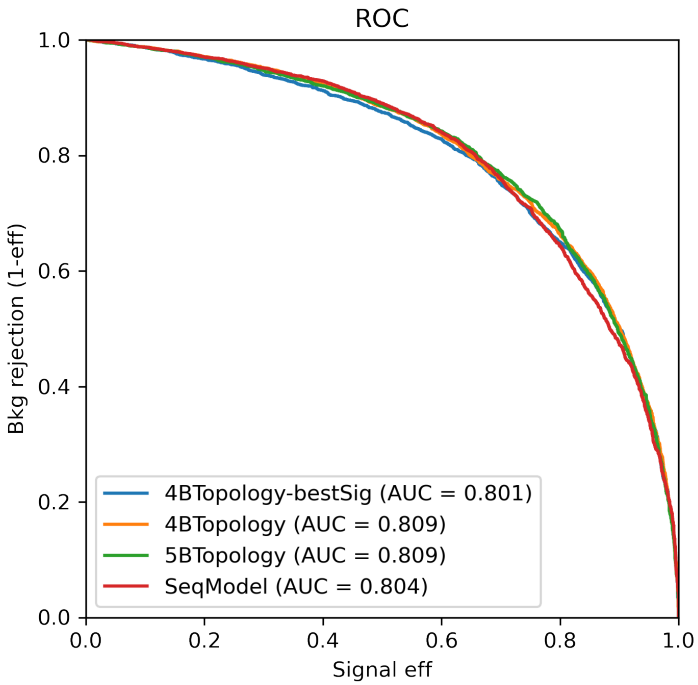


Figure 5.3: ROC curves of the models scoring best on the AUC score and the model scoring best on the significance score.

6 Summary and Outlook

The observation of longitudinally polarised VBS will be an important step in the progress of measuring the predicted quantities of the SM. In this thesis, neural networks (NNs) were studied as a tool to discriminate between $W^{\pm}W^{\pm}jj$ events with two longitudinally-polarised W bosons (LL) as signal process and events with at least one transversely-polarised W boson (TT,TL) considered as background. Our study was only preliminary, since only TT and TL were taken into account as background processes. Assuming we would be able to isolate only $W^{\pm}W^{\pm}jj$ in an analysis, a NN with four input vectors was able to provide a good discrimination, with a significance of 5.66 sigmas. In reality it is expected with $3000\,\mathrm{fb}^{-1}$ for CMS and ATLAS to reach 3 sigma in the study of $W^{\pm}W^{\pm}jj$ [44]. This is 20 times more data than already obtained and still not enough to claim an observation by a single experiment. That is why NNs are explored as tools to make an observation more probable.

The best performing NN consisted of four input branches, each with 4 hidden layers of 50 nodes, combined into a single branch of 2 hidden layers of 100 nodes. The NN with only one input vector performed good (Z=5.5) as well. Since these are easier to implement, they could be useful for big and fast studies in future. The discrimination between LL and TL is especially challenging, since TL has longitudinal fingerprints. TL was also the process with the least MC data available. That is why with more time and MC data the result definitely could have been improved.

If more background processes like $WZjj$ or W/Z+jets are taken into account the classification task will become more complex and therefore the presented topologies will perform worse. Still, NNs seem to be a tool that will play an important role in the analysis of the longitudinally-polarised VBS.

The small inconsistencies of the MC data revealed by the closure tests in chapter 4 need to be addressed in further research, to make sure the generators provide reliable data. Here the MC data was assumed to be sufficiently well modeled, despite those small deviations. With the use of different MC generators or even experimental data, fits on the NN discriminant will be done in the future to measure the actual fraction of LL events. This was not achievable in this work because of the limited time span of the thesis.

In this thesis first steps on the long road of an analysis were taken, showing that the discrimination between the polarised W boson processes is achievable with NNs. A next step on this road could be the extension of this study with detector simulated MC data and more

background processes, such as WZjj, included. In the end, a major analysis of data acquired in the High-Luminosity LHC will hopefully bring an observation of polarised VBS with two W bosons.

A Appendix

A.1 Activation Functions

This short section is supposed to give a small overview on the different activation functions, introduced in chapter 3. Here w represents any weight, x any input vector and b any bias, so that the weighted input of some neuron is $z = w \cdot x + b$.
The following functions will be covered: sigmoid, softmax, ReLU, tanh, which are all shown in Figure A.1
The sigmoid function is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{A.1}$$

It is basically a smoothed out version of a step function [22], so that it is differentiable at 0. A caveat of neurons that have a sigmoid activation is that the training process is slowed down vastly when the output is near 0 or 1 since the slope in these areas is quite small. If used as activation function for a NN with one output neuron its output represents a probability distribution over a binary variable [45].
The hyperbolic tangent function tanh is defined as

$$\tanh = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{A.2}$$

It is a rescaled version of the sigmoid function, as Figure A.1 shows.
The rectified linear unit (ReLU) describes neurons using the rectifying function $\max(0, z)$.
The softmax function also known as normalised exponential function is defined as

$$softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{A.3}$$

where z_i denotes a specific neuron and the sum goes over all neurons in the respective layer. If used as activation function for output neurons its output represents a probability distribution over a discrete variable with n possible values, i.e. n different classes. It can be seen as a generalisation of the sigmoid function [45].

Unfortunately there is no real recipe on what activation function should be used in which case, since there is still some behaviour that needs to be understood [22]. Nevertheless, modern NNs

seem to work well with the ReLU used in hidden units and the softmax or sigmoid function used as output activation function, depending on whether the task is a multi-class, multi-label or binary classification.

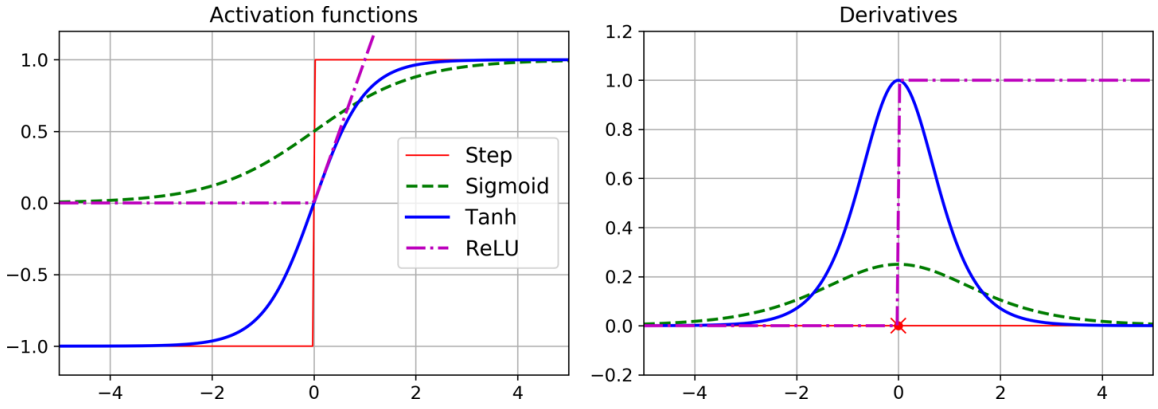


Figure A.1: Common activation functions and their derivatives taken from [21].

A.2 Backpropagation

The main steps in this section are taken from [22].

In section 3.2 we introduced Neural Networks and how they are trained. Main steps are: Determining the output or activation a of each neuron in each layer with

$$a_j^l = \Psi(\sum_k w_{jk}^l \cdot a_k^{l-1} + b_j^l) = \Psi(z_j^l) \tag{A.4}$$

where a_j^l denotes the activation of the j^{th}-neuron in the l^{th}-layer, b_j^l similarly the bias of the j^{th}-neuron in the l^{th}-layer and w_{jk}^l the weight for the connection from the k^{th}-neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th}layer. We can rewrite A.4 in a matrix form. Therefore we define a weight matrix w^l for each layer l, the entry in the j^{th} row and k^{th} column is w_{jk}^l. The activation function needs to be vectorised. That is when applying Ψ to a vector it gets applied to every element in the vector, $\Psi(\mathbf{v})_j = \Psi(v_j)$. Thus the matrix form of Eq. A.4 is

$$a^l = \Psi(w^l a^{l-1} + b^l) = \Psi(z^l) \tag{A.5}$$

After computing the loss function L to have a measure for deviation of the desired output, e.g. $L(w,b) = \frac{1}{2N}\sum_x ||y(x) - a||^2$ with $y(x)$ as desired output given x as input vector, a stochastic gradient descent step is performed. In such a step the parameters of the NN, i.e. the weights w_l and biases b_l, get updated to minimise the Loss via

$$w_k \to w_k' = w_k - \frac{\eta}{m}\sum_j \frac{\partial L_{x_j}}{\partial w_k} \tag{A.6}$$

$$b_l \to b_l' = b_l - \frac{\eta}{m}\sum_j \frac{\partial L_{x_j}}{\partial b_l} \tag{A.7}$$

Here η denotes the learning rate and the sum goes over all inputs of a mini-batch of size m. For each input the partial derivatives (gradient!) need to be computed. This is where the backpropagation algorithm starts its work.

For the algorithm to work, the loss function L needs to meet two assumptions: it can be written as an average $L = \frac{1}{N}\sum_x L_x$ over loss functions L_x for individual training examples and it can be written as a function of the outputs from the neural network $L_x = L_x(a^{lastlayer})$.

We want to be able to compute the elementwise product of two vectors, $\mathbf{s}$ and $\mathbf{t}$. The operation for that is called Hadamard product, $\mathbf{s} \odot \mathbf{t}$ with $(\mathbf{s} \odot \mathbf{t})_j = s_j t_j$.

To compute the partial derivatives $\partial L_x/\partial w_{jk}^l$ and $\partial L_x/\partial b_j^l$ we introduce an intermediate quan-

tity, the error in the j^{th} neuron in the l^{th} layer, δ_j^l.

$$\delta_j^l = \frac{\partial L_x}{\partial z_j^l} \tag{A.8}$$

The expression in Eq. A.8 is somehow a measure on how strong the influence on the cost function would be if we would vary the weighted output.
Starting with the error of the neurons of the last output layer, we can easily derive from Eq. A.8 the Eqs. A.9,A.10 where the latter is the same expression as the first, only in vector/matrix notation.

$$\delta_j^L = \frac{\partial L_x}{\partial a_j^L} \frac{\partial \Psi(z_j^l)}{\partial z_j^l} \tag{A.9}$$

$$\delta^L = \nabla_a L_x \odot \frac{\partial \Psi(z^l)}{\partial z^l} \tag{A.10}$$

Since we know the error of some neuron in between (Eq. A.8) depending on the weighted output it is better to know how the intermediate error can be calculated in relation to the weights and the errors of higher layers, which is given directly in matrix notation in Eq. A.11. It describes the error in layer l depending on how the weights between layer l and $l+1$ and the activation of the neurons in layer l are influencing the error in layer $l+1$.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \Psi'(z^l) \tag{A.11}$$

With Eqs. A.10 and A.11 we can now compute δ^l for any layer in the NN. What's left is to find equations that give us a relation between the Loss function and the errors, regarding the weights and biases, since these are the parameters we want to update.
These are given in Eqs. A.12 and A.13. Showing the gradients left to compute during our gradient descent step in Eqs. A.6, A.7.

$$\frac{\partial L_x}{\partial b_j^l} = \delta_j^l \tag{A.12}$$

$$\frac{\partial L_x}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{A.13}$$

A complete derivation of the Backpropagation formulas goes beyond the scope of this thesis. Although, the proofs of the four equations are only applying the chain rule over and over again, it makes sense to read a far more detailed motivation on this topic as for example in [22].

A training loop would look like:

- get the set of input variables from the mini-batch
- For each training example x: Set the activations $a^{x,1}$ of the first layer and
  - Feedforward: compute for each layer $l = 2, .., L$ the weighted input $z^{x,l} = w^l a^{x,l-1} + b^l$ and the activation $a^{x,l} = \Psi(z^{x,l})$
  - Compute the output Error like in Equation A.10
  - Backpropagate the error, i.e. compute the error for each layer $l = L-1, L-2, ..., 2$ like in Equation A.11
- Make the Gradient Descent Step: Update the parameters of the NN via Eqs. A.6, A.7 with the help of Eqs. A.12, A.13 for each layer $l = L-1, L-2, ..., 2$

A.3 Polarisation sensitive variables - plots

Plots of chosen variables as input to NNs. Histograms are normalised to make shape comparison easier.

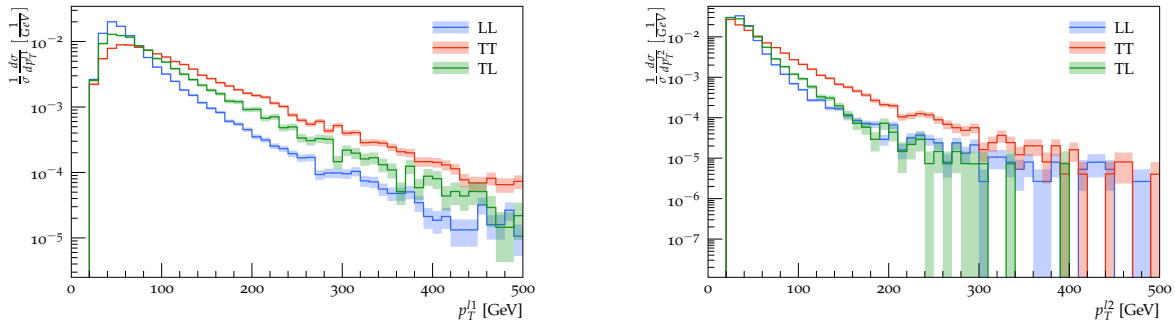


Figure A.2: Plot showing cross section fractions of the transverse momentum distribution of the leptons originating from the W bosons.

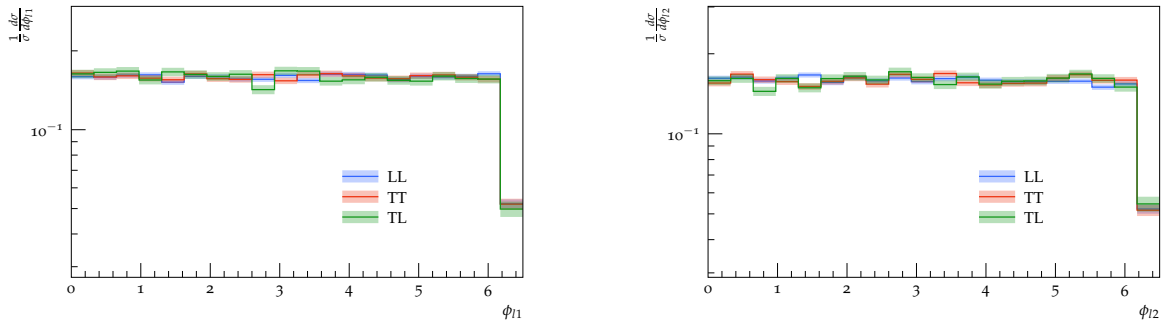


Figure A.3: Plot showing cross section fractions of the phi coordinate distribution of the leptons originating from the W bosons.

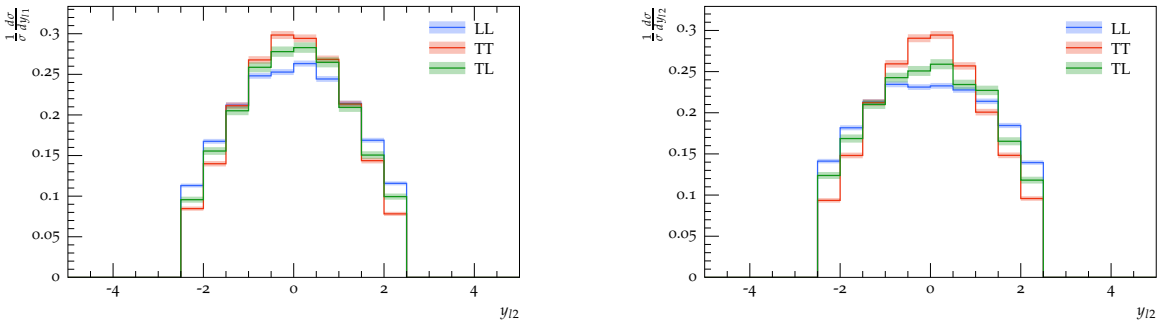


Figure A.4: Plot showing cross section fractions of the rapidity distribution of the leptons originating from the W bosons.

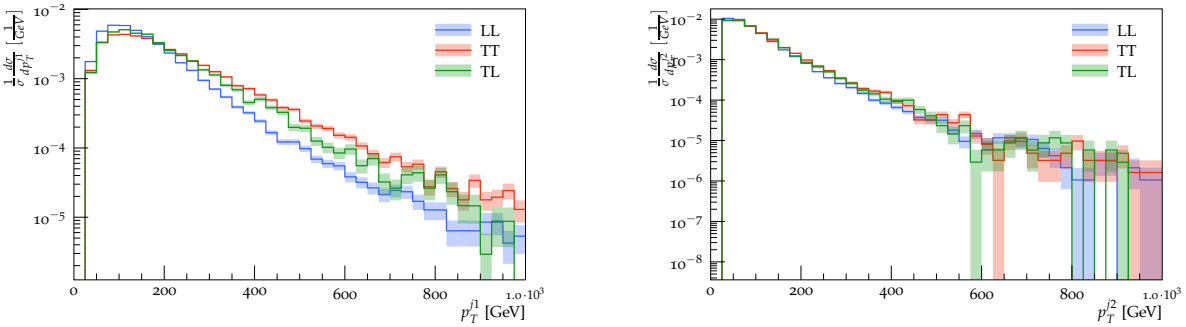


Figure A.5: Plot showing cross section fractions of the transverse momentum distribution of the jets.

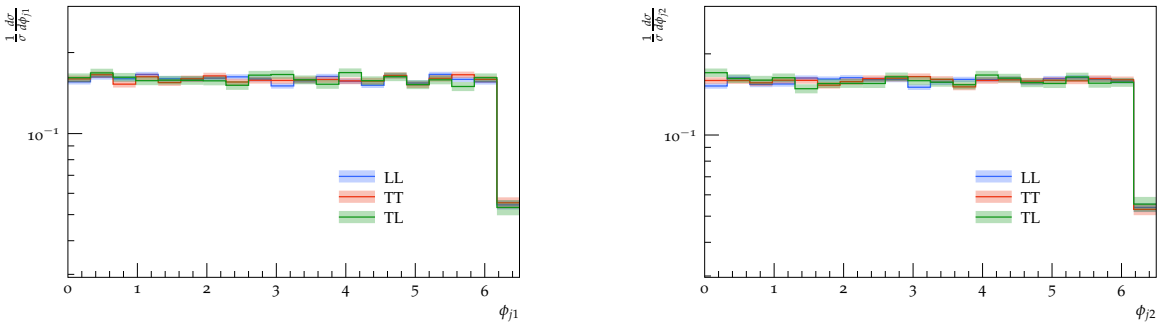


Figure A.6: Plot showing cross section fractions of the phi coordinate distribution of the jets.

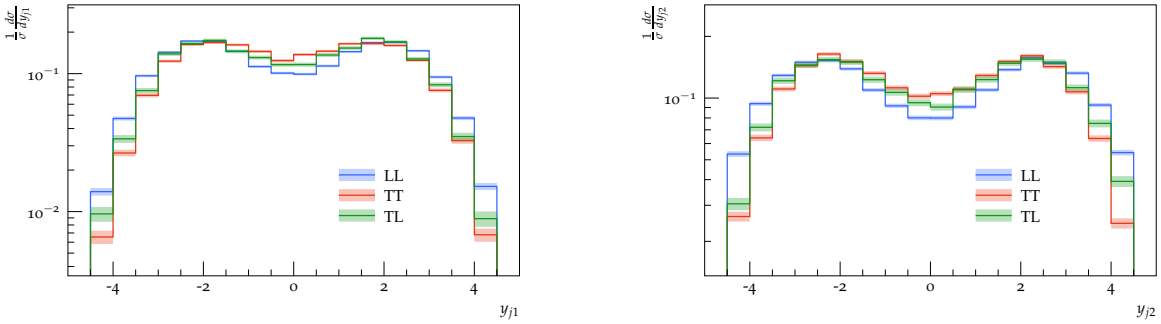


Figure A.7: Plot showing cross section fractions of the rapidity distribution of the jets.

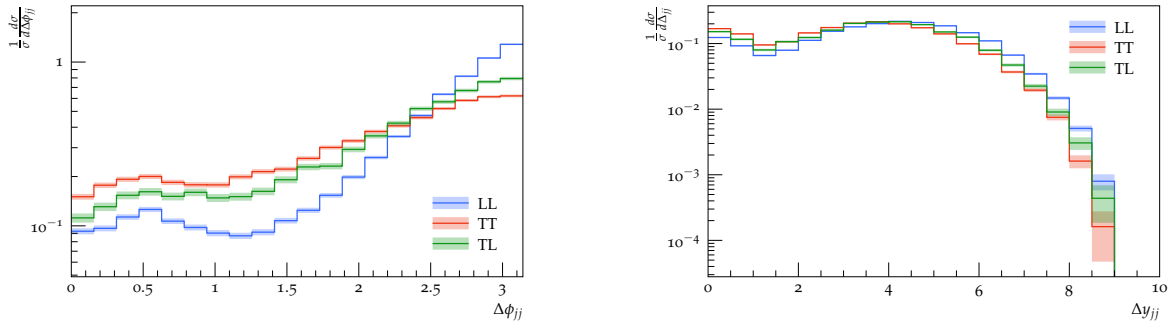


Figure A.8: Plot showing cross section fractions of the $\Delta\phi_{jj}$ (left) and Δy_{jj} (right) distribution of the jets.

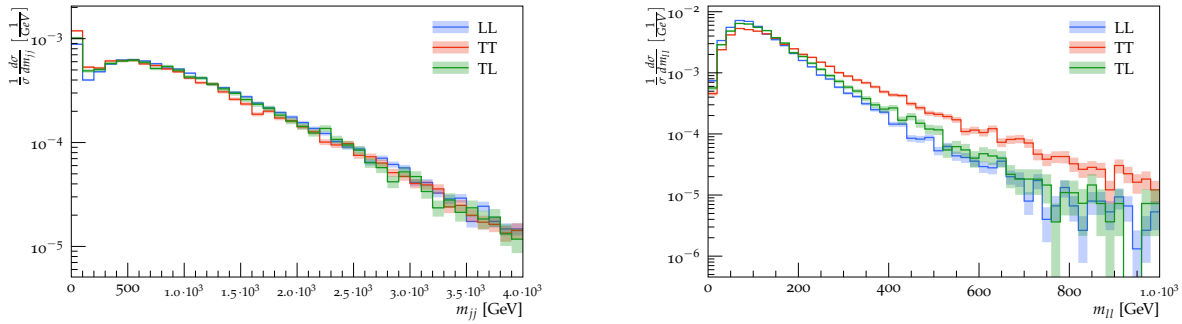


Figure A.9: Plot showing cross section fractions of the m_{jj} (left) and m_{ll} (right) distribution.

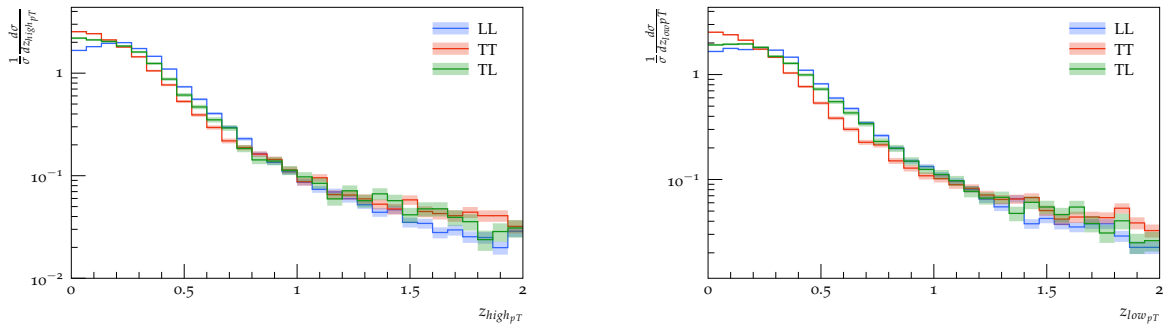


Figure A.10: Plot showing cross section fractions of the $z_{high_{pT}}$ (left) and $z_{low_{pT}}$(right) distribution of the jets.

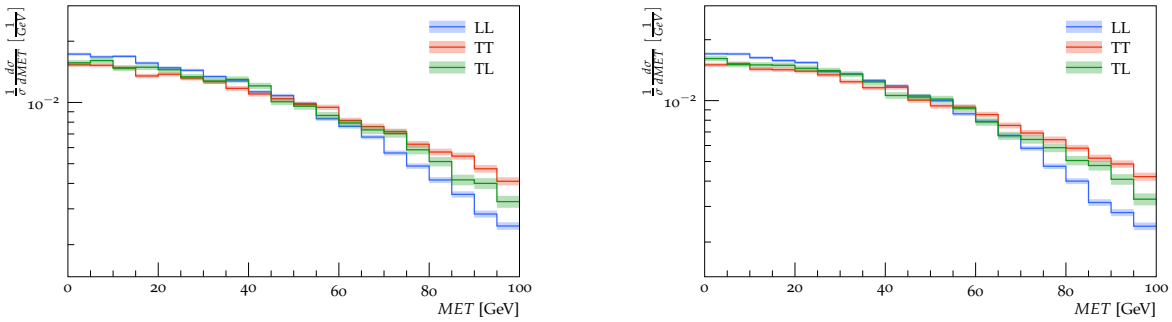


Figure A.11: Plot showing cross section fractions of the distribution of the missing transverse energy components, MET_x on the left and MET_y on the right.

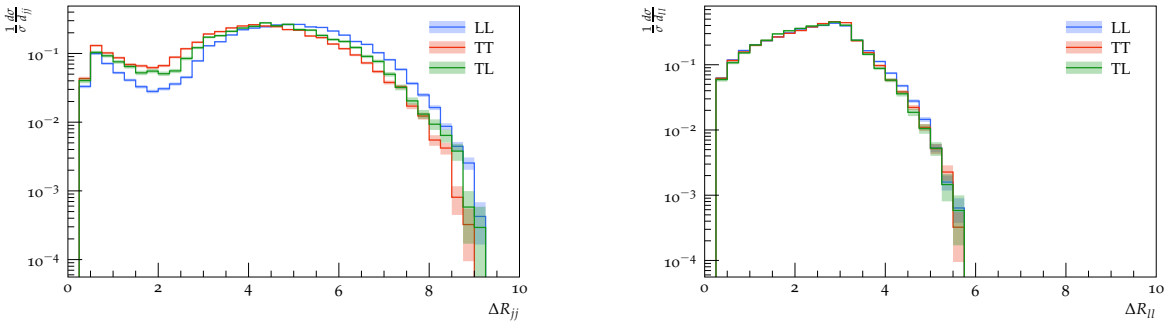


Figure A.12: Plot showing cross section fractions of the distribution of the distance in the rapidity and ϕ space, for the jets on the left and for the leptons on the right.

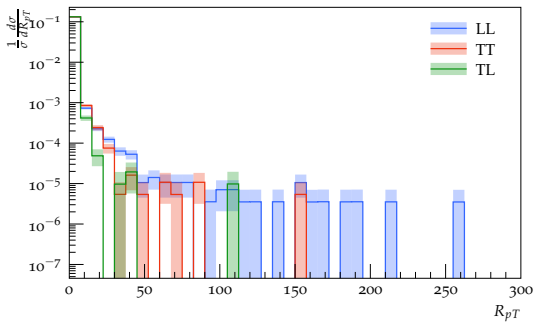


Figure A.13: Plot showing cross section fractions of the distribution of the R_{pT} variable as introduced in Section 4.3.

Topology	Input variables
Sequential Model	['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'pT_j1', 'pT_j2', 'eta_j1', 'eta_j2', 'phi_j1', 'phi_j2', 'mll', 'mjj', 'd_eta_jj', 'd_phi_jj', 'DeltaR_ll', 'DeltaR_jj', 'R_pT', 'MET_x', 'MET_y', 'isLL']
	['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'pT_j1', 'pT_j2', 'eta_j1', 'eta_j2', 'phi_j1', 'phi_j2', 'mll', 'mjj', 'z_low', 'z_high', 'd_eta_jj', 'd_phi_jj', 'DeltaR_ll', 'DeltaR_jj', 'R_pT', 'MET_x', 'MET_y', 'isLL']
	['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'pT_j1', 'pT_j2', 'eta_j1', 'eta_j2', 'mll', 'mjj', 'z_low', 'z_high', 'd_eta_jj', 'd_phi_jj', 'DeltaR_ll', 'DeltaR_jj', 'R_pT', 'MET_x', 'MET_y', 'isLL']
	['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'pT_j1', 'pT_j2', 'phi_j1', 'phi_j2', 'mll', 'mjj', 'z_low', 'z_high', 'd_eta_jj', 'd_phi_jj', 'DeltaR_ll', 'DeltaR_jj', 'R_pT', 'MET_x', 'MET_y', 'isLL']
2 branch model	a1 = ['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'pT_j1', 'pT_j2', 'eta_j1', 'eta_j2', 'phi_j1', 'phi_j2']
	a2 = ['z_low', 'z_high','d_eta_jj', 'd_phi_jj', 'DeltaR_ll', 'DeltaR_jj', 'R_pT', 'MET_x','MET_y', 'mll', 'mjj']
	b1 = ['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'mll', 'MET_x','MET_y', 'z_low', 'z_high', 'DeltaR_ll', 'R_pT']
	b2 = ['d_eta_jj', 'd_phi_jj', 'DeltaR_jj', 'R_pT', 'MET_x','MET_y', 'mjj', 'pT_j1', 'pT_j2', 'eta_j1', 'eta_j2', 'phi_j1', 'phi_j2']
	c1 = ['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'mll', 'z_low', 'z_high', 'DeltaR_ll', 'R_pT', 'MET_x','MET_y', 'd_eta_jj', 'd_phi_jj', 'mjj', 'pT_j1', 'DeltaR_jj', 'pT_j2', 'eta_j1', 'eta_j2', 'phi_j1', 'phi_j2']
	c2 = ['pT_l1', 'pT_l2']
3 branch model	a1 = ['mll', 'mjj', 'pT_l1', 'pT_l2', 'R_pT', 'pT_j1', 'pT_j2']
	a2 = ['eta_j1', 'eta_j2', 'eta_l1', 'eta_l1', 'd_eta_jj']
	a3 = ['phi_l1', 'phi_l2', 'phi_j1', 'phi_j2', 'd_phi_jj', 'DeltaR_jj', 'DeltaR_ll']
	b1 = ['MET_x', 'MET_y', 'R_pT', 'mll', 'mjj']
	b2 = ['pT_l1', 'pT_l2', 'eta_l1', 'eta_l2', 'phi_l1', 'phi_l2', 'pT_j1', 'pT_j2', 'eta_j1', 'eta_j2', 'phi_j1', 'phi_j2']
	b3 = ['DeltaR_jj', 'DeltaR_ll', 'z_low', 'z_high']
4 branch model	a1 = ['pT_l1', 'pT_l2', 'pT_j1', 'pT_j2', 'R_pT', 'mll', 'mjj',]
	a2 = ['d_eta_jj', 'eta_l1', 'eta_l2','eta_j1', 'eta_j2',]
	a3 = ['phi_j1', 'phi_j2', 'phi_l1', 'phi_l2', 'd_phi_jj',]
	a4 = ['z_low', 'z_high', 'DeltaR_ll', 'DeltaR_jj', 'MET_x','MET_y']
5 branch model	a1 = ['pT_l1', 'pT_l2', 'pT_j1', 'pT_j2', 'R_pT', 'mll', 'mjj']
	a2 = ['d_eta_jj', 'eta_l1', 'eta_l2','eta_j1', 'eta_j2']
	a3 = ['phi_j1', 'phi_j2', 'phi_l1', 'phi_l2', 'd_phi_jj']
	a4 = ['z_low', 'z_high']
	a5 = ['DeltaR_ll', 'DeltaR_jj', 'MET_x','MET_y']

Table A.1: Input variable configurations for the different topology types of neural networks.

A.4 Implementation of Neural Networks with Keras

The implementation in python can be done using the high level deep learning API Keras[41]. Keras relies on a computation backend, in this thesis the TensorFlow[42] platform was chosen, a powerful machine learning library. The Scikit-Learn Library[43] is used for its simple implementation on the splitting of the data into a train, validation and test dataset.
Keras is an easy tool that helps to create different models of neural networks in a simple way. To create for example the NN shown in Figure 3.2 one can use the following syntax:

```
model = keras.Sequential()
model.add(keras.layers.Dense(4, input_shape=(6,), activation="relu"))
model.add(keras.layers.Dense(3, activation="relu"))
model.add(keras.layers.Dense(1, activation="sigmoid"))
optimizer = keras.optimizers.Adam(lr=0.001)
model.compile(loss='binary_crossentropy', optimizer=optimizer,
    metrics=['accuracy'])
```

After creating or building such a model it needs to be compiled, so to specify the training algorithm (variation of SGD, exact type of loss function). The type of this model is called sequential, i.e. a plain stack of layers where each layer has exactly one input and one output [41].

There is also another possibility of defining NNs with greater flexibility in keras by using the functional API. It can handle models with non-linear topology, shared layers, and even multiple inputs or outputs[41].
It is similarly easy to create NNs that way, see for example the syntax for a simple model with 2 different inputs and one output.

```
# define two sets of inputs
inputA = Input(shape=(10,))
inputB = Input(shape=(7,))

# the first branch operates on the first input
x = Dense(units=50, activation="relu")(inputA)
x = Dense(units=50, activation="relu")(x)
x = Model(inputs=inputA, outputs=x)

# the second branch opreates on the second input
y = Dense(units=30, activation="relu")(inputB)
y = Dense(units=60, activation="relu")(y)
y = Model(inputs=inputB, outputs=y)

# combine the output of the two branches
```

```
combined = concatenate([x.output, y.output])

z = Dense(units=40, activation="relu")(combined)
z = Dense(1, activation="sigmoid")(z)
# our model will accept the inputs of the two branches and
# then output a single value
model = Model(inputs=[x.input, y.input], outputs=z)
# Compile model
model.compile(optimizer=keras.optimizers.Adam(lr=0.001),
    loss='binary_crossentropy', metrics=['accuracy'])
```

After building and compiling the model we can train it with

```
history = model.fit(x_train, y_train, batch_size=64, epochs=2,
    validation_data=(X_valid, y_valid))
```

In the history object is an entry for each epoch with the Losses and Accuracies so that this can be plotted over the epochs to obtain the learning curves for this particular model.

B Bibliography

[1] L. Wang et al. “Will any crap we put into graphene increase its electrocatalytic effect?” In: *ACS nano* 14.1 (2020), pp. 21–25.

[2] C. Visions et al. "Kinesin protein walking on microtubule". URL: `https://www.youtube.com/watch?v=y-uuk4Pr2i8`.

[3] A. W. Hunter et al. “The kinesin-related protein MCAK is a microtubule depolymerase that forms an ATP-hydrolyzing complex at microtubule ends”. In: *Molecular cell* 11.2 (2003), pp. 445–457.

[4] M. A. Nielsen et al. "Quantum computation and quantum information". 2002.

[5] J. J. Thomson. “XL. Cathode rays”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 44.269 (1897), pp. 293–316.

[6] A. Romer. “Proton or prouton?: Rutherford and the depths of the atom”. In: *American Journal of Physics* 65.8 (1997), pp. 707–716.

[7] S. Weinberg. “A model of leptons”. In: *Physical review letters* 19.21 (1967), p. 1264.

[8] G. Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29.

[9] T. Albahri et al. “Measurement of the anomalous precession frequency of the muon in the Fermilab Muon g- 2 Experiment”. In: *Physical Review D* 103.7 (2021), p. 072002.

[10] A. M. Sirunyan et al. “Observation of electroweak production of same-sign W boson pairs in the two jet and two same-sign lepton final state in proton-proton collisions at s= 13 TeV”. In: *Physical review letters* 120.8 (2018), p. 081801.

[11] M. Aaboud et al. “Observation of Electroweak Production of a Same-Sign W Boson Pair in Association with Two Jets in p p Collisions at s= 13 TeV with the ATLAS Detector”. In: *Physical review letters* 123.16 (2019), p. 161801.

[12] J. Lee et al. “Polarization fraction measurement in same-sign W W scattering using deep learning”. In: *Physical Review D* 99.3 (2019), p. 033004.

[13] P. Gagnon. “The Standard Model: a beautiful but flawed theory”. In: (). URL: `https://www.quantumdiaries.org/2014/03/14/the-standard-model-a-beautiful-but-flawed-theory/`.

[14] M. T. et al. “Review of Particle Physics”. In: *Phys. Rev. D* 98 (3 Aug. 2018), p. 030001. DOI: 10.1103/PhysRevD.98.030001. URL: https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[15] M. Thomson. "Modern particle physics". Cambridge University Press, 2013.

[16] D. Griffiths. "Introduction to elementary particles". John Wiley & Sons, 2020.

[17] M. Kobel et al. "Teilchenphysik - Unterrichtsmaterial Ab Klasse 10". In coorporation with Netwerk Teilchenwelt. Joachim Herz Stiftung.

[18] L. Evans et al. “LHC machine”. In: *Journal of instrumentation* 3.08 (2008), S08001.

[19] G. Aad et al. “The ATLAS experiment at the CERN large hadron collider”. In: *Jinst* 3 (2008), S08003.

[20] F. Pastore et al. “The ATLAS Trigger System: Past, Present and Future”. In: *Nuclear and particle physics proceedings* 273 (2016), pp. 1065–1071.

[21] A. Géron. "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems". O’Reilly Media, 2019.

[22] M. A. Nielsen. "Neural networks and deep learning". Vol. 25. Determination press San Francisco, CA, 2015.

[23] J. Brownlee. "Machine Learning Mastery - Blog". URL: https://machinelearningmastery.com/.

[24] D. Silver et al. “Alphago: Mastering the ancient game of go with machine learning”. In: *Research Blog* 9 (2016).

[25] S. Vieira et al. “Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications”. In: *Neuroscience & Biobehavioral Reviews* 74 (2017), pp. 58–75.

[26] D. P. Kingma et al. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[27] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[28] D. E. Rumelhart et al. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[29] G. Cowan et al. “Asymptotic formulae for likelihood-based tests of new physics”. In: *The European Physical Journal C* 71.2 (2011), pp. 1–19.

[30] C. Bittrich. “Evidence for Scattering of Electroweak Gauge Bosons in the $W^{\pm}Z$ Channel with the ATLAS Detector at the Large Hadron Collider”. PhD thesis. Technische Universitaet Dresden (DE), 2020.

[31] J. Alwall et al. “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations”. In: *Journal of High Energy Physics* 2014.7 (2014), p. 79.

[32] T. Sjöstrand et al. “An introduction to PYTHIA 8.2”. In: *Computer physics communications* 191 (2015), pp. 159–177.

[33] G. Aad et al. “The ATLAS simulation infrastructure”. In: *The European Physical Journal C* 70.3 (2010), pp. 823–874.

[34] D. B. Franzosi et al. “Automated predictions from polarized matrix elements”. In: *Journal of High Energy Physics* 2020.4 (2020), pp. 1–46.

[35] C. Bierlich et al. “Robust independent validation of experiment and theory: Rivet version 3.” In: *SciPost physics.* 8.2 (2020), p. 026.

[36] ATLAS Collaboration. “Rivet Analysis of ssWWjj. Repository will be made public when the analysis is published.” Repository will be made public when the analysis is published. URL: `https://gitlab.cern.ch/atlas-physics/sm/ew/ssww-run2/ssWW-Rivet`.

[37] M. Bühring. “Study of Vector Boson Polarisation in the Scattering of Two W Bosons with Equal Charge with the Atlas Detector at the LHC”. Bachelor Thesis. Technische Universität Dresden, 2020. URL: `https://cds.cern.ch/record/2747259?ln=de`.

[38] J.-E. Nitschke. “Studies on polarization in the scattering of two vector bosons with the ATLAS experiment at the LHC”. MA thesis. Technische Universität Dresden, 2019.

[39] W. Stirling et al. “Electroweak gauge boson polarisation at the LHC”. In: *Journal of High Energy Physics* 2012.7 (2012), p. 124.

[40] "MG3.1.0 undershoots events for VBS ssWWjj LT and TL production - Launchpad Question". 2021. URL: `https://answers.launchpad.net/mg5amcnlo/+question/697056`.

[41] F. Chollet et al. "Keras". `https://keras.io`. 2015.

[42] Martın Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[43] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[44] P. Azzi et al. “Standard Model physics at the HL-LHC and HE-LHC”. In: *arXiv preprint arXiv:1902.04070* (2019). URL: `https://arxiv.org/abs/1902.04070`.

[45] I. Goodfellow et al. "Deep Learning". MIT Press, 2016. URL: `http://www.deeplearningbook.org`.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Lisa Marie Lehmann

Digital unterschrieben von Lisa Marie Lehmann
DN: c=DE, o=Technische Universitaet Dresden, ou=Physik, cn=Lisa Marie Lehmann
Datum: 2021.06.07 12:23:31 +02'00'

Lisa Lehmann
Dresden, 07. Juni 2021