Discrimination of HH and HZ Final States Using Neural Networks

Master Thesis at the Faculty of Physics
of
Ludwig-Maximilians-Universität München

submitted by
Lars Linden
born in Essen

Munich, November 24th 2022

Unterscheidung von HH und HZ Endzuständen mithilfe neuronaler Netze

Masterarbeit der Fakultät für Physik
der
Ludwig-Maximilians-Universität München

vorgelegt von
Lars Linden
geboren in Essen

München, den 24.11.2022

Abstract

The Standard Model of particle physics is performing quite well when describing fundamental physics at small scales, but at the same time it is unable to account for some clearly observed phenomena, such as dark matter, leading to the conclusion that additional new physics models are needed. Within the Standard Model, di-higgs production processes provide a valuable insight into possible new physics models. The measurement of such processes proves to be difficult however and has not been conducted yet as a result. The reason for this difficulty is that production of a higgs pair is exceptionally rare and overshadowed by background processes, making it tough to achieve a clean classical analysis.

In order to deal with this problem an event classification using neural networks is applied, in particular to distinguish di-higgs events from one of its background processes, the production of a higgs and Z boson pair, based on event jet information in simulated Monte Carlo data. In order to improve classification results from such a previous analysis, experiments aiming to find additional variables sensitive to these two final states and able to aid a neural network in its classification task were conducted. Further, effects of changes to the provided neural network infrastructure were investigated. Both of these analyses efforts lead to a big performance improvement over the previous analysis. This results suggest that usage of neural network based event classifications is a viable method for enabling di-higgs production measurements and analyses.

Contents

Chapter 1

Introduction

Particle physics has done a successful job at broadening our understanding of the fundamental rules and functionality of our universe by constructing and extensively testing its fundamental theory, the Standard Model of particle physics. Countless of its theoretical predictions were verified to a high degree of precision, leading to the discovery of its final unobserved building block, the higgs boson, in recent history. However, the model is far from complete, because it is still missing explanations for observed phenomena such as dark matter. Further, the Standard Model's precise predictions were mostly verified by experiments, which is starting to become a problem since there are only a few areas left where theoretical uncertainty could lead to a further expansion of the model. One of these areas is physics concerning the higgs boson, which is still hardly understood despite its discovery a decade ago. In particular, the production of this boson in pairs is predicted but still remains undiscovered so far, making it interesting for potential physics models that go beyond the Standard Model. Because of this, measuring such a higgs boson pair production process with high accuracy can provide certainty for both the higgs boson's exact properties and the possibility of expanding the Standard Model with new and potentially verified theories.

One of the biggest obstacles for such a di-higgs production measurement is the extreme rarity of the process, combined with far more common irreducible backgrounds given by similar processes, overshadowing the desired signal. An example for a process like this is the production of a Z and higgs boson pair, which happens through a process similar to higgs pair production, including a similar final state, making them hard to distinguish. Facilitating this selection is an important step necessary for a proper analysis of pair production for higgs bosons to make sure a sample collection of these processes is clean and large enough to arrive at a definitive conclusion.

A possible approach to improve selection of the correct processes is to employ artificial neural networks for this data classification task. This work aims to improve a previous attempt to make this classification on simulated Monte Carlo data by providing event jet information to a neural network, with the goal of making it more accurate. For this purpose a variety of potential variables will be investigated regarding their abilities to distinguish between the aforementioned processes, to increase the amount of relevant information a neural network model will receive from its input data. Further, the model structure given by the provided neural network will be expanded and adjusted through various tests in order to expand its classification capabilities.

This work will firstly introduce the necessary theoretical and experimental particle physics background in chapter 2. Afterwards an introduction to artificial neural networks is given in chapter 3. Finally the conducted experiments are summarized, where chapter 4 presents the results of the searches for new variables and chapter 5 covers all investigated neural network experiments.

Chapter 2

Physics Background

Particle physic aims to explain the very fundamental constituents and laws of our universe. These elemental building blocks are called *elementary particles* and their behavior and interaction is described by the *Standard Model* of particle physics (SM), the theoretical basis for particle physics. This theory has been really successful over the years and was continually expanded to answer further questions, but still remains incapable to account for all phenomena observed in our universe so far.
This chapter will establish the theoretical concepts in particle physics, necessary for the considerations made in later sections as well as an outlook on other current developments inside of the field. Further, the basics of experimental particle physics present at colliders are explained, where a special focus will be placed on the ATLAS experiment currently operating at CERN.

2.1 Theory of Particle Physics

Elementary particles exist in two physical regimes due to their typical sizes and energies, which are described by different theories. Since these particles are very small, they must obey the rules of quantum mechanics and because they are highly energetic and move at speeds close to the speed of light, their behavior is influenced by relativistic mechanics as well. The theory combining these two descriptions into a single framework is called *quantum field theory* (QFT) and describes an elementary particle as an excitation of an underlying field at the smallest scale. These fields can interact with each other via the rules of the fundamental forces of our universe. As both, a simplification and visualization of QFT, Feynman invented diagrams named after him, making it easy to create pictures representing particle processes. An example for such a *feynman diagram* for a particle anti-particle annihilation can be seen in figure 2.1. Resulting from a description of all known fundamental forces in the universe, except for gravity, using QFT one arrives at the Standard Model of particle physics.

2.1.1 Elementary Particles

Elementary particles describe the smallest set of particles we know of without any apparent further substructure, which make up all visible matter as fundamental building blocks. These particles can be distinguished and classified into different groups via their quantum numbers. The most general separation is based on the spin quantum number in units of $\hbar$, where *fermions* are particles which have half integer spin whereas *bosons* have integer spin. Furthermore each elementary particle has a corresponding antiparticle, these particle-antiparticle

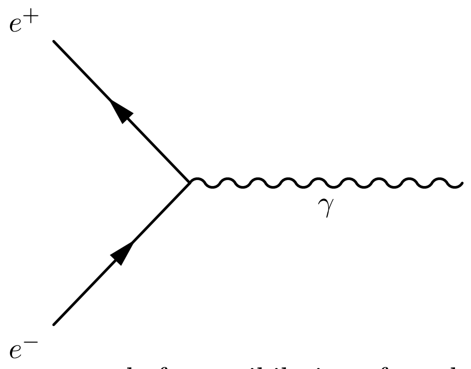

Figure 2.1: Feynman diagram example for annihilation of an electron positron pair via electromagnetic interaction, producing a photon. The vertical axis represents space, whereas the horizontal axis represents time.

pairs can be created from the vacuum or annihilate each other to create energy. An antiparticle has the exact same properties as the opposing particle, but charge-like quantum numbers have the opposite sign. There are two groups of fermions, the *leptons* and *quarks*, each with a total of six particles and three different so called families or generations consisting of particle pairs. The lepton families are given by a pair, which consists of a negatively charged particle ℓ^- and a corresponding charge and massless neutrino ν_ℓ. Out of the three massive leptons only the electron is stable, meaning the other two will decay in some form after a while whenever they are produced from a fundamental process. Quarks are grouped by their electrical charge into *up* and *down* type quarks with $+\frac{2}{3}$ and $-\frac{1}{3}$ of electrical charge respectively. The quark family arrangement is then done by grouping up and down types of similar mass value together. An overview of the different fermions can be seen in table 2.1. Quarks cannot exist as free particles and engage in a process called hadronization whenever not part of a bound state. Hadronization produces hadrons, bound states of multiple quarks,

	1st Generation		2nd Generation		3rd Generation	
Quarks	u	d	c	s	t	b
Charge	+2/3	-1/3	+2/3	-1/3	+2/3	-1/3
Mass	2.16 MeV	4.67 MeV	1.27 GeV	93.4 MeV	173 GeV	4.18 GeV
Leptons	e^-	ν_e	μ^-	ν_μ	τ^-	ν_τ
Charge	-1	0	-1	0	-1	0
Mass	511 keV	<1.1 eV	106 MeV	<0.19 eV	1.78 GeV	<18.2 MeV

Table 2.1: The three fermion generations for leptons and quarks, charges are given in units of the elemental charge e. The six quarks up, down, charm, strange, top and bottom are represented by their first letter, while the leptons electron, muon, tau and their respective neutrinos are represented via the letter e and the greek letters for mu μ, tau τ and nu ν. Values for neutrino masses are an upper limit since they are very light particles and exact measurements do not exist yet, which is why the standard model assumes them to be mass less. Values are taken from the pdg listing [1].

which can be both bosons in case of a quark anti-quark pair and fermions for a bound state of three quarks. On top of that there are five fundamental bosons in the standard model, which are responsible for mediating forces between particles. For simplicity in theoretical calculations, natural units given by $c = \hbar = 1$ are used. Energies and masses are therefore both given in the same units, for which commonly electronvolts (eV) are used. Gravity does not have a significant influence at scales of elementary particles and is therefore ignored for now. The effects of the other three fundamental forces (electromagnetism, weak and strong nuclear force) on all these elementary particles is then described in a complete theory, the Standard Model. This enables the calculation of observable quantities concerning particle reactions like scattering, decays and particle lifetimes [2].

2.1.2 The Standard Model

The SM describes everything in the life of a particle, but most importantly it provides a theory for interactions via three of the fundamental four forces in nature, weak and strong nuclear force and the electromagnetic force. Each of these forces has their own underlying theory with a *lagrangian* as a mathematical base for the theoretical description. These lagrangians were constructed in a way such that they would reproduce known equations of motions, for instance the dirac equation describing electromagnetic interactions for massive fermions, which were normally found based on phenomenology. Symmetries play a large role in physical descriptions because of their close connection to conservation laws due to Neother's theorem, which yields for instance a conservation of angular momentum from a rotational symmetry in a system. Beyond that other types of symmetries such as gauge and discrete symmetries are also relevant in these theories. All three forces have different characteristics for their interactions and each force acts on a different property of the particles involved. Electromagnetism acts on the electrical charge of a particle, while the weak force acts on a quantum number called weak isospin and the strong force acts on colour charge of particles. The latter being a property exclusive to quarks and therefore this force only affects quarks. Gauge bosons and their properties represent the field mediating an interaction and therefore share characteristics of the respective force and are normally massless. Terms in the lagrangians can be interpreted as different processes within the force, for instance a particle or gauge boson propagating freely. Adding the additional requirement of local gauge invariance leads to a new term which can be interpreted as an interaction of a free particle with a gauge field.
Global gauge invariance is derived from the observation, that adding a divergence term of the form $\partial_\mu M^\mu(\phi_i, \partial_\mu \phi_i)$ does not change the resulting equations of motion and has therefore no impact on the physical result. Demanding that such a global gauge variance should also hold locally, meaning the transformation now shows a space-time dependence, has proven to be valuable for particle physics, as shown in the following. The dirac lagrangian, which reproduces the dirac equation, is given by

$$\mathcal{L} = i\bar{\Psi}\gamma^\mu \partial_\mu \Psi - m\bar{\Psi}\Psi \tag{2.1}$$

and can be modified to be invariant under a local gauge transformation of the form $\Psi \rightarrow e^{i\theta(x)}\Psi$ by adding a term containing a gauge field A_μ transforming like $A_\mu \rightarrow A_\mu - \partial_\mu \frac{\theta(x)}{q}$ with a coupling constant q. The now locally gauge invariant lagrangian looks like

$$\mathcal{L} = [i\bar{\Psi}\gamma^\mu \partial_\mu \Psi - m\bar{\Psi}\Psi] - (q\bar{\Psi}\gamma^\mu \Psi)A_\mu \tag{2.2}$$

where the final term can be interpreted as an interaction of a fermion given by its spinor Ψ with a gauge field A_μ via a coupling constant q affecting the strength of this interaction. Each fundamental force has its own coupling constant, depending on specific particle properties, for example the strength of an electromagnetic interaction is given by the charge of participating particles. The first term describes free propagation of the fermion and the second term its mass m. Repeating the same steps for the electromagnetic boson gauge field A_μ yields an expression which cannot contain a mass term anymore, since such mass terms contain a factor $A^\mu A_\mu$ which breaks local gauge symmetry for boson lagrangians. In consequence this means that in order to eliminate this term, all gauge fields in the standard model need to be massless [3]. Even though fermions can theoretically be massive, visible in equation (2.2) above, they are still regarded as massless in the SM which allows for splitting right and left chiral parts of the spinor into distinct parts allowing an interpretation as separate particles [4].

This procedure works well for the electromagnetic and strong interactions, because their respective gauge bosons the gluons and photons are really massless, but the weak force is mediated by W and Z bosons, which are both massive and have been observed in experiment. Combining the electromagnetic and weak force into the electroweak theory did not help to explain this either, however the observed gauge bosons W, Z and photon were now created through mixing of massless gauge fields B^μ and W^μ via the weak mixing angle θ_W. This means that the only thing still needed to make the theory work is a way to explain the generation of masses for fermions and the weak gauge bosons. This theory was provided by Higgs in 1964, further predicting a new boson named after him, which could then be used in 2012 to confirm the theory [3].

2.1.3 The Higgs Mechanism

In order to tackle the problem of masses in the SM one needs to find a modification of local gauge invariance that allows for a mass term. This can be achieved by introducing a potential term to the lagrangian which has the characteristic of a nonzero vacuum state, meaning its minimum is not located at the origin. The existence of a lagrangian with such a potential can be interpreted as describing a background field, always existing even in vacuum. This field, called the higgs field, is essential for the creation of mass terms in the SM, made possible by the properties of its potential, as shown in the following. An example for such a potential of a complex field $\phi = \phi_1 + i\phi_2$ in three dimensions is given in equation (2.3) and illustrated in figure 2.2, this is the simplest possible potential showing the desired properties.

$$V(\phi) = -\frac{\mu^2}{2}|\phi|^2 + \frac{\lambda^2}{4}|\phi|^4 \tag{2.3}$$

Parameters μ and λ are not fixed by the theory and have to be determined experimentally. The potential has a rotational symmetry and its multiple minima all lay on a circle as well,

$$|\phi|_{min} = \pm\frac{\mu}{\lambda} \tag{2.4}$$

but in order to bring the lagrangian into a form showing couplings and eventually a mass term, one of the minima has to be chosen as the ground state in order to make the following coordinate substitution for the real and imaginary parts of ϕ

$$\eta = \phi_1 \pm \frac{\mu}{\lambda} \text{ and } \xi = \phi_2 \tag{2.5}$$

which breaks the symmetry. By further using a specific choice of gauge transformation $\phi \to \phi'$, allowing the resulting ϕ' to be real by demanding that $\phi'_2 = 0$, eliminates the second field ξ from the new lagrangian. As a result one obtains mass terms for gauge fields and a new massive particle described by the η field, the *higgs boson*. This procedure is known as *spontaneous symmetry breaking*, which describes the case when the symmetry of a lagrangian is not shared by its ground state, achieved here through the coordinate transformation performed above, and is the foundation of the higgs mechanism. The exact shape of the higgs potential remains unknown, but the underlying principle of using spontaneous symmetry breaking to generate particle masses always works the same, so reality might be more complex than described here, but still makes use of the same underlying principles and steps as discussed above [3].

In 2012 both ATLAS and CMS experiments at CERN confirmed the existence of a boson showing the predicted properties of a higgs boson, which validated the theory [5][6]. During the following years lots of further measurements were conducted, confirming couplings and branching ratios predicted by theory. A coupling of the higgs to masses of fermions for instance was already confirmed and measured for the heavy top and bottom quarks and most

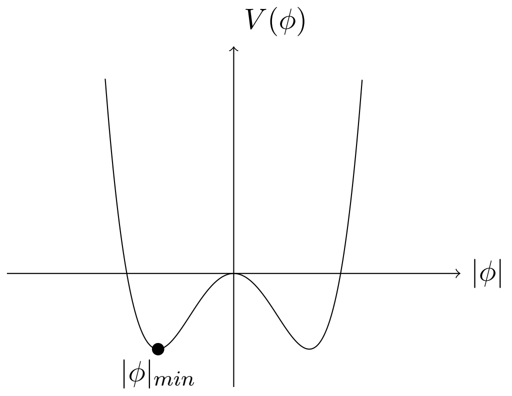

Figure 2.2: Cross section through the three dimensional higgs potential along an arbitrary axis containing the origin. Via the choice of a ground state $|\phi|_{min}$, used for description, breaking of the potential's rotational symmetry is achieved.

major decay processes have been quantified. The higgs boson was found to most likely be a spin 0 particle with even parity and has a measured mass of about 125 GeV, implying a high likelihood for a meta stable electroweak vacuum. The most likely production mechanism at a proton-proton collider such as the LHC is via gluon fusion with a cross section of roughly 48.5 pb at a center of mass energy of 13 TeV, which is relatively small making the occurrence of higgs events overall rare. A higgs particle mainly decays into a pair of bottom quarks ($\approx 58\%$) or a W boson pair ($\approx 21\%$). Apart from a series of measurements regarding higgs properties there are currently also various experiments searching for possible signs of an extended higgs sector or Standard Model extensions [1].

Higgs selfcoupling

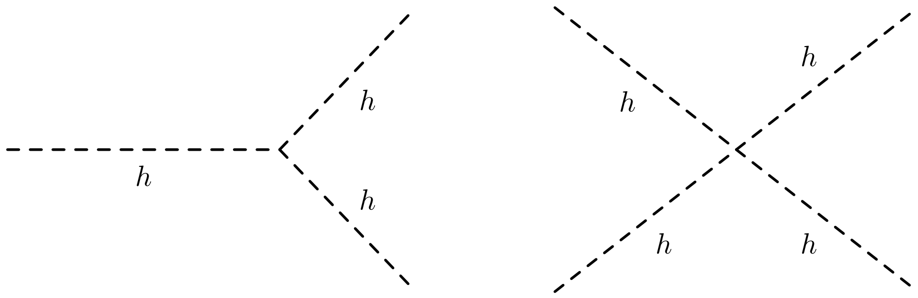

Figure 2.3: Feynman diagrams for the two possible self coupling processes of higgs bosons.

Spontaneous symmetry breaking does not only create a massive higgs particle, but it also leads to the appearance of further terms in the lagrangian showing that vertices where three or four of these newly created particles are interacting exist. So the higgs boson possesses the ability to self couple, meaning it can interact with other particles of its kind, as shown in figure 2.3. The terms appearing in the lagrangian are further proportional to the potential parameters μ and λ, where μ is also being proportional to the higgs mass, making it possible to learn more about the higgs potential by measuring these self coupling processes [3].
Higgs self coupling also plays a role as one of the possible processes responsible for producing higgs pairs in the final state. Searches for higgs pair production are an important measurement in the search for various models extending the Standard Model. However the existence of self coupling further diminishes the already small cross section of this process making it incredibly rare [1].

2.1.4 Open Questions

The SM is very successful in describing physics at the elementary scale and has been verified to a high accuracy. However nature still exhibits phenomena not explainable through this theory alone. Examples for this are dark matter and energy postulated based on observations in astronomy and cosmology, as well as the fact that our universe seems to mostly consist of matter. Furthermore this model has some inherent deficiencies since it does not offer a description of the fourth fundamental force of gravity and it cannot offer a solution to the hierarchy problem. These observations suggest that the Standard Model has to be expanded in some way in order to explain nature more accurately, these expansions of the Standard Model are called *new physics* or *physics beyond the Standard Model* [4]. The direction these expansions should have remains unclear since the theory seems accurate to an incredible degree making it tough to spot potential issues with the model. This problem leads to a large variety of searches for new physics ranging from precision measurements to tests of specific models beyond the SM. A selection of such investigations into new physics and their results are discussed in the following.
The higgs sector is rich of potential investigations, since experiments so far might have confirmed its existence, but values found in these experiments can still be explained through multiple different models aside from the most simple one used in the SM. Further uncertainties concern the fact that some higgs decay channels were not measured yet due to experimental challenges and it is still unclear whether the higgs is even an elementary particle opposed to a composite one. Especially measurements of the self coupling of higgs bosons, which is proposed in the SM, but has not been measured yet due to its rarity, are important since it provides a way to measure the trilinear higgs coupling. Additionally searching for resonant higgs boson pair production is important for several new physics models. As a conclusion the higgs sector is currently an important field in the search for new physics [1].
There are a few recently found numerical deviations from the SM in specific experiments which have not yet been explained, possibly hinting at the explicit nature of new physics. Some of these are a deviation of the measured muonic dipole moment by $4.2\,\sigma$ at the g-2 experiment at Fermilab [7] and a significantly larger measured mass of the W boson by $7\,\sigma$ obtained from a precision measurement conducted by the CDF collaboration [8].
Due to the lack of guidance from the SM, many different models and theories for new physics outside of the SM have been created. One such theory involves introducing right-handed neutrinos to the SM, using the see-saw mechanism to explain generation of neutrion masses and especially their small value. Another model is the WIMP model for dark matter, introducing very heavy particles only interacting via the weak force to explain the nature of dark matter. Neither WIMPs, nor right-handed neutrinos have been discovered by an experiment so far [9].

2.2 Experimental Framework

Particle physics is mostly concerned with processes at high energies which means that it is studied at particle colliders which can provide this high energy environment by colliding highly energetic particles with each other. This will then produce various particles, most of which will decay due to their instability, making analysis of physical processes present in those decays possible. Another way to test our understanding of particle physics with this setup is to directly produce particles predicted by theory and identify them through their expected final state properties. A significant amount of work in the field of experimental particle physics has been done at the various colliders present at the european center for nuclear research CERN in Geneva, Switzerland.

This section will give an introduction to important scientific results obtained at CERN in the past, its current methods and goals as well as future challenges and perspectives. Further an explanation of detector experiments will be given with a focus laying on ATLAS since the scope of this work only considers an analysis to be conducted at this experiment.

2.2.1 History of CERN

The first physics run at CERN started in 1958 after construction of the Synchro-Cyclotron, which could reach proton energies of 600 MeV. This first accelerator was used to study meson physics and discovered new decays of pions. It also contributed to μ physics for instance by measuring the constant g-2. Around the same time the more energetic Proton-Cyclotron was build, producing proton beams of 1.4 GeV energies and creating more advancements in beam physics and accelerator technology. The achievements and insights gained from this experiment were mostly of organizational and technological nature laying the foundation for the next generation of accelerators. After construction and usage of more new colliders, for example the proton-antiproton collider IRS, knowledge of accelerator physics improved more and more which then led to the construction of a powerful electron-positron collider called LEP. This collider started running in 1989, colliding beams at the energy of the Z boson at first and later on increasing the beam energies to produce W pairs. LEP was very successful in producing numerous results confirming the Standard Model, but discovering the higgs boson remained out of reach [10]. From 2001 onward LEP was replaced by the large hadron collider LHC, a proton-proton collider which started running in 2007 and is described in more detail in section 2.2.2. This collider made studying higgs boson physics finally possible and lead to its discovery in 2012. After multiple luminosity and detector upgrades the LHC will finish running in 2038. Current plans suggest a large 100 km future collider using proton-proton collisions at energies of the order 100 TeV, to probe the existence of weakly interacting dark matter particles, as a long term goal. As an intermediate stage an electron-positron collider, energetic enough to function as a higgs factory, is planned making more precise measurements of the electroweak theory possible. Regardless of the results these future initiatives will further deepen our understanding of the fundamentals of the universe and lead to more technological advancements similar to the achievements CERN has already reached in the past [11].

2.2.2 The Large Hadron Collider

The LHC is located inside of the roughly 27 km long circular tunnel beneath CERN previously used for the LEP experiment. Within this collider bunches of protons are colliding at a center of mass energy of 14 TeV achieved by having two proton beams traveling in opposite directions and intersecting at various collision points located around the ring, which lay inside of the several detectors. Proton beams are guided and focused using strong, superconductive dipole and quadropole magnets. The detector output produced by a single beam collision is called an event. The number of times a specific event is created N_{event} is given by the relation

$$N_{event} = L\sigma_{event} \tag{2.6}$$

only depending on the event cross section σ_{event} and the machine luminosity L. Since rare events are characterized by a small cross section, large luminosities need to be achieved at the LHC to make measuring these rare events possible. Luminosity on the other hand is only dependent on properties of the collider, mainly the beam properties, so in order to allow measurements of rare events high beam energies and intensities are required.
There are four detectors located around the LHC belonging to different experiments. These

are LHC-B investigating B physics, ALICE focused on ion experiments, using collisions of ion beams and two all purpose detectors ATLAS and CMS collecting data of proton-proton collisions [12].
One of the goals for the LHC was achieved in 2012 by discovering the higgs boson, since this verified the theory for mass generation of elementary particles via a higgs field. However this was not the only unsolved question in particle physics the experiments at LHC wanted to answer. Further problems include the nature of dark matter observed in our universe and finding an explanation for the apparent matter-antimatter asymmetry [13]. The SM so far seems to be unable to explain these phenomena, while also not showing any obvious signs of problems within the theory. This is why CERN conducts precision measurements of rare and forbidden SM processes to search for possible inconsistencies of the SM, which could give hints to the nature of a desired theory needed to explain unanswered questions as well as a deeper understanding of the higgs particle [14]. In addition there are a lot of analyses regarding new physics beyond the SM at LHC trying to find new particles associated with new physics models developed as possible explanations for open questions. Examples for this include a more complicated higgs sector with an additional scalar field, allowing for dark matter coupling and heavy neutral neutrinos, related to a mechanisms for generating neutrino masses [9].
Future upgrades to the collider entering the second phase of the LHC program aim to provide a significant increase in luminosity, which will be achieved by further improving the beam parameters. This high luminosity LHC (HL-LHC) will be able to produce a significantly larger amount of data, leading to various improvements for different analyses. In the case of higgs boson studies the high luminosity enables more precise results for precision measurements and in addition makes studying rare higgs decays and even di-higgs production and higgs self-coupling possible. Further direct creation and observation of rare new particles becomes more viable, enabling tests for various new physics models. It is expected that the significantly larger dataset provided by the HL-LHC will improve its ability to detect new physics directly by discovery of new particles, or indirectly through precision measurements significantly [15].

2.2.3 ATLAS Detector

Detectors at particle colliders are used to determine all particles present in the final state of a collision so the collected data can be analyzed later. A detector is made up of multiple parts with different responsibilities. These include a tracking device to find particle trajectories, different calorimeters to capture certain types of particles and measure their energy and a muon detector for collecting muon information. Tracking is done by the inner detector which uses a strong magnetic field and pixel detectors to locate charged particles, measure momenta and identify electrons. The first calorimeter, the electromagnetic calorimeter, is located as the next layer above the inner detector. Here precision measurements of electron and photon properties take place. Around this the hadronic calorimeter is placed, where hadrons decay leading to their quarks hadronizing, which produces a large number of lighter hadrons in the final state. These roughly share the same direction of travel the original particle had, which leads to the created particles being distributed within a cone surrounding it. When processing the data, an approximation of the properties possessed by the original particle can be found by searching and combining closely located final state particle into a single one. The resulting combination of these final state particles, representing the original particle, is called a *jet*. Apart from measuring hadronic decay products, the measurements of missing transverse energy also takes place within the hadronic calorimeter. Finally, located on the very outside of the detector, is the muon system, which is again inside of a magnetic field as the inner detector. Within this part of the detector muons are identified and their tracks measured. The ATLAS detector is shown in figure 2.4.

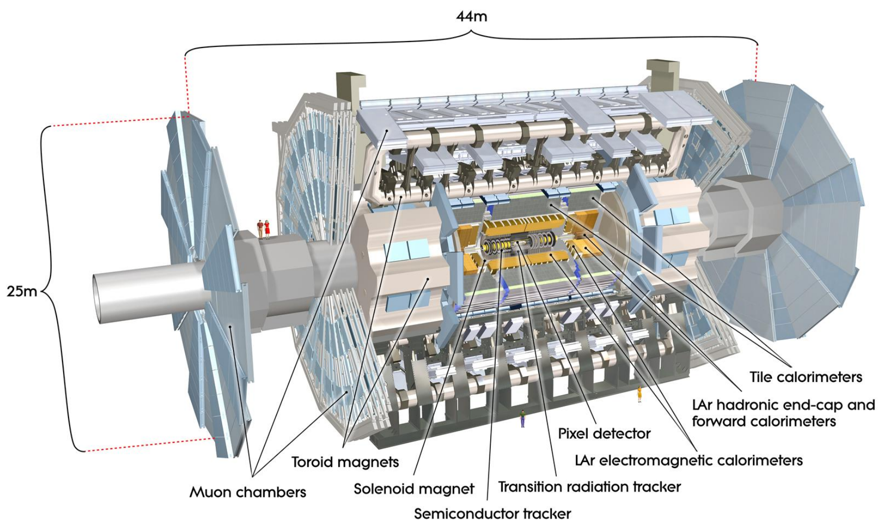
44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

Figure 2.4: Schematic cross section through the ATLAS detector highlighting the most important components for particle detection and measurements [16].

For the design and performance goals several of the intended experimental measurements to be conducted at CERN were taken into account, for example possible higgs decay channels needed for its discovery. Since the possible decays depend on the mass range of the higgs, which was unknown at the time of construction, the detector would have to be able to perform well enough for a discovery regardless of the scenario. Similarly discoveries for several new physics models such as SUSY particle and heavy gauge boson W', Z' measurements should be possible, leading to even more performance requirements, such as for instance lepton measurements with a resolution up to TeV in p_T for the heavy gauge boson decays. Furthermore, the main challenge of proton-proton collision is that QCD backgrounds are large, making discoveries of rare processes even harder, since specific characteristics of these desired rare process must be identified. This leads to the additional requirements of a large integrated luminosity and particle identification abilities of the detector. The detector components mentioned above were all constructed with these challenges in mind and try to go as far as currently technologically possible to fulfill all requirements.
Due to the high rate of generated events far exceeding the possible speed of data recording, many events need to be rejected before even finishing to process them. This work is done by the trigger system which is responsible for deciding whether or not an event is of interest and should be recorded. There are multiple layers to the trigger system which lead to a more specific reduction of data in each step. This ensures that data obtained will most likely be of interest for current experimental searches and analysis, in the case of LHC research most of these are characterized by high transverse momentum or a large amount of missing transverse energy, which are the main criteria for the triggers.
The coordinate system used by the detector and within this work is described as follows. The origin lays at the point where the protons collide, with the z-axis pointing along the beam axis. The x-axis points towards the center of the LHC and the y-axis points upwards. The azimuth angle ϕ is measured around the beam axis and the polar angle θ describes the angle from the beam axis. However instead of the polar angle often pseudorapidity η given by

$$\eta = -\,ln\ tan(\theta/2) \tag{2.7}$$

is used. Further the transverse momentum, energy and missing energy p_T, E_T and E_T^{miss} are defined in the x-y plane [16].

Chapter 3

Artificial Neural Networks

Among the numeral machine learning techniques introduced over the years, Artificial Neural Networks (ANN) and especially their deep variants, have become the most popular way of approaching machine learning in recent times. They have proven to be highly capable of a big number of different tasks, producing results rivaling or even outclassing human performance as well as outperforming other machine learning techniques. The main strength of ANNs lays in their ability to learn from large amounts of data and use the acquired knowledge for a variety of different applications [17]. ANN are commonly used nowadays for a wide range of problems within the field of data analysis, including face recognition on images, automated language translations, creating self driving cars and a lot of applications within medical analysis [18]. Furthermore ANNs can be used for the task of classifying data into different categories, which is a particularly useful application when trying to find specific processes within datasets created by particle physics experiments.
The following chapter will introduce the basic theoretical concepts of how an ANN is structured and the procedure it can apply to learn correlations and connections of data, in order to solve a wide variety of problems. Furthermore common issues that can appear when trying to work with an ANN as well as possible solutions and workarounds are presented.

3.1 Motivation for Artificial Neural Networks

Artificial Neural Networks are – as the name suggests – based on our current understanding of the human brain and nervous system, trying to mimic their structure and functionality in order to replicate their enormous information processing capabilities. A nervous system consists entirely of a large number of neural cells, called *neurons*, which can be combined into various complex structures. In a simplified way the main properties of neurons are a cell body with complex interior for processing information and a way to receive and transmit it. Further, an information storage capacity exists at the contact point of two neuron cells. Artificial neurons simulate these properties and therefore have input, output and a cell body for processing and information storage as well. Furthermore the flow of information will always occur in a specific direction in a biological context which is represented in an ANN by using directed graphs, an example of which is depicted in figure 3.1. For a neuron to actually react to an input a certain activation threshold must be surpassed by the electrical input signals. This is combined with each input signal being assigned an importance, since the produced activation strength varies, depending on which connection the signal originated from. To include these observed functionalities in an artificial structure, numerical *weights* for each connection as well as a specified *activation function* for each of the artificial neurons are introduced [19].

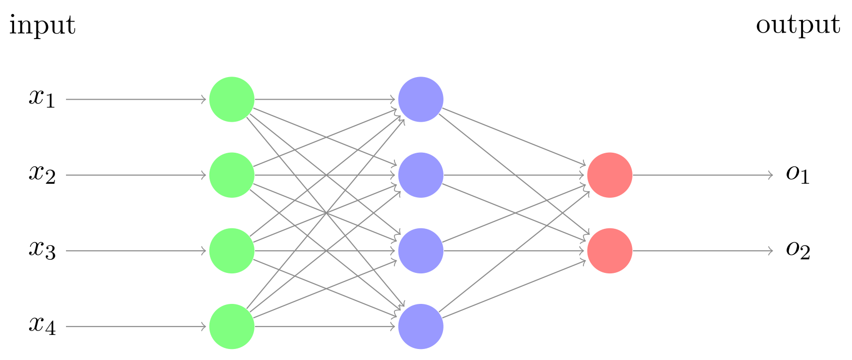

Figure 3.1: Example of a feed forward neural network with 4 different inputs x_{1-4} entering an input layer (green) of size 4, which the network then converts into 2 output values $o_{1,2}$ of the output layer with 2 neurons (red) via the use of a hidden layer (blue) with 4 neurons

Using these basic building blocks one can assemble an information processing network, which in the following will simply be referred to as a *Neural Network* (NN). The following sections will now show how learning can be achieved using NNs.

3.2 Learning in a Neural Network

After introducing all the essential parts needed, they can now be combined to form a full neural network. An example for a simple NN is shown in figure 3.1, where information in the form of data is fed into the network to the left and is converted into an output, while traveling through the network towards the right, which can then be interpreted as a classification for instance. This means that in general a Neural Network will receive input data of a set size and produce an output of a set size based on these inputs, while essentially functioning as a black box, because their inner proceedings are not easy to interpret. This section will deal with the intricacies and mathematical formulations that make this procedure possible, while also focusing on the features enabling a network to find and learn relations between data points and use this information to make predictions.

3.2.1 Structure of a Neural Network

A NN is a graph of interconnected artificial neurons arranged in a way that information traverses the network from start to end without any loops, by collecting neurons in so called *layers*, where neurons share the same input and output directions without distributing information among themselves. Every neuron can typically take on any real value. The collection of neurons belonging to the same layer are arranged in a vertical line of neurons in figure 3.1. There are three general classes of layers: The input layer on the very left that accepts the data input, the output layer on the far right, producing final values in its neurons which can then be accessed and interpreted to get a result and all the layers in between called hidden layers. As a result the input data as a whole is passed from layer to layer, affecting all neuron values, which in the end generates a network output. NNs are commonly classified based on their hidden layer structure as deep for a large number of layers, and wide for a large amount of neurons within each hidden layer. Every single possible connection between two neurons has an associated weight, which influences the importance of all received input values of a neuron when calculating the propagation of data through the network. A layer can also have different types depending on how its neurons are connected to the next layer, which are further explored in section 5.2.3. For now only *fully connected* or *linear* layers will be

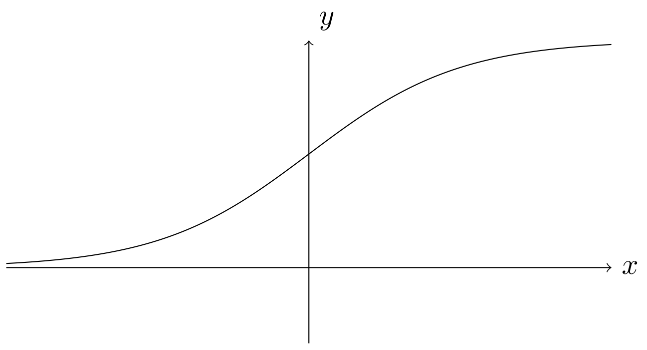

Figure 3.2: Plot of the historically used sigmoid activation function, chosen due to its similarity to measurements of the biological activation function.

considered, where each neuron of a layer is connected to every single neuron in the next layer. Located inside of neurons are their respective activation functions used when calculating a neuron value from its inputs [19]. An example for a possible activation function is shown in figure 3.2. This is the Sigmoid s_c activation function, given by the expression

$$s_c(x) = \frac{1}{1+e^{-cx}} \tag{3.1}$$

where c is a constant which can be chosen freely. The main reason for using a Sigmoid is motivated by its closeness to measured biological activation functions. However, nowadays it is clear that other types of activation functions, for instance ReLU and its variants are more beneficial for NNs. A more detailed explanation of these activation functions can be found in section 5.2.4. It was shown that ReLU outperforms the more classical approach with Sigmoids in deep networks [20]. Another advantage of sigmoids, which is also shared by ReLU, is that its derivative, given by

$$\frac{d}{dx}s_c(x) = \frac{c\,e^{-cx}}{(1+e^{-cx})^2} = c\,s_c(x)(1-s_c(x)) \tag{3.2}$$

can be expressed through the original sigmoid function, which is helpful for computations that are required in the learning step explained in section 3.2.3, improving computational speed. Further, values in the network are restricted to a range between 0 and 1, which helps keeping activations from becoming too large.
When calculating the output of a network one needs to find the values of all neurons in the network given by the input data and weights of neuron connections. The values of a neuron in the next layer will be computed from the values of all neurons connected to it from the previous layer multiplied by the respective weight of the connection, resulting in a weighted sum of neuron values. This weighted sum is then passed to the activation function of the layer and the final result of all calculations is then assigned to the neuron as its value. A simple example for this procedure is visualized in figure 3.3. As a result the general way to calculate a neuron value v is given by

$$v = f(\sum_i n_i\,w_i) \tag{3.3}$$

where i is the number of connected nodes from the previous layer with respective values n_i and connection weights w_i, while f denotes the activation function. This is repeated until the output layer is reached, where the values assigned to the output neurons allow for a conclusion, for instance the classification of an image, where each output neuron value represents the probability of the input belonging to a certain class [19]. Because the weights are normally

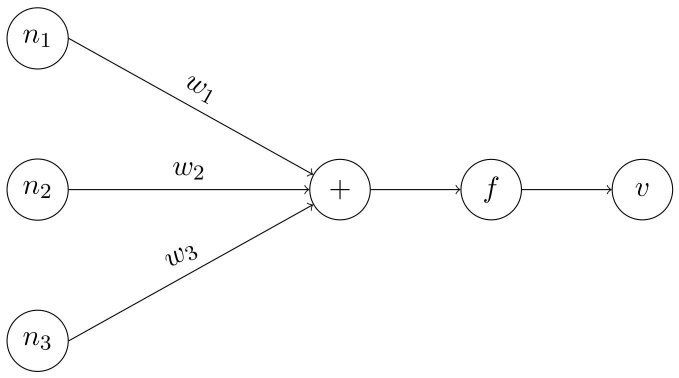

Figure 3.3: Example for calculating the neuron value v of a neuron connected to three neurons (n_1, n_2, n_3) in the previous layer via weights w_1, w_2, w_3 using an activation function f. First the weighted sum gets created from the inputs and weights, which is then passed to the activation function to produce the neuron value. In this case the value of v can be calculated via $v = f(\sum_{i=1}^{3} n_i\, w_i)$.

chosen from a random distribution in the initial state of a network, this result will be bad in general. Hence, in order to make learning possible using this structure further methods are necessary. More precisely a way to measure the accuracy of the NN output is needed which will be the focus of section 3.2.2. Further, a method allowing the network structure to use this information to learn from its mistakes is needed, which will be discussed in section 3.2.3.

3.2.2 Error Function

The error function of a neural network will give an estimation of the prediction quality a given neural network provides in each step. There are multiple ways to realize an error function and the most effective method depends on the problem at hand. To make usage of an error function possible in the first place, a labeled dataset is required, meaning that each data point has an identifier (often called *target value*) which takes some kind of numerical form, describing the network output this given data point should produce, so the results predicted by the network can be compared to the expected result. The goal of learning is to make the network generate an output, which is as close as possible to these expected values. In general the error function is given by a non-negative function of the network weights, that yields zero only if all output values are identical to all targets for a given dataset [21]. During training the error values start rather large since weights are normally initialized in a random fashion and should decrease with each successful training step.

Example: Mean Square Error

In a network with n output nodes producing the values o_i and their corresponding targets t_i with a label $i = 1, ..., n$, where n is the total number of data points in the dataset, the Mean Square Error of the network is given as

$$E = \frac{1}{2} \sum_{i=1}^{n} |o_i - t_i|^2 \qquad (3.4)$$

This example also shows that the error function can be regarded as a function of the network output and therefore can be seen as a composite function of all the activation functions and neurons in the entire network. As a result the error function will inherit properties from the activation functions such as differentiability and continuity given these properties are provided by all activation functions [19].

Because a good result is characterized by a small valued error function, the problem of learning in the network can be described as finding the minimum of the error function. So a requirement for learning in a NN is to minimize the error by making subsequent changes to the general architecture to slightly alter the output results. Since the only free parameters of a network are the weights, these will have to be slightly altered in each step until a minimum, or at least a state close to the minimum, of the error function is found.

3.2.3 Backpropagation and Gradient Descent

The idea behind making learning possible in a NN is to let the network learn from its (classification) errors by slightly adjusting the weights, going backwards from the final to initial layer in a way that ultimately minimizes the error function. This process is called *backpropagation*, which is a commonly used algorithm for implementing learning in neural networks. This entire problem is essentially the same as minimizing the error function in weight space, which is achieved by the backpropagation algorithm using a method called *gradient descent*. This means, that updates of the weight values are done in small steps into the direction the gradient is pointing to in each calculation step, since the gradient of an error function points towards a minimum in high dimensional weight space. This procedure is repeated until a minimum is reached, the corresponding weight values are then the most optimal parameters of the network found through training.
Explicitly this means that one wants to calculate the gradient of the error function E with respect to the entire network, so all calculations that arrived at the output are taken into account, and find the minimum of E, which is the state of the network which yields $\nabla E = 0$. This also shows that a useful activation function must be differentiable and its derivative should be easy and fast to compute for a computer, in order to not waste too much time during computation (e.g. ReLU, Sigmoid, etc.). This also ensures that the error function is both differentiable and continuous. Because of this the gradient of the error function is given by

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_m} \right) \tag{3.5}$$

for a total number of m weights in the entire network. Using the method of gradient descent we now get a way to update the weights after each step defined as

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \text{ for } i = 1, \dots, m \, . \tag{3.6}$$

The parameter γ in this context describes a constant of learning, influencing the step size, called *learning rate*. Its value can be chosen freely but should be kept relatively small to make sure the learning process converges towards a solution in weight space.
Backpropagation is now achieved by explicitly calculating the gradients of the error function regarding the weights, which will, due to the error function's composite structure and the chain rule, result in a calculation backwards through the network.
Instead of using the calculated output neuron values for calculation, the "input" value propagating through the network from end to start is 1 in backpropagation. Because of this the output value of a neuron is regarded as a constant in backpropagation, but the gradient is calculated implicitly with respect to the final output value. The results of backpropagation up to a certain node is stored as a value inside of the node (called *backpropagated error δ* from now on) together with its output value calculated when determining all neuron values.

Using all of this information for all nodes in the network one can now achieve a learning algorithm used to update the weights based on gradient descent and backpropagation.

Example for a two-layer network

Considering a simple network with only one input and output layer each and no hidden layers, where w_{ij} denotes the weight of a connection between input node j and output node i. Backpropagation computes the gradient of the error function E with respect to an output o_i, treating it as a constant which means the gradient of E with respect to the weight w_{ij} can be expressed as

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i \partial w_{ij}} \tag{3.7}$$

where the right hand side can be further simplified to $o_i \delta_j$ by expressing the partial derivative part as the backpropagated error at node j called δ_j. Inserting this into equation (3.6) one arrives at an explicit expression for the weight update given by

$$\Delta w_{ij} = -\gamma o_i \delta_j \; . \tag{3.8}$$

For networks with hidden layers one would need to repeat this step for each layer starting from the output and also take all possible routes through the network possible from the current node location into account. This will in general lead to more complicated expressions for δ depending on the previously calculated layers, but the general method remains the same. It is important however to only update the weights *after* all calculations for the backpropagated error are done, to avoid corrections influencing backpropagation calculations of the current step and therefore produce wrong gradient directions [19].

3.2.4 Validation

It was shown, that by providing labels of expected results a given set of data should produce, the network can use an error function and backpropagation to learn from its mistakes. This will ultimately lead to a point where the network is able to reproduce values similar to the targets from the input data. At this point it is still unclear however, if the network simply learned to reproduce the given dataset or if learning of underlying principles in the dataset really occurred. This is an important distinction, since in reality the use case for NNs is to classify data on their own, without the ability to check the correctness of labels produced by the network. To make sure the network has actually learned something from the data it was provided, its performance has to be judged on data not used for this training step yet.
For this reason providing a setting where the network is forced to classify data it has never seen and therefore also never used in training before, during the process of creating a network is absolutely necessary. This is achieved by splitting up the dataset into data points used purely for training and optimizing the network referred to as *training* set and another set used to simply judge performance, without further adjusting weights, on unseen data called *validation* or *test* set. Because all of this is labeled data the NN performance on validation data can be judged and the result be taken into account for further network design. The ability of a NN to classify unseen data well, based on a good classification of the training data is called the generalization ability of the NN. A network that generalizes well has no problem labeling unseen data correctly and is therefore desired [21].

3.3 Issues of Neural Networks

Because of their black box nature and statistical predictions, NNs suffer from a variety of issues which have to be accounted for when applying them to solve problems. In the following a short list of the biggest and most common issues, as well as possible solutions and ways to avoid the problems outright, will be discussed.

Underfitting

Underfitting describes a case where the NN cannot learn the inherent structure of the data leading to a bad performance in both training and testing. It is normally caused by a network model which is too simple and therefore has not enough available parameters to accurately fit the data. Solving this problem is normally relatively easy, since introducing complexity into the network is generally easy to achieve by adding more neurons and/or layers. This however comes at the risk of replacing the problem of underfitting with overfitting.

Overfitting

Similar to underfitting, overfitting describes another case where the network is unable to learn, but this time instead of not being able to model the data in training at all, the network learns everything from the data, even including noise, which leads to a high error when exposing the network to data it has not seen before, due to noise being unrelated to the general data structure. Consequently a network which shows signs of overfitting learns well in training but generalizes poorly. The main cause for this is that a network has too many free parameters, which are then used to memorize the input data. However reducing the model complexity is not always possible, which is why other ways of dealing with this problem are needed as well. The techniques used to fight overfitting are called regularization. Because overfitting is one of the most prominent issues users of neural networks have to deal with, there exist many different ways to implement a regularizing effect into a network. The main ones are L1 and L2 regularization, dropout and batch normalization. Another way to reduce overfitting is

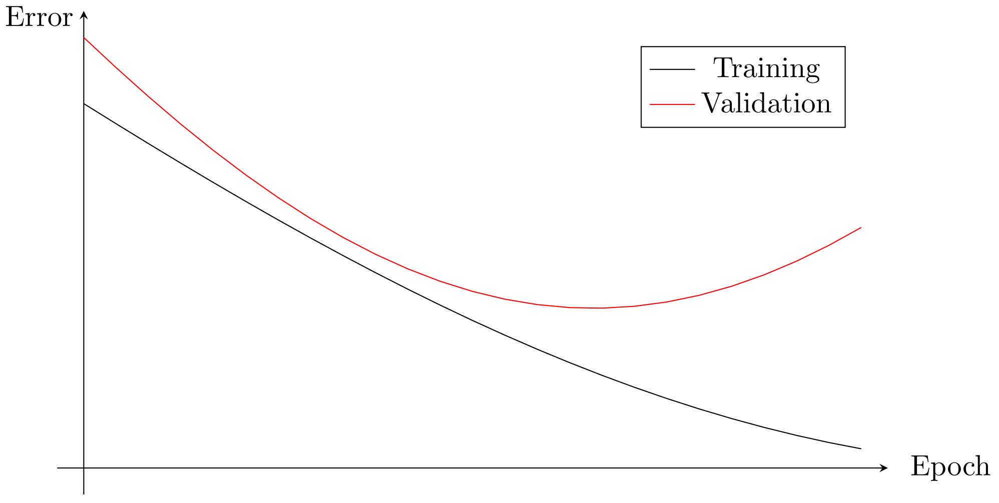

Figure 3.4: schematic comparison of error performance in training and validation data for an overfitting NN. When the gap between the graphs begins to increase the network entered a state of overfitting. For underfitting the error would not show a significant decrease in value instead.

also to provide bigger datasets to the networks, since this increases the amount of parameters a model can have without being capable of memorizing the dataset [21]. The details of some of these methods are discussed in section 5.3. Figure 3.4 shows a visual representation of the effects of overfitting on the error function values in training and validation data.

Data requirements

Neural Networks as a whole, but especially deep NNs require a large amount of data to assure best possible performance and avoid overfitting as mentioned above. Most of the time there is a sufficient amount of data available, but even when this is not the case there are ways of expanding the available data to circumvent this issue. One such method is data augmentation, where additional data is for instance created by mirroring or rotating existing data points. Especially when working with images this method has proven useful, but it can also be used for any kind of numerical dataset to introduce new points in data space [17].

Computational issues

Because of the need for big datasets and large network structures with lots of parameters, the memory requirement for using NNs becomes enormous for certain use cases (e.g. healthcare). Furthermore a large network requires a lot of computations to derive all needed values for arriving at an output value, which makes the NN require a lot of computational resources as well. This means that running complex networks on computers with limited computational power becomes impractical or at least very time intensive. One approaches to fix this is model compression, where the number of needed computations can be reduced by a variety of simplifications, whereas another way of solving this is using parallel processing provided by different hardware such as GPUs [17].
In addition the introduction of quantum computers in the future might help with solving computational shortcomings of current hardware. Quantum algorithms for neural network implementations are already in development and it is expected that a working quantum computer hardware will lead to a significant speedup in performance for learning with neural networks [18].

Chapter 4

Preliminary Analysis

Before the actual classification of final states using neural networks is investigated, a further analysis step to identify potentially useful jet variables needs to be conducted. These will then be used in the network model tests to provide new information in addition to the jet variables of energy, momenta and particle number already used in the previous analysis work [22]. In order to achieve this performance goal, the new variables have to be capable of distinguishing higgs pair events (HH) from its main background, the production of a higgs and Z boson pair (HZ). The analysis presented in this chapter was conducted to find such variables, which can then be tested in a neural network environment afterwards. Several of the investigated variables and their performance regarding final state classification will be discussed here.

4.1 Methodology

The analysis uses two Monte Carlo datasets generated using the programs POWHEG [23] and PYTHIA 8 [24], both of them containing a total of 1.2 million events each. One of them is the dataset producing signal events, a higgs pair final state from gluon fusion, while the other contains processes created from gluon fusion with a HZ final state. All possible decay modes of higgs and Z bosons are allowed and no detector noise or other types of irreducible background are added. These datasets are then further analyzed via ROOT [25]. Feynman diagrams of the most common physical processes of HH final states are shown in figure 4.1, while figure 4.2 shows the dominant diagram for the background process, where a higgs boson is radiated from a produced Z boson instead, leading to a HZ final state.
This analysis aims to find variables, that are sensitive to the differences found between these

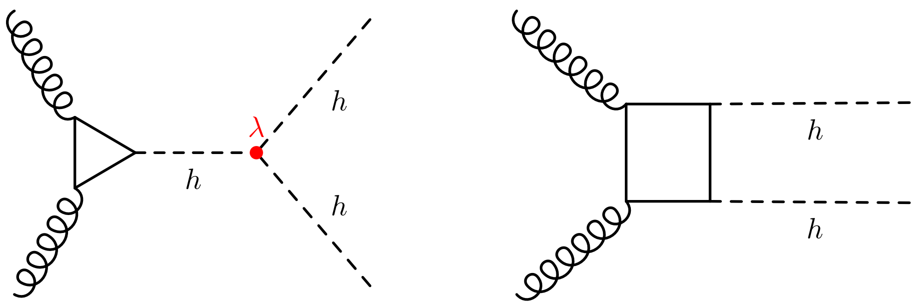

Figure 4.1: Dominant diagrams for higgs boson pair production at CERN. On the left is a self coupling interaction which has a strength dependent on the higgs potential parameter λ. On the right a box diagram where two higgs are produced independently with their coupling strengths depending on the yukawa coupling y is shown.

two final states. In order to keep matters relatively simple, only the most common decay mode of higgs particles shall be used for analysis, which is a decay into b quarks. This simplification is chosen since most decays of HH final states create at least one b quark pair ($\approx 82\%$) and roughly a third of decays are exclusively into b quarks [26]. As a result this approach should still cover a significant fraction of total processes. In the end jet information is supposed to be analyzed, meaning properties of jets resulting from b quarks are compared for both different final states. Further, comparisons between jets from b quarks and all jets inside of an event, may show ways to reduce statistical background, which in the end might also benefit the neural network. Lists of jets for each event are produced using algorithms provided by FastJet [27]. Jets are created via the exclusive variant of the *kt algorithm*, using a radius parameter of $R = 1$ and a dcut parameter value of 400 for clustering.

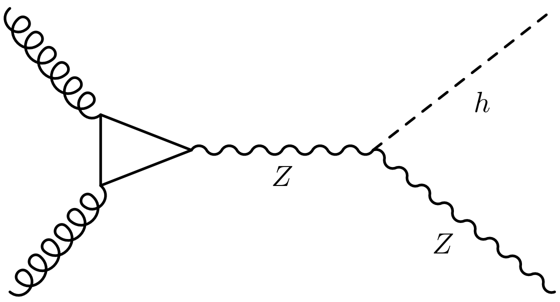

Figure 4.2: Main production process for the background HZ final state via radiation of a higgs off of the created Z boson. This process is very similar to a HH final state from a self coupling process making it hard to distinguish them.

B jets are formed by using the list of b quark information provided by the Monte Carlo generator and adding up 4-vectors of particles in close vicinity to the decayed quark. The final b quark jet will be made up of particles located within a cone of radius 1 around the original b quark 4-vector. Jets of b quarks created that way will be referred to as *cones* as opposed to jets from the kt algorithm from now on. Only pairs of b quarks that have a combined invariant mass of roughly the higgs mass at 125 GeV are considered for this step, where the majority of pairs in this group is expected to truly come from a higgs decay due to its high probability to decay into b quark pairs and two higgs bosons being present in all events. Further, cone pairs, where at least one of the cones contains only one or no particles at all are discarded, leading to a minimum number of two particles making up a cone. This step ensures that all of the quark cones in the analysis are actually relevant and have a high chance of being created by a decayed higgs without directly relying on Monte Carlo exclusive information. Furthermore the cones created this way are expected to have similar properties as the actual jets b quarks would produce, while also containing statistical background producing a similar jet pair mass, making it possible to investigate differences between all jets and only those jets originating from higgs decays while including some of the statistical uncertainty prominent when creating jet pairs to reconstruct decayed particles. No implementation of b tagging was used for this analysis, however this technique is of great value when identifying b jets in experiment data and should therefore be used outside of simulated data.

4.2 Variables

In this section the main variables that show the most potential for distinguishing between the datasets are presented. Hereby the focus will lay on variable values characterizing jet pairs, or in other words, variables showing the relation between potential decay products of the higgs or Z bosons from their decay into a b quark pair. Further, these variables are able to hint at useful characteristics of jets to be included in a network dataset. As a result not only the variable related to the jet pair, but also the jet variables used to calculate it will be provided to the network.

4.2.1 Relative Mass

Since the higgs decay into a quark pair leads to the creation of two identical particles, their properties and therefore also the properties of their final state particles produced through hadronization, should be pretty similar. The idea behind this variable is therefore, that the jets created from these final state particles are expected to share these similarities. Especially the invariant masses given by the jet 4-vector variables should be of around the same size for pairs originating from a higgs decay, making for a great variable identifying possible b jet pairs created through the same decay opposed to a random jet combination only having the jet mass by pure chance. As a result this variable is expected as a possible way to reduce statistical error when reconstructing higgs particles from jet pairs. The resulting *Relative Mass* variable for a pair of jets is given via

$$m_{rel} = \frac{|m_1 - m_2|}{m_1 + m_2} \tag{4.1}$$

which takes on values from 0, for the case of jets with identical mass, to 1 when one of the jets is significantly heavier than the other. In theory this variable provides a possible statistical background recognition or even reduction tool for the neural network since quark pairs from higgs or Z decays are expected to have mostly small values, while the distribution for a collection of randomly formed jet pairs is expected to be even on most of the interval.

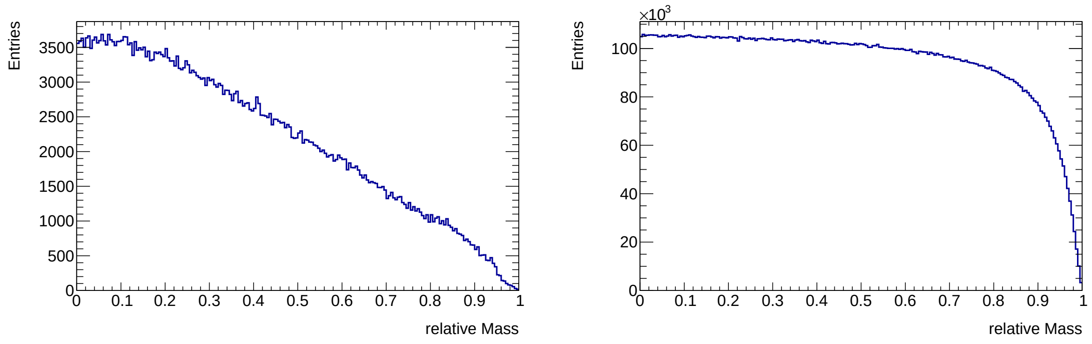

Figure 4.3: Histogram showing the distribution of relative mass values for cone pairs on the left and jet pairs on the right. Different shapes of the distributions lead to a possible background reduction when providing this variable to the network.

The results depicted in figure 4.3 show, that these assumptions appear to be correct. The distributions for cone and jet cases show different qualitative shapes leading to a possible separation into relevant and irrelevant jet pairs by the network. This is because the distribution for cone pairs falls linearly whereas the jet pair distribution stays mostly even with a sharp drop toward higher values, leading to relative mass values of relevant jet pairs being mostly small while the variable value is much closer to an even distribution for arbitrary jet pairs. As a result the desired background reduction is possible to achieve for a network using this variable and its effect on model performance shall be tested. However the plots also show that background reduction is hard to achieve due to the sheer amount of possible jet pairs, so useful variables to the network should be able to achieve an effect beyond simply reducing background. As a result, further variables should combine an ability of discriminating the datasets while possibly also separating important jet pairs from statistical background.

4.2.2 Angle Difference

This variable describes a distance of two points in the angle plane given by the relation

$$\Delta R = \sqrt{(\phi_1 - \phi_2)^2 + (\eta_1 - \eta_2)^2} \tag{4.2}$$

with azimuth ϕ and pseudorapidity η of the two jets forming the pair. To ensure ΔR is always the smallest possible distance, the difference $\phi_1 - \phi_2$ is chosen in a manner that the absolute value is smaller than 180 degrees. Now this *Angle Difference* variable can be interpreted as describing the closeness of two jets in the angle plane. Due to the significantly different masses of higgs and Z, this variable is expected to have at least slightly different distributions between the two datasets because of the differing kinematic conditions. Further, since a higgs decay happens into two particles of the same mass, this distribution is expected to have a characteristic peak for the most common configuration of these decay products, which should still be reflected by their jets. Since the b quark pair of a higgs or Z decay is always produced into the original direction the mother particle was traveling, the difference in pseudorapidity should be small for these pairs. This further introduces background reduction since in general for any possible jet pair this η difference is not small leading to a bigger Angle Difference value.

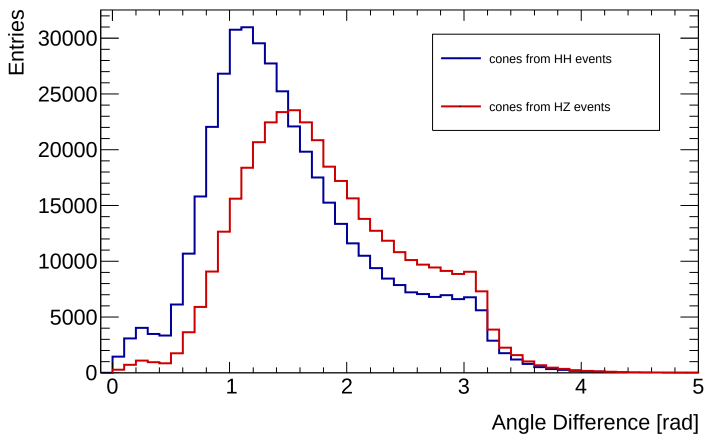

Figure 4.4: Histogram comparing the distribution of angle difference values for cone pairs from HH and HZ final states. While the general shape of the distribution is similar and there is a large amount of overlap, the peak values are distinct which might allow for a better dataset discrimination by the network.

Figure 4.4 shows a comparison of the distributions for this variable for cone pairs in the different datasets. The general shape of these distributions is similar and there is a large region of overlap, however the peak regions are slightly shifted. Because of this the angle difference value expected for most of the jet pairs belonging to each final state is also slightly different, which is expected to be beneficial for a neural network to differentiate between datasets. Further, the values are relatively small for most of the cone pairs also leading to the expected background reduction. Due to these observed properties this variable shall also be tested in the neural network model later on.

4.2.3 Angle Between Jets

Another way to make use of the aforementioned expected kinematic differences of higgs and Z decays into b quark pairs is by looking at the angle between the two produced jets. This angle can be calculated from 4-vector information of the two jets as follows

$$\theta = \arccos\left(\frac{j_1 \cdot j_2}{|j_1||j_2|}\right) \tag{4.3}$$

where j_1 and j_2 are the momentum 3-vectors of the two jets. This variable serves a similar purpose as the angle difference discussed above and the expectations are mostly the same, but it is calculated using different properties of the jet pair, making it a potentially worthwhile addition to the network. Since jets from b quark pairs formed as a result of a higgs or Z boson decay are roughly of same mass and boosted into the movement direction of the original particle, this angle is expected to be significantly smaller than 90 degrees for most cone pairs. Because of this a background reduction effect is also expected from this variable, since the angle between two random jets in an event should be roughly equally distributed.

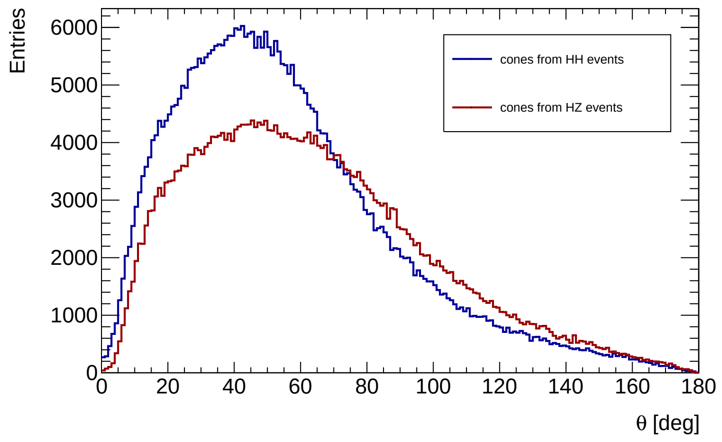

Figure 4.5: Comparison of the angle between cones for the two datasets. The different peak values here are not as prominent as for the angle difference, but this variable shall be tested nevertheless.

Figure 4.5 compares this angle for both datasets. In general the peaks of these distributions are very wide and therefore cover mostly the same regions, but again a very slight shift between the maxima is visible. This means, that the angle between a jet pair might also be useful for the network to help with its decision. On top of that this variable also shows the expected background reduction, potentially further helping the network. The main reason for adding this angle variable to the network was the hope it could teach new relations between variables to the network due to its similarity to angle difference while being calculated from different jet properties.

4.2.4 Transverse Momentum

In a particle physics context the *Transverse Momentum* p_T describes the part of the momentum perpendicular to the beam axis. Since momentum in this direction exists due to physical processes happening in the collision, this variable is of great importance when conducting experimental searches at colliders. In this case the expectation would be that the different nature of interactions leading to either a HH or a HZ final state leads to specific values for the transverse momentum, which are different in both cases. This would not only be the case for the jet pair but also for the two jets themselves compared to other jets in the event and jets from Z bosons. Transverse momentum describes the momentum located in the x-y-plane and is given by the expression

$$p_T = \sqrt{p_x^2 + p_y^2} \tag{4.4}$$

using the x and y components of the momentum. There is no expected difference from all possible jet pairs in the distribution of this variable, meaning that no significant background reduction is predicted.

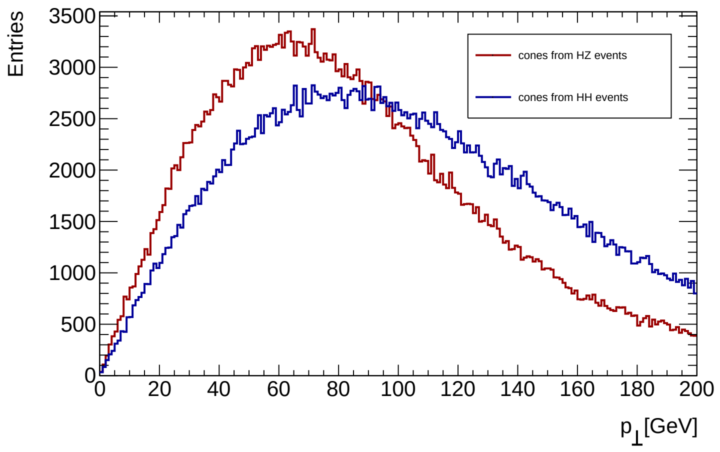

Figure 4.6: Comparison for the histograms showing transverse momentum of cone pairs for the different datasets. Similar to the previous variables a clear difference in the peak positions can be observed.

Figure 4.6 shows the results of studies of p_T in the datasets. The distributions have similar shapes with lots of overlap and a slight difference in peak regions. Due to the clearly visible different peak values this variable seems suited for distinguishing between the two datasets. As expected, the distribution for all possible jet pairs looks qualitatively similar with a slightly shifted peak value, making this variable unsuited for background reduction. Since discrimination between the datasets is more important than background reduction however, this variable shall still be tested in the neural network model.

4.2.5 Dataset Creation

The information gathered from this analysis step will be collected into a comma separated value (csv) file which can be imported into the neural network structure. In order to test different combination of variables and structures of the dataset, multiple different csv files were created for the experiments conducted in chapter 5. No further restrictions to the jets used from an event are made for now and all jets of each event will be collected into the csv files. For each jet a set of variables will be saved in the csv file, which will then be used to train a neural network. In the case of variables for pairs of jets, as described above, all possible jet pairs inside of an event are created. The corresponding variables are then calculated for all these pairs and saved in the file. More details on the exact shape of these datasets will be mentioned in the related experiment sections later.

4.3 Additional Results

The aforementioned variables are not the only ones investigated over the course of this analysis. The following section will show a selection of other attempts to find variables sensitive to HH and HZ final state discrimination. Most of these variables might still have general value when provided to the network, even if some of them are unable to provide a significant distinction of the final states on their own, the information they add to the data may still be valuable for the network. These variables were not tested in the neural network, but some could potentially be used for follow up analyses as they are sensitive to the differences in final states, at least in theory, and their information could be desired for an improved classification. Similarly searching for further variables with theoretically predicted differences in HH and HZ final states beyond the selection presented here, might prove useful for future studies.

Decay Angles

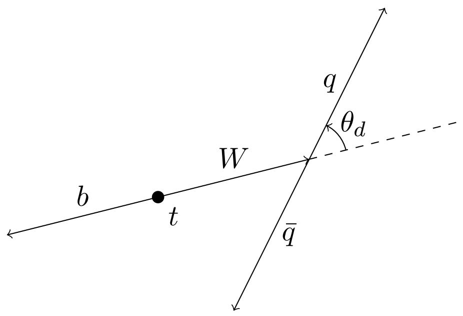

Figure 4.7: Visualization of the decay angle θ_d from the decay of a W boson in the top quark rest frame, produced from the decay of a top quark into W boson and b quark. This angle is expected to be characteristic for the given decay process allowing for background reduction and possibly even discrimination between datasets.

A produced top quark pair $t\bar{t}$ can mimic the final state of a HH or HZ boson pair via decays into a $WWb\bar{b}$ final state due to the most common top quark decay $t \to Wb$. To avoid wrongly identifying these produced final states, the spin properties of W and higgs bosons are used. W and Z bosons are spin 1 particles whereas the higgs is a scalar spin 0 particle. Because of this a W boson pair produced from a decay has a kinematic restriction due to spin conservation, which will affect its decay angle based on its origin from a HH or tt pair. The reason for this being, that a HH pair has no spin correlation opposed to the top quark pair. This difference should be visible in the decay angle variable which might therefore help a neural network decide which particles to consider for its classification.
The setup considered is shown in figure 4.7, from this one can derive the decay angle θ_d, from the kinematic relations with respect to the angle given by

$$\frac{E_1}{E_2} = \frac{1 - \beta_d \cos(\theta_d)}{1 + \beta_d \cos(\theta_d)} \tag{4.5}$$

with relativistic beta factor β_d of the decaying W boson and energies E_1 for the quark and E_2 for the anti quark in figure 4.7. This then makes it possible to determine θ_d via

$$\cos(\theta_d) = \frac{\frac{E_1}{E_2} - 1}{-\beta_d(\frac{E_1}{E_2} + 1)} \tag{4.6}$$

This was implemented for all used cones in the hope of observing a significant peak or other qualitative difference in the distributions for each particle type or between the datasets.
Figure 4.8 shows, that the distributions of the decay angle for the two datasets are pretty much identical. This means the desired effect was not achieved with this variable. However the described influence of the particle decay kinematics on the variable value given the different datasets might still be reflected in a more complicated variable relations inside the dataset, in which case this decay angle variable would still be helpful. It is hard to tell whether this is really the case, without further testing or revisiting the assumptions and approximations done in calculations. For this reason the decay angle variable was not used in the neural network tests.

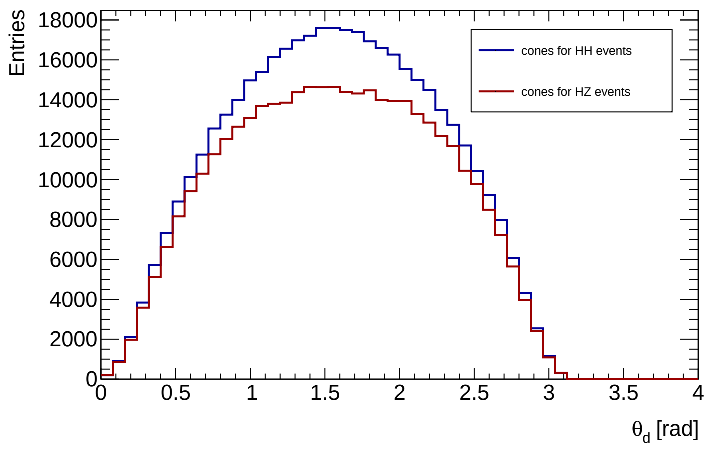

Figure 4.8: Comparison of histograms showing the decay angle variable for cone pairs for the two different dataset. No qualitative difference is noticeable when comparing these two distributions.

Relative Energy

Similar to the idea of using relative mass for a reduction of background, another possible relative variable for jet pairs capable of reducing background effects was defined. This variable is the relative energy, which is given by

$$E_{rel} = \frac{|E_1 - E_2|}{E_1 + E_2} \tag{4.7}$$

The reasoning behind this variable comes from the observation that higgs decays happen mostly into a pair of the same particle type like a b quark or W boson pair. Because of this, energy is expected to be distributed relatively evenly leading to jets resulting from the higgs decay still having relatively similar energy values. This is also not expected to be the case for an arbitrary pair of jets where values are again expected to be more evenly distributed. To test this hypothesis the relative energy variable was calculated and plotted for all cone pairs in each of the events and then compared with the distribution of relative energy for all possible jet pairs present in the event.
The resulting histograms are shown in figure 4.9. It can be seen that the distributions for cone and jet pairs have the same qualitative shape. As a consequence, the relative energy variable seems to offer no help in background reduction. This also means that the premise

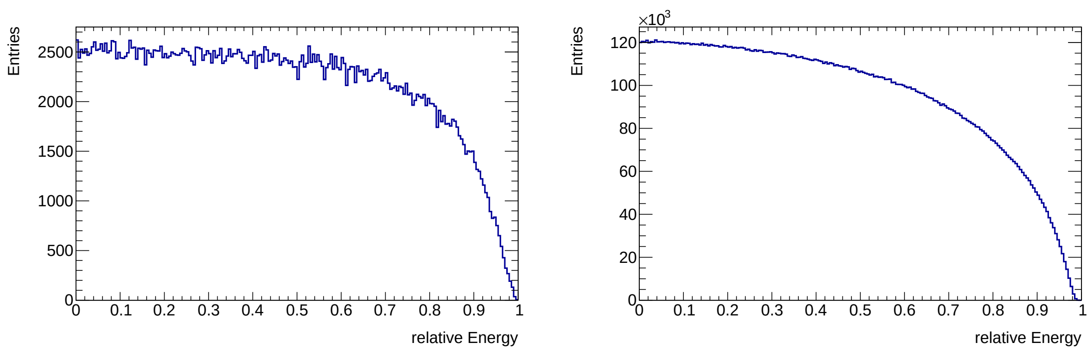

Figure 4.9: Histograms comparing the relative energy variable for pairs of cones on the left and jets on the right. The distribution shapes are very similar which is why no background reduction effect is achievable using this variable.

of jet pairs from higgs and Z boson decays producing jets with similar energies does not appear to be true for these processes. As a result this variable is seen as useless for the neural network classification task and shall therefore not be tested in the network later on.

Higgs momentum

Another way the kinematic differences between processes producing HH and HZ final states respectively could be visible, would be the total momentum of the decaying bosonic particle. The idea being that a different momentum value would be favored for higgs bosons of the different final states due to the available energy not being the same because of an approximate difference in higgs and Z bosons masses of 35 GeV. This might be enough to lead to a variance of peak positions in the momentum distributions between the data sets, which would be another helpful distinction to be used in the neural network analysis. This difference should then also be visible in the same momentum distributions for cone pairs since these pairs roughly represent the properties of a decayed higgs particle. This distribution shape is expected to be the same for all jet pairs in the event, meaning that there is no possible background reduction achievable using this variable. To test the effectiveness of this variable for dataset discrimination, distributions were plotted for cone pairs from higgs decays in both datasets.

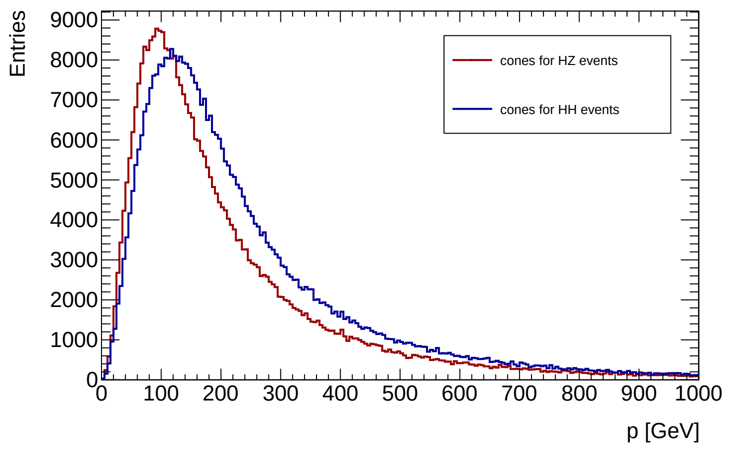

Figure 4.10: Histogram showing the total momentum for higgs particles reconstructed via cone pairs in the two different datasets. A clear difference in peak values is visible, meaning that this criterion is very promising for helping the neural network with its classification task.

The result of comparing this variable for the different datasets is shown in figure 4.10. For both datasets the shape of the distribution is the same with a peak at relatively low values. The position of those peaks are significantly different however, verifying the expectation. Due to this fact the total higgs momentum is another variable helping to make distinctions between the two datasets possible and is therefore most likely helpful to the network. As expected, a background reduction is not possible to achieve using this variable since the qualitative shape of the distribution of cone and jet pairs are pretty much identical. Despite its promising results this variable was not directly tested in the neural networks due to time constraints, it was however included in the final experiment discussed in section 5.4.2, due to the dataset set up. Furthermore future efforts might benefit from this result as well.

Higgs angle distributions

Goal of this analysis was to establish whether or not the decaying higgs bosons from the different processes prefer certain directions in the coordinate system. This was tested by looking at the pseudorapidity and azimuth angle distributions of recreated higgs particles using the Monte Carlo information about b quarks from higgs decays. The expected result was mainly concerning the polar angle since physics should be symmetrical in ϕ because of the experiment's symmetry with respect to that variable, which would lead to an equal distribution along that axis. The same variable analysis was done for constituent b quarks to see if they have a preferred decay direction as well. It is expected that no such preferred direction in both variables exists for both the higgs boson and its decay products.

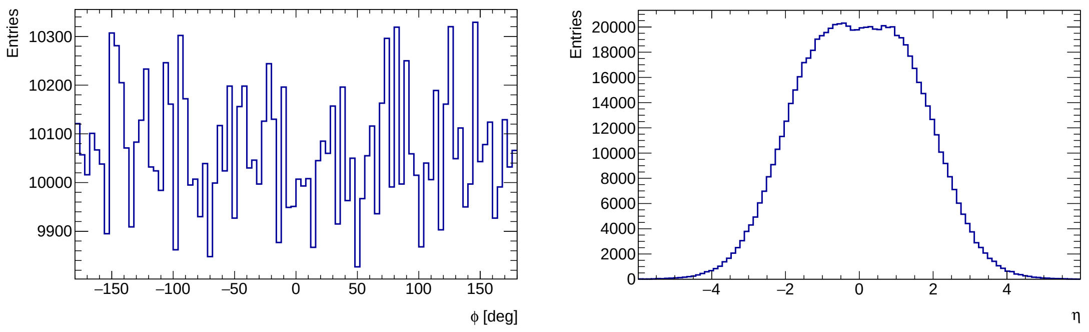

Figure 4.11: Plots showing the distributions of the azimuth angle on the left and pseudorapidity on the right for higgs bosons in the event. These plots look as expected and show no signs of any direction being favored for higgs decays

The results show that, as expected, there is an even distribution for the azimuth angle and also for the pseudorapidity no significant differences from the expected distributions can be observed as can be seen in figure 4.11. As a conclusion there seems to be no preferred direction of higgs production or decays, meaning the absolute angle values are not useful for the network. As a result of these observation, the focus for angle related variables was shifted towards higgs decay products and their angle relations as discussed in section 4.2 above.

Chapter 5

Neural Network Analysis

To improve the neural network classification of HH and HZ final states, which was conducted in a preceding bachelor thesis [22], its original neural network model structure shall be improved. In order to achieve this, alternative designs for the neural network were tested to construct a final model, able to include the pair variables introduced in the previous chapter and producing an overall better result than the original model. This chapter will describe the initial neural network used as a base model and all tested modifications leading to its final structure. Further, the dataset used is expanded by new potentially beneficial variables from the previous chapter and their impact on the model performance is compared. In the end it is expected that these changes to the original neural network model will lead to a significant improvement of its classification abilities.

5.1 Introduction

The neural network analysis was implemented in Python 3.7 using a plugin called PyTorch [28], which provides tools for neural network training. Starting from an initial model, both the network and dataset structure shall be changed in multiple steps to increase performance. These changes will include various expansions and restructurings of the initial model and the addition of more different input information into the dataset. This section will introduce the initial model, its problems and the methods used in the following analysis to judge classification performance. Since the overall network structure will be made up of multiple smaller neural networks, in the following the term *model* will be used when talking about the structure as a whole and the term *network* when talking about the individual model parts.

5.1.1 Initial Model

This section provides an overview over the basic structure of the initial model. It will act as a base for further expansions and experiments, but the core ideas and principles of the model will remain unchanged. The goal will be to make major improvements to this model via slight changes, which are tested individually and combined in the very end to form a final model, which should lead to a significant boost of model performance. Furthermore the basic set up of the datasets is explained, which will also be subject to change when trying to improve the data processing abilities of the model by adding further jet information.
There are two different datasets, one for each final state considered, created with the corresponding Monte Carlo file, but the structure of both datasets is identical. For each jet a total of six variables is saved in these files, the jet energy, its momentum components in x,y

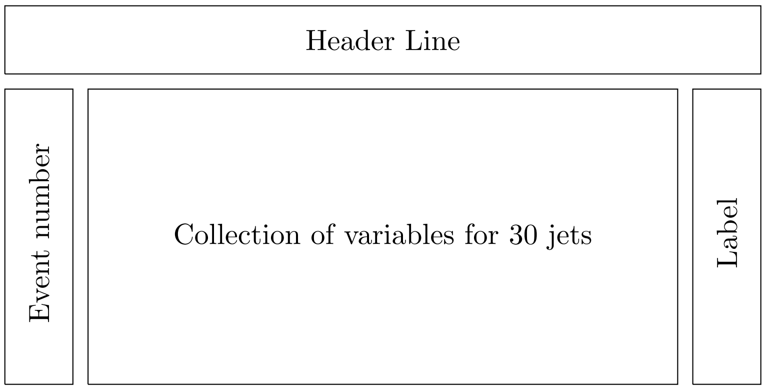

Figure 5.1: Schematic view of the structure used for the csv file. The contents of each line represent the information of one event. Variables for each of the 30 jets are always arranged in the same order to make processing easier.

and z direction, the number of particles that make up the jet and a boolean value stating whether or not the jet originated from a b quark. All jets of an event are listed one after another in one line of the dataset with a total of 30 jets per event, where the remaining jets not present in the actual event are filled up with 0 values. At the very end there is an extra column for each event, used for labeling the datasets with target values. These are 1 for events of the signal HH final state events and a value of 0 for the background HZ process. When importing the datasets into the model, further processing steps are conducted. First the signal and background datasets are combined using 60 thousand events each, leading to a total number of 120 thousand data points used by the model. Data is imported from the csv file into a pandas data frame used to do the following processing steps. Each variable is divided by the rough order of magnitude its largest values can have (e.g. energy is divided by 1000, number of particles by 100 etc.), which results in all variable values being roughly restricted to the range from 0 to 1. Then for each row in the dataset jets are created and collected into a $n \times m$ matrix saved as a PyTorch tensor object accessible to the network functions, where n is the number of jets in the event and m the number of jet variables in the datasets. Because of the way the model is set up a minimum value for n of five is used, if there are less than five jets in an event the remaining lines are filled up with zero values for each variable. This matrix is then used as an input for the model. A schematic representation of the dataset structure is shown in figure 5.1.

Figure 5.2 visualizes the general model and neural network structure. The model consists of two consecutive networks, where the first one does preprocessing of all jets individually and the second network receives the results as inputs to do the final event classification. The first network consists of two hidden layers and a skip connection from the input to the second hidden layer, where all network layers are fully connected liner layers. The skip connection uses the output value of the input layer and adds it to the value of the second hidden layer before applying its activation function. Network layers use leaky ReLU as an activation function with a slope of 0.1 and the network's input and output layers have a size of five with hidden layers containing ten neurons. The produced result is therefore a $n \times 5$ matrix, which then gets reduced to a 5×5 matrix by selecting the five biggest values for each variable among all jets. All of these 25 variables are then passed to the second network all at once, which has a similar structure to the first one. Differences are the new number of input neurons at 25, hidden layer size of 15 and an output layer size of two. The final layer also uses a *softmax* activation function here, which is given by the equation

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{5.1}$$

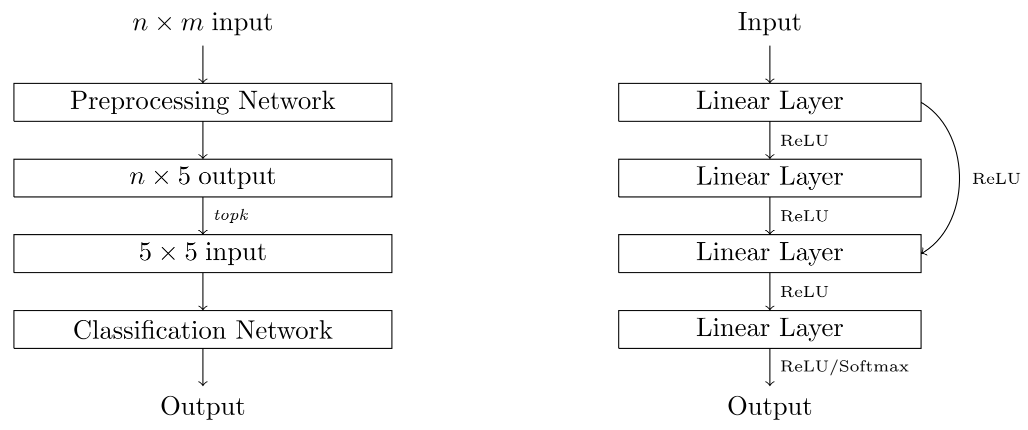

Figure 5.2: Overview over the basic model structure on the left and general structure of both neural networks. The topk function is used to select the five biggest values per variable to create an input for the classification network. Both neural networks have a skip connection, combining the output of their input and final hidden layers.

and ensures the sum of the two output neuron values is always 1, this makes a probabilistic interpretation of the classification result possible. Because of this setup, the first component of the two final outputs is compared to the labels, where the threshold chosen is 0.5, meaning that all values larger than that are classified as signal and smaller values are labeled as background. To initialize the weights He initialization is used, which randomly distributes weight values according to a Gaussian distribution with zero mean and a standard deviation of $\sqrt{2/n_l}$, where n_l is the number of the current layer in the network whose weights should be initialized, and with a bias of 0. This is optimized for usage of ReLU activation functions and makes sure weights are initialized in a way that allows for proper scaling of input signals, which helps to create converging deep networks [29]. As a result, this weight initialization technique is well suited to be used with the current neural network setup.

The loss (= error) function chosen for the network is called *cross entropy*, which is really useful for the task of labeling multiple classes and is implemented in PyTorch like

$$\ell(x, y) = -\sum_{c=1}^{M} w_c \, log \left(\frac{\exp(x_c)}{\sum_{i=1}^{M} \exp(x_i)} \right) y_c \tag{5.2}$$

for a total of M different classes with target value y_c and weight w_c from an output x_c. This loss function is minimized via the ADAM optimization algorithm. The loss and optimizer functions will remain unchanged over all of the experiments, but changes within this structure might be needed as well since PyTorch uses a softmax to calculate the output probabilities which is already done in the final layer of the classification network in the initial and all later models.

The model is run using a batch size of 10 thousand, which is the number of data samples an epoch uses to calculate its weight update. Further, network runs will be conducted over one thousand epochs with a learning rate of 0.01. These values will be kept the same over all experiments so long term performance, such as network convergence and improvement over time can be evaluated. The theoretical differences for different batch sizes and possible advantages of specific values are explained in section 5.2.5, but in practice no real benefit of changing this parameter were found.

Figure 5.3 shows plots for the performance of the initial model. In a test environment the model achieves an accuracy of roughly 63.5%, which is not really sufficient to reliably select states for physical analysis, since the statistical errors are quite high and a clean selection is needed for these rare processes to ensure a good analysis result. Further the network shows

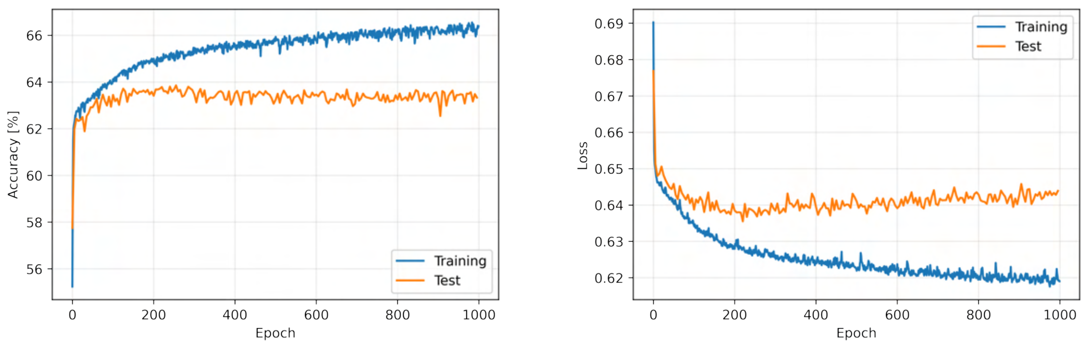

Figure 5.3: Accuracy and loss plots for the initial model. Strong overfitting effects can be observed from epoch 100 onward, which is one of the biggest weaknesses of the initial model. The highest accuracy achieved in testing was roughly 63.5 %.

clear signs of strong overfitting, which has to be addressed in order to enable significant improvements of its result. In the following sections the experiments made to address these two main issues with the initial model are described. In section 5.3 the efforts done to reduce overfitting are discussed while section 5.2 deals with attempts to improve the model structure and dataset information provided to the networks inside the model.
Because all input variable values are normalized to a value range between 0 and 1 and further the boolean variable saying whether a jet originated from a b quark or not can only take absolute values of either 0 or 1, it is expected that the impact of this variable on the final result is rather large. To avoid this variable to potentially overshadow effects caused by introducing new structure and variables it will be ignored for the further analysis. Rerunning the model while omitting this variable leads to a slightly worse performance of roughly 60%, which will be used as a comparison value for future models.
Furthermore these unambiguous b jet classifications are unrealistic and a network trained with them has the risk of a diminished performance on real data, due to adding a variable that essentially uses Monte Carlo information unavailable in actual experiments. A more accurate way to include such a variable in future studies would be a simulation of b tagging for jets in the dataset.

5.1.2 Analysis Methods

A classification of values into two possible classes, called positive and negative here, can result in four different outcomes.

1. An item belonging into the positive class is correctly classified as positive. This case is called *True Positive* or TP.

2. An item belonging to the negative class is correctly classified as negative. This case is called *True Negative* or TN.

3. An item belonging to the negative class is incorrectly classified as positive. This case is called *False Positive* or FP.

4. An item belonging to the positive class is incorrectly classified as negative. This case is called *False Negative* or FN.

All results of a classification task can be identified as one of the four above cases, which allows to define variables that can be used to judge the performance of a classification result. One of these is the *accuracy* given by

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

and describes the fraction of correctly classified results. Accuracy will be the main variable besides the loss values used to evaluate model performance. Under normal circumstances this characteristic variable is expected to increase quickly in the beginning of training and then converge toward a final value. It is however not suited to observe overfitting since the increasing error does not necessarily lead to a shrinking accuracy it might only fluctuate or stagnate, which makes overfitting harder to spot when only focusing on accuracy.
Loss is the error calculated by the model in order to make weight updates and create a learning effect. As such, it is expected to start at an arbitrary value and then diminish in value as the model learns. After a while it should converge when the maximum learning capacity of the model is reached. This variable will be the main factor in comparing performance of different network models, where a model able to achieve smaller error values is seen as superior. On top of that, overfitting is relatively easy to spot when comparing loss in training and test environments. The absolute loss values however are not really suited for quantitative evaluation of the model output which is why accuracy is used as well.
As explained in section 3.2.4 testing a model outside of its training environment is important to assure its results are useful. For this reason the datasets used are split into Training and Test data where a fifth of the data is used for testing and the rest for training of the model. Because a good training performance is practically useless if a model fails in the test stage, the testing values of variables will be considered for model evaluation and since the best possible state of the model can be extracted and saved, the peak values will be considered.
Due to time constraints the models could not be run multiple times to get rid of statistical fluctuations present in the results, because of the statistical nature of neural networks. As an attempt to make up for this, only sufficiently large differences in performance will be seen as a clear sign of a significant qualitative difference in classification results produced by the models.
Other performance variables are also recorded when running the models, but these are not the primary source for the final judgment. These variables are precision, recall and specificity. Their definitions are given below.

$$precision = \frac{TP}{TP + FP} \tag{5.4}$$

$$recall = \frac{TP}{TP + FN} \tag{5.5}$$

$$specificity = \frac{TN}{TN + FP} \tag{5.6}$$

The differences between these variables are small but each of them offers a unique perspective on the qualities of obtained results. Precision shows the percentage of samples classified as positive, that are actually labeled correctly. On the other hand recall measures the fraction of positive events classified into the correct class while specificity does the same for negative events. Depending on the desired result, for instance a clean background suppression, this introduces different constraints on the values these quantifiers should have for a model to be considered adequate. This also means that a complete evaluation would use all of these variables and then decides whether or not the model fulfills the demanded requirements. To keep things relatively simple, this analysis only considers loss and accuracy as its main performance metrics with the goal of a high accuracy while minimizing loss as far as possible. The three quantities defined above will then only be used to further evaluate the final results, by comparing their values at the point of best model performance, meaning lowest loss, to get more insight into the quality of obtained final results.

The experiments will be structured as follows. In general models are compared to a corresponding baseline model performance, based on their achieved accuracy and loss metrics. The baseline model in each experiment is a simple common starting point all models tested had, while making small adjustments or changes, for example different parameter values. After each comparison step, introducing a new feature or structure, that proved to be beneficial to the model, into the final model can be evaluated. There are three major experimental parts with different areas of change for the model. First were structural changes to the networks, then afterward regularization efforts were tested and finally, after constructing a final model based on the previous results, different datasets for the variables found in the preliminary analysis were tested. The experiment results of these efforts will be discussed in more detail over the next sections.

5.2 Restructuring

The first effort done to improve performance of the initial model was to do extensive restructuring of its networks with the goal of improving performance and potentially reducing overfitting effects. These changes also made it possible to add the new variables which are based on jet pairs. This made additional infrastructure in the model necessary to account for these pairs. Experiments in this section only consider adding one of the pair variables, the angle difference. Further the output size of the first network was changed to six in order to match its input size. A large variety of changes to the model structure were tested and will be described within this section.

5.2.1 Adding Pair Variables

In order to test different implementations capable of providing pair variables to the model, values of the angle difference variable were added to the dataset for all possible jet pairs in an event as a block of values between the result column and the rest of the jet variables for each line. Because of the qualitative difference between jet pair and other jet variables, the preprocessing of them should happen separately.
The general idea of a preprocessing network and a final classification network shall be kept even after the structure change, so the pair variable will be processed in its own network and afterwards the results of the two initial networks will be combined into an input for the classification network. The general structure of the pair variable processing network will be the same as of the jet variables, with an equal amount of hidden layers and layer neurons, while still including a skip connection. Similar to other inputs the pair variables will also be normalized to values between 0 and 1 via division by 10. For the realization of a pair variable processing network two different approaches were considered

1. All nonzero pair variable values are given to the network at once, without using a certain jet pair twice. This includes padding to reach a minimum number of five inputs for the network, then select the five biggest output values via *topk* function and add them to the final network input.

2. Construct a total of n variable-jets, where n is the number of nonzero jets in the event. The i-th variable of the j-th jet would be the pair variable value for the i-j jet pair. Input and output size of the pair variable network are then n and from the output the five biggest values per column are selected as input for the final net, similar to the original preprocessing network. The number of variable-jets is padded to a minimum number of $n = 5$ using 0 values.

Both of these approaches require variable input sizes of the additional preprocessing network which are implemented such that the output size will always match the input size. When varying the input size it is important for the network performance that the input variables in the same position always have the same meaning. This is the case for the second option but not for the first, so the expectation would be that the second structure performs better.

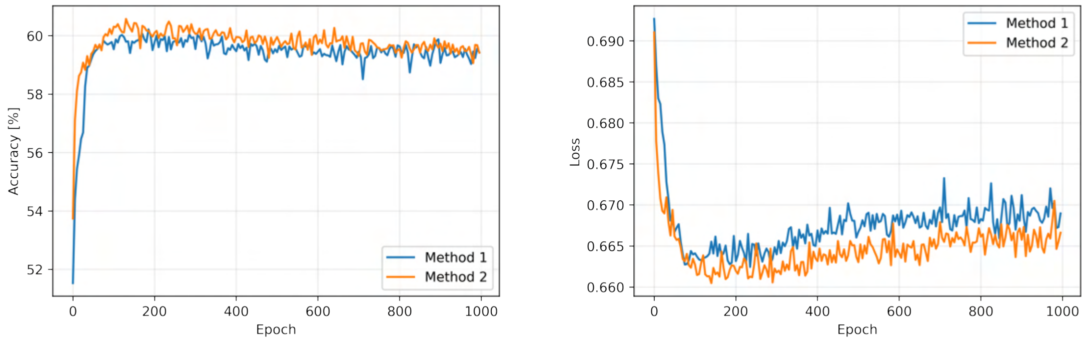

Figure 5.4: Test performance comparison of the two methods for implementing the pair variables. As expected the second method outperforms the first one, the differences are small however.

Comparing these two realizations of a pair variable preprocessing network yields the results visualized in figure 5.4. This shows that the second approach for adding these variables seems to perform better and is therefore chosen as the way of implementing pair variables for all future models. The addition of an entire additional network introduced more parameters to the model, which seems to have counteracted the regularization effect from the added information to the network by introducing the new variables. As a result, overfitting is still clearly visible in these performance plots and has to be solved differently.
Future experiments will then aim to improve this model, capable of adding pair variables, further by testing alternatives to the initial model structure. In addition, as an attempt to show the network more connections between variables and adding more information at the same time, relevant jet variables such as the angle ϕ and pseudorapidity η in the case of angle difference, are added to the datasets for the pair variables from now on. These are normally related to the pair variable by either being the same variable for jets or being used for calculating the pair variable. This step seems to have a slight positive effect on training while slightly affecting the test runs negatively, which is expected to lead to an overall improvement of performance as soon as overfitting is reduced.

5.2.2 Input Normalization

So far the normalization of inputs has been rather arbitrary by simply picking a power of 10 that is the closest to the order of magnitude of the given variable and dividing its value by that number. This procedure already helps the network by making sure all variables are of a roughly equal size while keeping the relative sizes of variable values intact, but it is not optimal. This is the case, because outlier values larger than 1 are still possible and variables are normalized to different ranges of the 0 to 1 interval. In an attempt to improve these sub-optimal variable normalizations, several other possible normalization techniques were tested.

Statistical Normalization

These types of normalization are a transformation of the input data, fulfilling specific statistical properties for the transformed data. Two of these possible transformations were tested for normalizing the input data, in order to see if they have a positive impact on model performance. The first method is *centering* of data achieved via the transformation

$$\hat{x} = x - \mathbb{E}(x) \tag{5.7}$$

using the statistical mean of all data points $\mathbb{E}(x)$, which ensures the transformed data will have a mean of zero by subtracting the mean of the input data from each data point. The other tested method is *standardizing* of the data which is done via the following transformation rule

$$\hat{x} = \frac{x - \mathbb{E}(x)}{\sigma} \tag{5.8}$$

Here centered data is further scaled using the square root of its variance σ^2, which leads to the transformed data having zero mean and a unit variance [30].
Since padded zero values for variables should not end up in the used data samples in the majority of cases, they are ignored when calculating statistical properties of the input data. Further, two types of tests were conducted, one where each variable is transformed individually and another one, where the entire input data is transformed together. These different scenarios are then compared to the initially used normalization for both centering and standardizing of data.

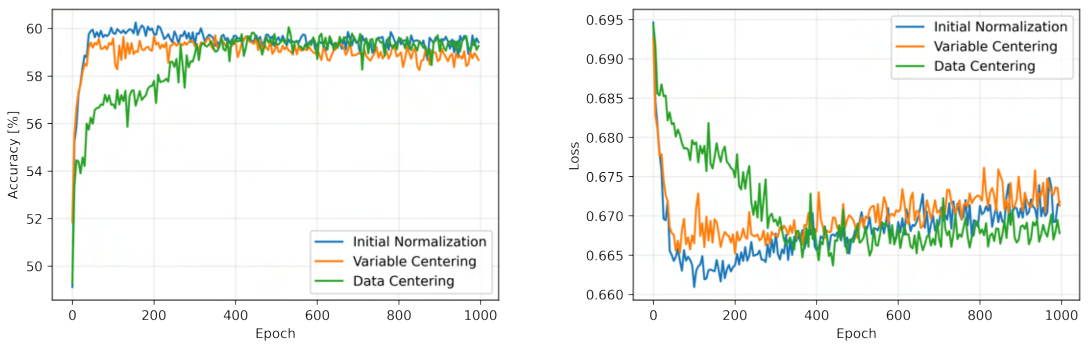

Figure 5.5: Comparing the effect of centered input data on model test performance. No clear improvement is visible, the initial normalization procedure shows the best performance.

Figure 5.5 shows the results of testing centered data, whereas figure 5.6 illustrates the same comparisons for standardized inputs. In general both approaches perform worse than the initial method used for normalization, meaning that the specific statistical properties of the transformed input data do not help the model learn. Because of this another approach for a new normalization procedure needs to be found.

Max Value Normalization

The idea of this normalization technique is pretty much the same as with the initial normalization procedure. Variables shall be normalized such that their values are of similar size, while still respecting outliers and statistical distribution of values. This value range will be from -1 to 1 for angle and momentum variables and from 0 to 1 for all others. To achieve this, all variables are divided by the biggest absolute value it can take on within the entire dataset. This change is expected to further enhance the performance boosting effect achieved

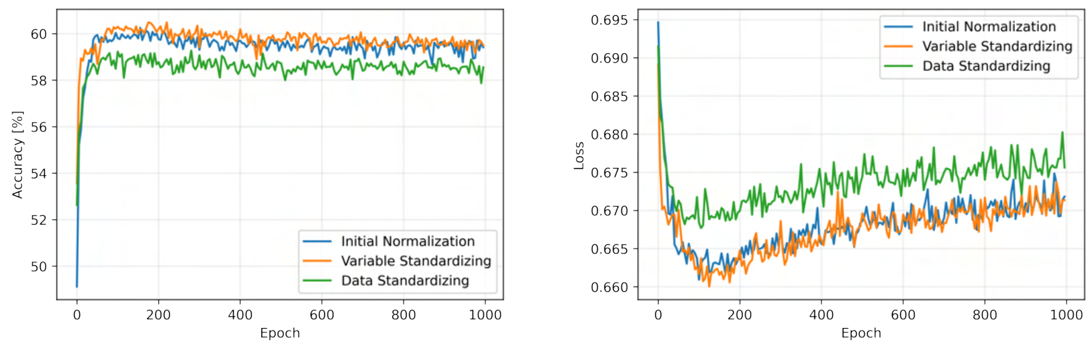

Figure 5.6: Comparison of model test performance for the initial normalization and standardized input data. The model does not seem to benefit from standardized input since there is no significant difference in performance visible.

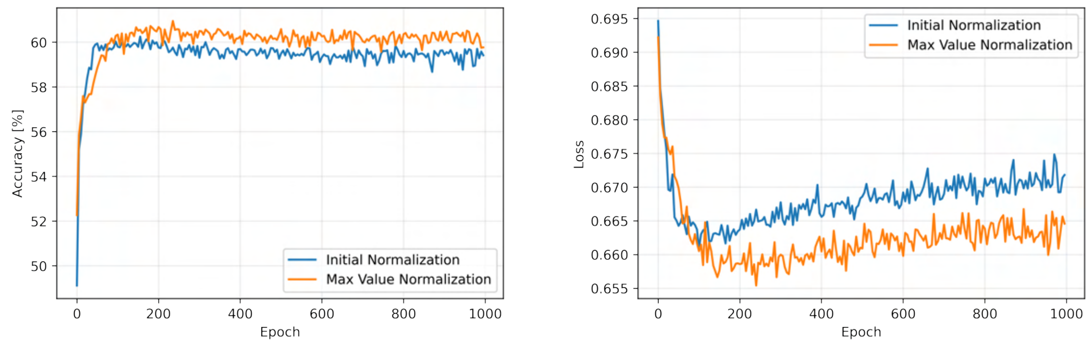

Figure 5.7: Comparison of test performance between initial and max value normalization. The results show a clear improvement over the old method by using max value normalization.

from the initial normalization by making sure all desired properties are always present. Comparing this approach with the old way of normalizing the input data yields figure 5.7. A clear difference in performance is visible using this new normalization technique, meaning the model benefits from this data preprocessing step. Due to the success of this normalization procedure, input data of the final model shall be normalized using their respective maximum absolute value.

5.2.3 Layers

For this restructuring step different design philosophies with respect to the layer structure were considered. These are *residual* networks where layers can additionally be connected via skip connections similar to the initial model, *deep* networks which are classified by a large number of hidden layers and *convolutional* networks using a special type of layer in combination to the simple linear layers used so far. There is no clear separation between these different network structure types and elements can be freely combined which was done in order to find a new overall architecture for the network model, which should outperform the initial model. For the purpose of this investigation a slightly altered model was used as a comparison. This model differs from the initial model in the number of hidden layers and the lack of skip connections, all networks have three instead of two hidden layers for the comparison. Reason for this change is to compare each of the different structures with a common base model lacking all features to be compared.

Deep networks

For the purpose of this comparison a deep network is defined as having twice the amount of total layers compared to the baseline model, which has three hidden layers, meaning the total number of layers is five, leading to eight hidden layers present in a deep network. Further, performance of various models where different networks of the model are deep while others remain unchanged are tested and compared. The three experiments conducted concern models where either the preprocessing or classification network are deep and a final test where all networks are deep. The number of neurons for all additional hidden layers will stay the same as in the initial model for their respective network.

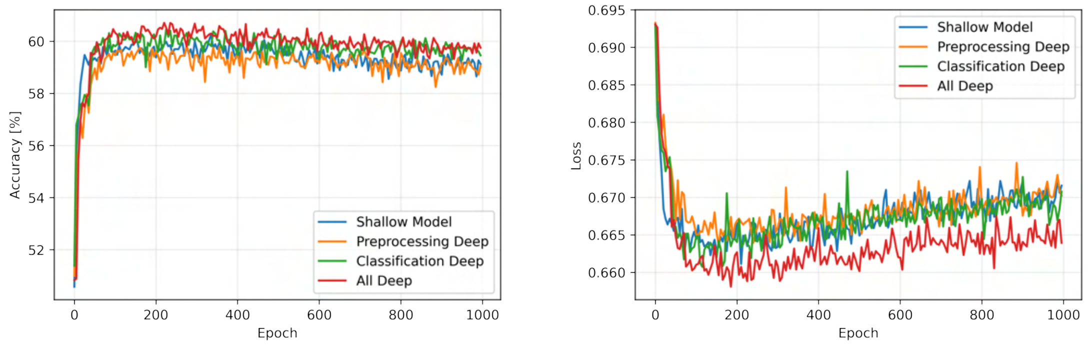

Figure 5.8: Comparison of the test performance of various deep network models tested. The results suggest that making the classification network deeper is beneficial for the overall model performance.

The results visualized in figure 5.8 show that making all networks deeper improves test performance. However, the computational resources needed for this change are quite large making it unsuited for further testing within this framework. Despite the added model complexity and capacity there are no significant increases in overfitting effects observed however, meaning this change to the model structure is possible without making the problem worse. Due to these results, adding hidden layers to the classification network is considered for usage in the final model as a compromise between improving the model and keeping computational requirements as low as possible, since a small improvement can be seen for this change to the model structure.

Residual Networks

Residual networks commonly use skip connection as illustrated in figure 5.9 and present in the initial network. These kinds of structures are common in large deep models used for image recognition and have proven to be really successful in aiding their training. They were introduced to fight a saturation effect observed for training loss when stacking a significant amount of layers, which was only related to model depth regardless of overfitting. Due to the elimination of this saturation a residual network structures enables an even deeper model while benefiting training performance [31]. The way skip connections are implemented for the following tests is different from the initial model and closer to the implementation used in residual networks for image recognition. The skip connection will connect the network input to the output layer in preprocessing networks and the final hidden layer in the classification network. Afterwards the input will then be added to the respective layer output, before applying the activation function to produce a network result. Similar to the deep network experiments, various model versions in which different networks have these residual skip

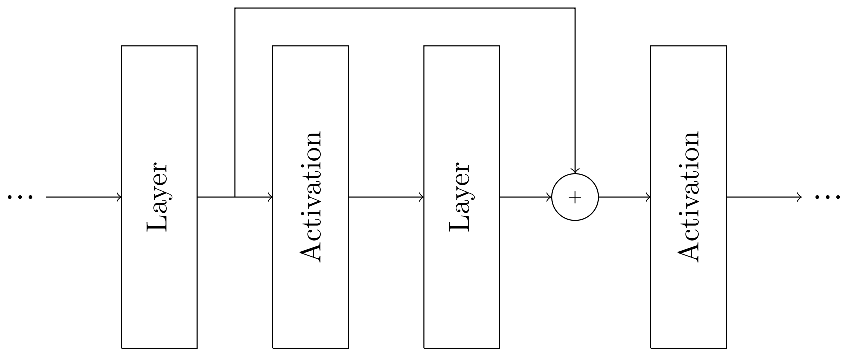

Figure 5.9: Schematic representation of a skip connection as used in the initial model. In a residual network often the input is added to a later layer output instead. Skip connections between layers have proven to be useful in deep networks for image recognition.

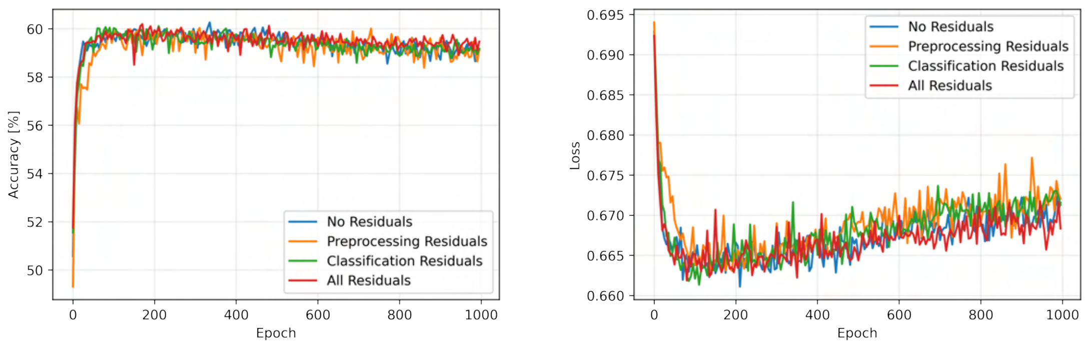

Figure 5.10: Plots for the test performance of the different tested residual model structures. In general the models perform equally well and improvements over the comparison model are small.

connections are tested and compared.
These tests result in the plots visualized in figure 5.10, which show no major differences between the different experiments. Further the improvements compared to the comparison model are minor and mostly only due to slightly less overfitting occurring. In addition only one of the tested models is able to achieve a slight improvement in training accuracy. Because of this skip connections seem to offer only a small value to the model and will be disregarded for future model structure changes.

Convolutional Networks

These types of networks use special types of layers called *convolution layer* typically followed up by a few fully connected linear layers to make a classification in the end. They are commonly used in image recognition tasks, due to their grid based data processing. The general principle behind a convolution layer is depicted in figure 5.11, together with a so called *pooling layer* it is often combined with in practice. Advantages of these networks include a reduction of parameters useful for efficient neural network training involving large data inputs and therefore also lessening the impact of overfitting due to a smaller model complexity. In a convolutional layer the *kernel* scans through the input and creates a different output value for each area in the data, reducing the dimensionality of the original input. Further, a kernel can learn and recognize patterns in its input, which is a helpful property for learning of the model as a whole. Pooling layers then reduce the dimension of its input even further by

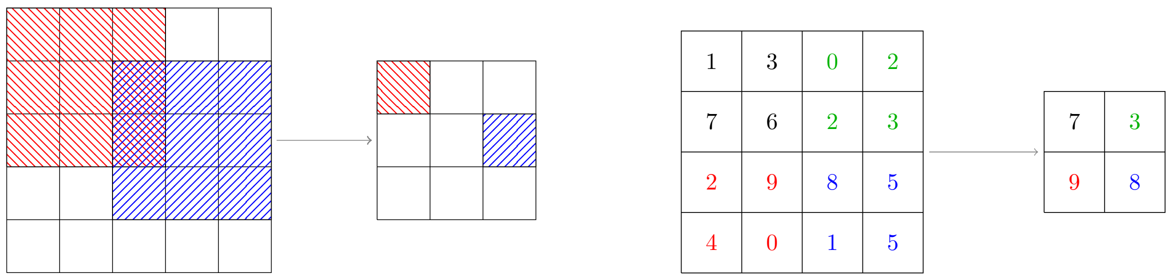

Figure 5.11: Schematic illustration of the two main components in convolutional networks. A two dimensional convolution layer using a 3 × 3 kernel is shown on the left and a type of pooling layer called max pool, which selects the biggest input value in a 2 × 2 region on the right. While kernels in convolutional layers may overlap, using certain input values multiple times when calculating outputs, the same behavior is normally avoided for pooling layers.

selecting a part of its input and compressing it into a single number via averaging its inputs or selecting the maximum value for instance [32].
For testing, convolutional layers were only used within the classification network due to their strong reduction of data dimensions. Two different setups were considered only distinguished by the amount of linear layers following the convolution. The first experiment only used a shallow version of the classification using a single hidden linear layer before the output layer, whereas the second version adds six hidden linear layers, making the classification network deep. The convolution layers used are one dimensional as provided by PyTorch and each of them is followed by a max pooling layer. Used parameters for all of these layers are summarized in table 5.1. This added structure produces a 30 variable input for the linear layers, which use 15 neurons as before and reduce the output to two dimensions in the final layer using a softmax activation.

	1st Convolution	2nd Convolution	3rd Convolution	Maxpool
In-Channels	1	10	15	-
Out-Channels	10	15	5	-
Kernel Size	1	3	3	2
Stride	1	1	1	2

Table 5.1: Parameters chosen for all layers used in the convolution network tests.

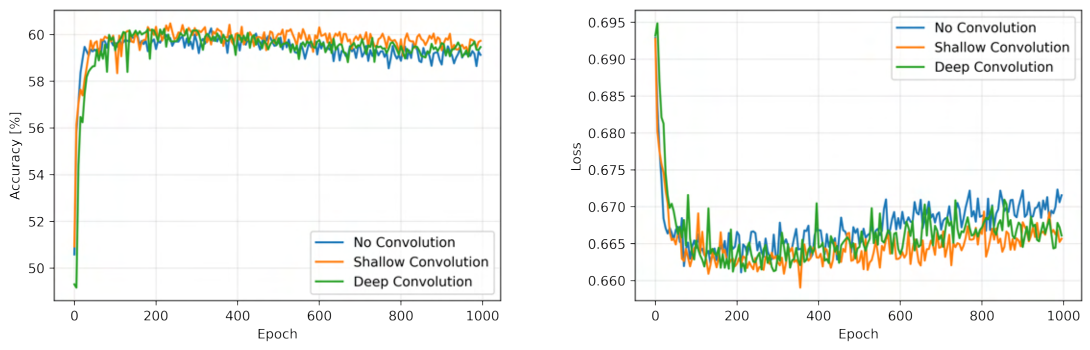

Figure 5.12: Test performance of various models using different structures implementing convolution layers. Usage of these structures seems to have a positive impact on the model, improving its performance slightly.

The results of these tests are shown in figure 5.12 and compared to a model which is not using convolutional layers. It can be seen that both convolution model variants perform well in comparison, but they cannot reach the level of improvement shown by the simpler fully connected deep networks discussed above. As a result of this, using convolutional layers in the final model was not considered despite their positive effect on the model performance.

5.2.4 Activation Functions

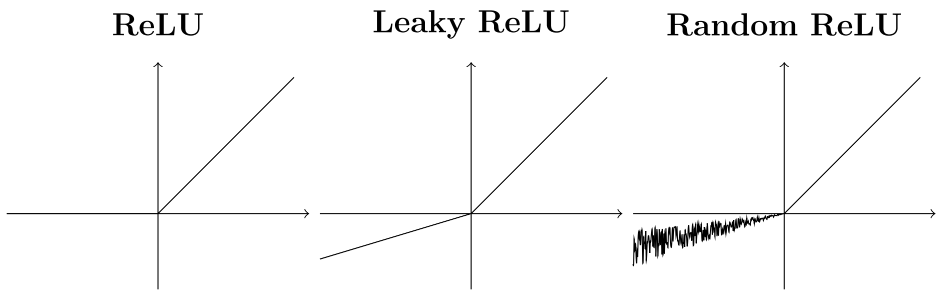

Figure 5.13: Comparison of a standard ReLU (left), a leaky ReLU (middle) and random ReLU (right) activation functions.

Compared to historically used activation functions such as the sigmoid function introduced in section 3.2.1, for more modern deep networks the use of non-saturated activation functions has proven to be superior, since they solve the problem of exploding and vanishing gradients while also improving the speed of convergence. The model uses a function of this type as activation function, the *rectified linear unit*, or ReLU for short, given by

$$\text{ReLU}(x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{5.9}$$

It was initially created to introduce sparsity to a network, which was believed to improve performance and was proven to outperform the sigmoid function in deep networks [20]. There are however further different types of ReLU functions, which were also tested and compared to see if using any of them increases the model performance. Plots for these different ReLU functions are shown in figure 5.13. Other types of ReLU considered here are all variations of the leaky ReLU function, which adds a slope parameter a for negative x values and is therefore given via the expression

$$\text{leaky ReLU}(x) = \begin{cases} x & \text{for } x \geq 0 \\ ax & \text{for } x < 0 \end{cases} \tag{5.10}$$

Variants considered here are the parametric and random ReLU. They differ from its leaky variant by the way its parameter a is defined. For a leaky ReLU this parameter is simply chosen when creating the network, but for a parametric ReLU it is a parameter that is trained through backpropagation along with the neural network model and for the random ReLU it is randomly sampled from a uniform distribution [33]. The initial model uses a leaky ReLU activation function for its linear layers with a value of 0.1 for the negative slope, which will act as a comparison to judge the impact of tested ReLU variants have on the model.
Comparing test performance for models only differing in the used activation function yields the plots shown in figure 5.14, which illustrate a significant improvement for a model using the random ReLU activation function. As an additional side effect, this activation seems to have a regularizing impact on a model since it is the only one out of all tested models

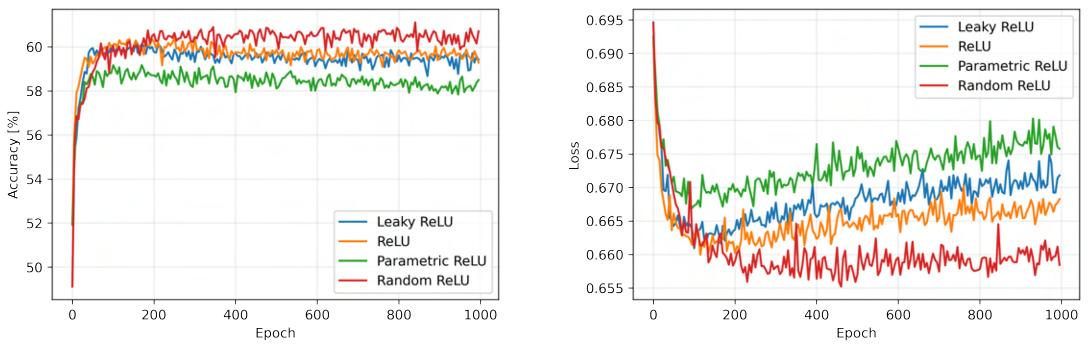

Figure 5.14: Comparison of the effect different types of the ReLU activation functions have on the test model performance. From this one can conclude that the random ReLU shows the greatest benefit for the model visible through a clear performance advantage.

that only shows slight overfitting effects. Due to these two advantages using a random ReLU has over the other tested alternatives, it shall be used as the activation function in the final model.

5.2.5 Batch Size

The batch size parameter of a neural network determines how many data samples are used to compute a weight update in each iteration. Typical batch sizes have values from 64 to 512 meaning that the number of 10,000 used within the initial model is relatively large. There is empirical evidence that models using large batch sizes show issues with generalization [34], which is why a variety of smaller batch sizes and their effect on the model performance were tested.

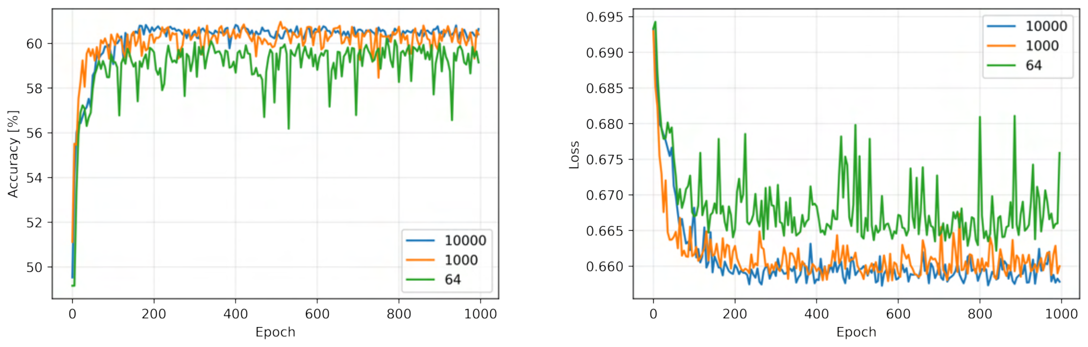

Figure 5.15: Plots comparing different values of batch size in the model. Lowering this size does not improve the results and leads to strong fluctuations.

Test results are shown in figure 5.15 and compare batch sizes of 1,000 and 64 to the standard batch size of 10,000 used so far. The model used for these comparisons is the final network model further described in section 5.4, which includes all investigated changes resulting from the conducted experiments. Results of the comparisons show that no advantage can be gained from smaller batch sizes for the model, since the test performance is equal or worse and training times as well as generalization capabilities did not improve. Due to these results the batch size for all further experiments was kept unchanged at 10,000.

5.3 Regularization

This section will try to solve the problem of overfitting present in the initial model. Different regularization techniques were used on the model and their effects compared, to find an optimal solution. As mentioned before the problem of overfitting is a common one in neural network analysis and therefore a lot of approaches exist to minimize the effects it has on training results. However, currently there does not appear to be a standard procedure or method that can deal with overfitting to great effect independent of the problem. Because of this, there is a lot of trial and error involved in fighting overfitting. In general regularization only affects the training process with the goal to adjust training in a way that makes the predictions more reliable. The several approaches tested and their effects on the model performance are discussed in the following.

5.3.1 Batch Normalization

This method tries to create similar data distributions within the mini batches used in training. Since this decreases the variance of layer inputs, it will have a regularizing effect on the network. In order to achieve this, data points in a batch are normalized to zero mean and unit variance with respect to the batch mean μ_B and the batch variance σ_B^2, meaning that a normalized input data point $\hat{x}_i$ is calculated via

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{5.11}$$

where ϵ is a small constant number added for numerical stability. This is done for each input dimension of a layer separately. As a second step of this transformation the batch normalized values y_i are produced using the relation

$$y_i = \gamma \hat{x}_i + \beta \tag{5.12}$$

and afterwards passed to the layer. The parameters γ and β scale and shift the normalized data and are further parameters to be learned by the network. Batch Normalization applying the mini batch statistics is only used when training the network whereas during testing the normalization is given with respect to the mean and variance of the population [35].
Batch normalization was tested for all three different model structures discussed in section 5.2.3. The effect of this regularization was then compared to the original network performance showing overfitting. At first batch normalization was added to all layers except the

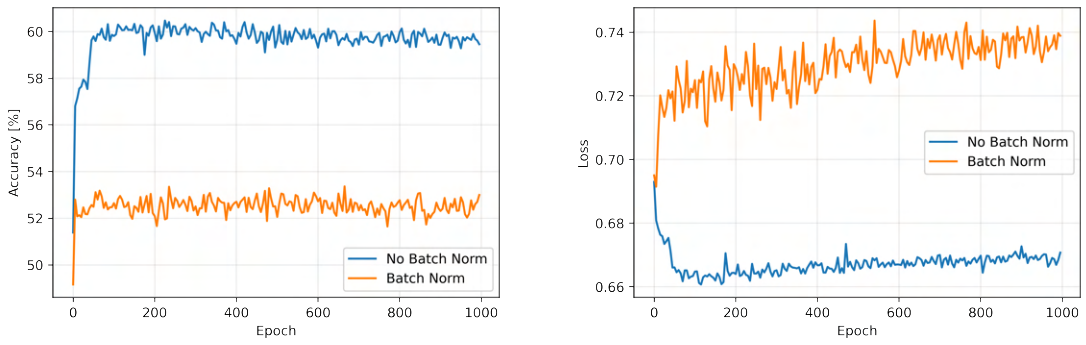

Figure 5.16: Comparison between identical models where one uses batch normalization and the other one does not. It is clearly visible that batch normalization harms the model and is therefore unsuited for regularizing the current model.

output layer, but this lead to a strong decrease in overall performance of the model, which is why for the comparison tests it was only used on the input layer for each network. The networks use the one dimensional batch norm layer provided by PyTorch using the order Layer → Activation → Batch Norm. Results for these tests were mostly the same across all tested configurations, as an example figure 5.16 shows the comparison for the deep network test. It can be seen that applying batch normalization leads to a major reduction in training and test performance, therefore making it unsuited to fight overfitting within the current model. This is the case for all different tested model structures and does not change when adjusting batch normalization parameters or the ordering of layers. Due to these results another method for regularization needs to be found for this model.

5.3.2 Weight Decay

The most intuitive way to reduce complexity of a model is to simply lower the total number of weights through changes in the model structure which decrease the number of neuron connections. Weight decay uses a different approach to yield a lower complexity model by keeping weights from growing arbitrarily large. This works as a simplification, since a restriction on the model is introduced reducing its complexity. It can be achieved by adding a penalty term for large weights to the error function, for instance given by

$$E(x) = E_0(x) + \frac{1}{2}\lambda \sum_i w_i^2 \tag{5.13}$$

where E_0 is the original error function and the parameter λ influences the strength of the weight decay effect. Using gradient descent this then further introduces an extra term $-\lambda w_i$ to the weight updates, effectively creating exponential decay for large weights. This method has the potential to regularize a neural network model, because of its reduction of model complexity [36].

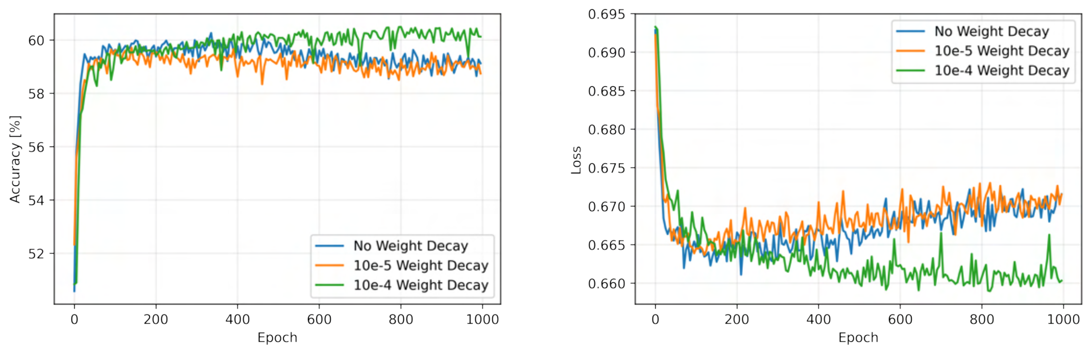

Figure 5.17: Test results for comparing different weight decay parameter values. it can be seen that the choice of 10^{-4} as a weight decay parameter leads to a significant regularization effect for the model.

In practice one needs to find a balance between making the parameter λ too large so it disturbs the network too much, which hinders learning, and making it too low to have an impact on the model at all. Two values trying to strike this balance were tested with figure 5.17 showing their results for evaluating model test performance. It can be seen that a value of 10^{-5} is too small to significantly affect the model as test performance is only slightly enhanced while still showing strong overfitting. The other tested value however is able to effectively fight overfitting and converge to better test metrics than the model without weight decay. These results are nevertheless not clear enough to consider weight decay for the final network model.

In addition better results could be obtained by using dropout, which is why weight decay was not implemented into the final model.

5.3.3 Dropout

Dropout was developed to reduce overfitting by reducing co-adaptations of neurons, which describes an effect where the results a neuron produces are only useful in the context of other specific neurons. This co-adaptation reduction is achieved by creating a variety of sparse networks only featuring a subset of the original neurons, to make sure that features learned from these neurons are helpful for the network in general. Such a sparse network is obtained by randomly deactivating a fraction of neurons in each layer according to a random distribution, where the probability to switch a neuron off is a parameter which can be chosen arbitrarily. This procedure is visualized in figure 5.18 and can prove to be really valuable in regularization. The reasoning why dropout helps to regularize the network is that its neurons can not rely on each other anymore, producing feature recognition, which is generally more helpful for the specific task of a model and leads to a better generalization in the end.

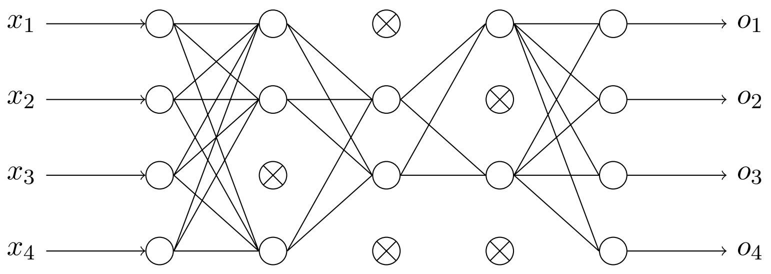

Figure 5.18: Example for a network after applying dropout to all three hidden layers, where inactive nodes are marked by X. The random deactivation of nodes is repeated in every iteration, creating a variety of different neuron connection configurations. Since deactivating a node essentially removes its contribution from the network, this forces it to generalize better.

During testing a mean network using all neurons is considered, which is making up for using more neurons than the training network by multiplying all weights with a factor of 1-p, where p is the dropout probability. Due to the nature of dropout preventing some information to advance through the layer it is not used on output layers, using dropout on the input layer is proven to be possible, but was not considered in the experiments conducted here. Further, the dropout probability parameter has to be chosen carefully since the training accuracy is diminished as an effect of dropout usage. Because of this an optimal parameter value needs to be found to obtain the regularization effect needed while doing as little harm as possible to the training process [37].
Dropout was placed on every hidden layer after the activation function during experiments. The base network model used is the same as in section 5.2.3, not using skip connections. Multiple different dropout probabilities were tested to find an optimal value for the desired regularization effect, without penalizing the network training too much. Due to the relatively low numbers of neurons in all layers, the tested values are relatively small. A comparison of the achieved model performances provided by different dropout values is illustrated in figure 5.19, where the comparison does not use dropout at all. The results show a clear reduction of overfitting from using dropout for all tested probability values, visible by the fact that test runs either converge or keep increasing in accuracy while dropping in loss value. Out of all tested probabilities, a dropout of 5% shows the best performance, which is also slightly better than the best performing weight decay test. Because of this a 5% dropout

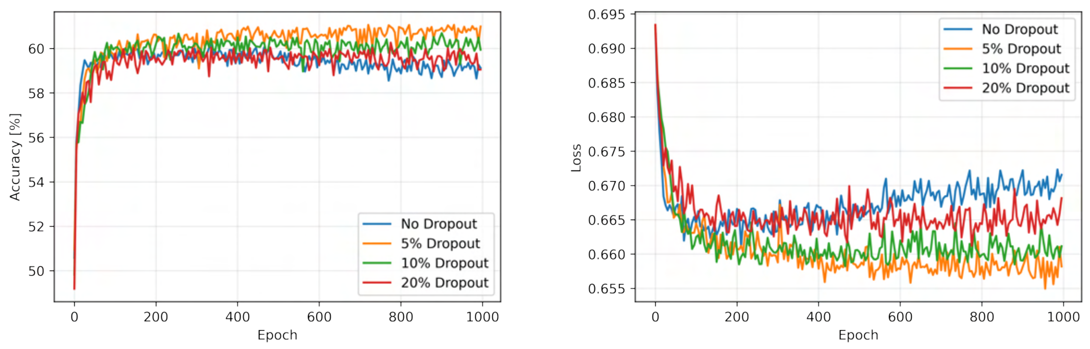

Figure 5.19: Comparison of different dropout values and their effect on model test performance. In general all tested values are able to achieve regularization, but the higher dropout probabilities only lead to small improvements, while a 5% probability seems to work best.

value shall be used in the final model for regularization. Attempts to create a model using a combination of weight decay and dropout for regularization in the hopes of an even stronger regularized network model failed.

5.3.4 Bigger Datasets

As was mentioned in section 3.3 overfitting occurs when the model capacity is big enough to practically memorize the data. This leads to the conclusion that more data is an effective method of fighting overfitting in the model. This experiment aims to test the extend of this effect by comparing two models where the only difference is the size of their input dataset. One model uses 120 thousand events while the other works with twice the amount of 240 thousand events.

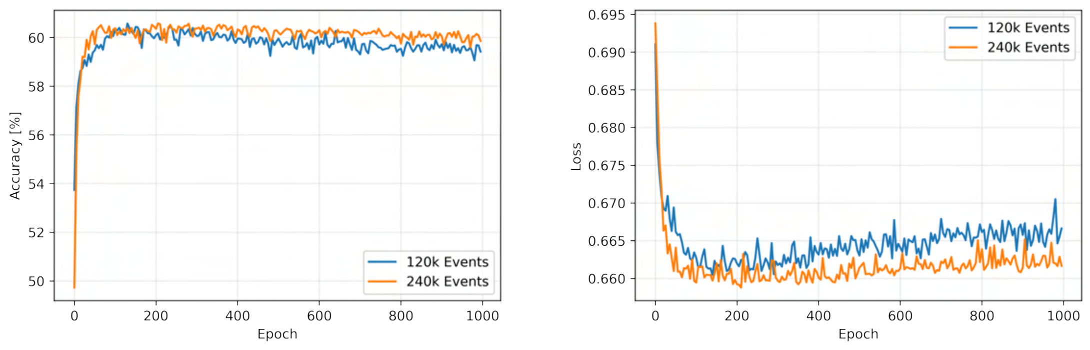

Figure 5.20: Comparing the impact of different sizes for used datasets on the model. Maximum achieved performance metrics remain roughly the same, however for a bigger data set the overfitting effect is reduced slightly.

The results illustrated in figure 5.20 clearly show a slight reduction of overfitting when doubling the amount of input data. This however comes at the cost of a reduced training performance similar to dropout. While large increases in dataset size can clearly lessen or even eliminate the overfitting effect this also comes at the cost of longer computing times and bigger memory requirements as discussed in section 3.3. This means a trade off between regularization achieved by using this method and keeping computational cost as low as possible has to be made. Because of this it was chosen that dataset size in the model should be increased whenever possible but only to a maximum of 240 thousand events, even though in theory a total of 2.4 million events would be available. As another consequence of this, regularization can not be achieved by simply adding a significantly larger dataset and the usage of dropout in the final model is also required.

5.4 Dataset Comparisons

After the considerations made in the previous sections a final model was constructed, which shall now be used to compare the effects of investigated variables from chapter 4. In order to achieve this, each of the jet pair variables is tested in the model on its own and the results are compared with each other. As a comparison two further experiments with the final model are conducted, one where only the basic jet variables are used, which also only uses a single preprocessing network similar to the initial model and one where all jet variables added to the datasets in various experiments are tested using a total of five preprocessing networks in the final model. The former will provide an estimation of the influence structure changes and added jet variables have on performance compared to the initial model, while the latter shows the maximum performance achievable with all changes made to the model and including all tested variables. Furthermore, comparing these two extra experiments allows to estimate the negative impact additional preprocessing networks have on model performance.

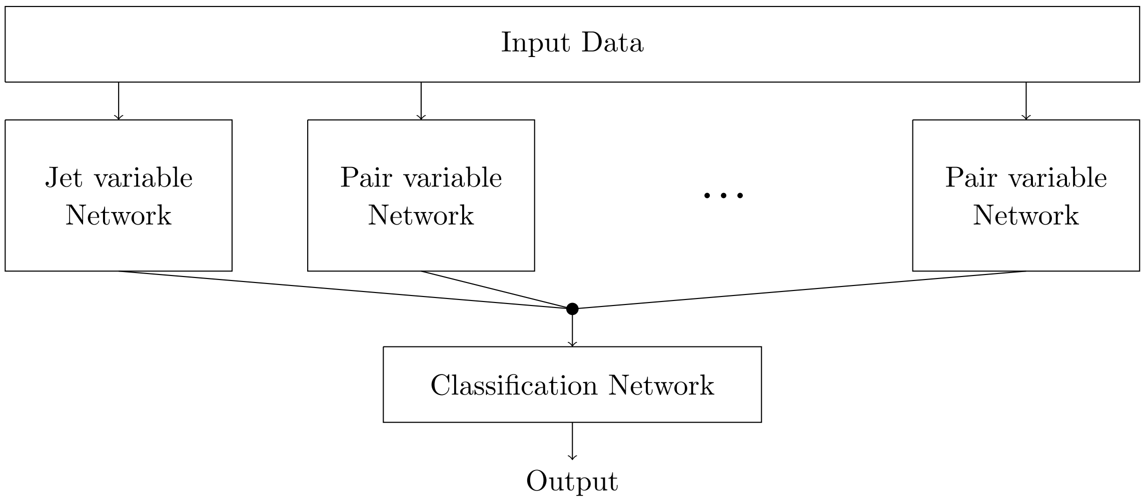

Figure 5.21: Schematic overview of the overall design of the final network. An Input dataset is split over multiple preprocessing networks, where the produced outputs are combined into an input for the final network handling the classification task. There are as many pair variable networks in the used model as there are pair variables tested in an experiment.

The final model is now structured as illustrated in figure 5.21. Other changes to the network details are summarized in table 5.2. The various tests of different datasets, testing the effectiveness of previously found variables are as follows. There is one test for each pair variable, judging its performance when adding its information and an additional preprocessing network to the model. Further, related jet variables are also added to the dataset, meaning they are used to calculate the pair variable or show similar properties of jets. These variables are the ϕ angle and pseudorapidity η in the case of angle difference, the mass of jets m_j for the relative mass variable, the absolute momentum $|j|$ for the angle between jets and transverse jet momentum $p_{T,j}$ in the case of jet pair transverse momentum variable. These jet variables are added on top of the basic energy, momentum component and particle number jet variables present in all datasets so far. In addition the two extra experiments use all jet variables.

	Preprocessing Layers	Classification Layers	Number of total Events	Activation Function	Skip Connections	Dropout
Initial Model	4	4	120000	Leaky ReLU	Yes	No
Final Model	5	10	240000	Random ReLU	No	5%

Table 5.2: List of the differences in model structure between the initial and final models. Not changed were the neuron numbers, learning rate, batch size and used optimization and loss functions.

5.4.1 Pair Variables

Each of the conducted experiments will be compared to each other and to the result achieved by the initial network model when used without the b jet information but with a doubled dataset size of 240 thousand events to make it more comparable to results obtained by the other experiments.

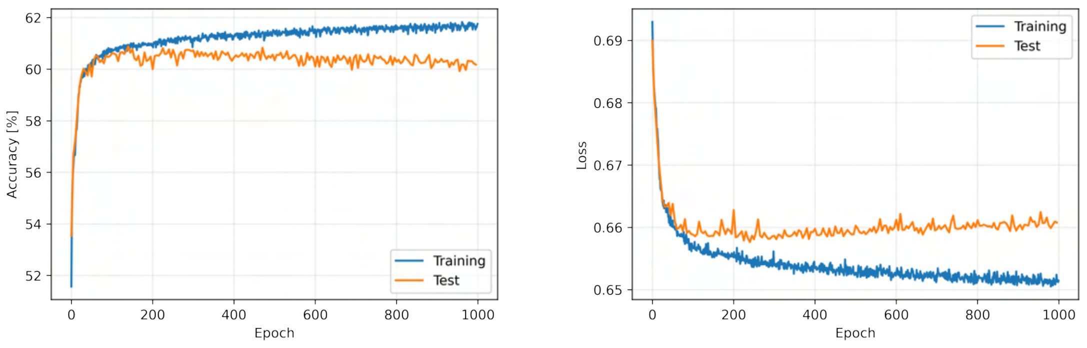

Figure 5.22: Performance of the initial model when using twice the amount of input data while excluding the b jet truth variable from its dataset. Qualitatively the results did not change significantly, however the highest achieved accuracy in testing is slightly higher than it was when using less events.

The performance results of this comparison run are depicted in figure 5.22. The results are generally similar to the original initial network performance metrics using less events, while showing a slightly higher maximum accuracy achieved at approximately 60.7%. When considering the other performance metrics mentioned in section 5.1.2, it can be seen that this model achieves a relatively high specificity of 68%, while scoring a relatively low recall of 53%. This means that the comparison model is performing well when it comes to correctly identifying HZ events but is lacking a good performance when trying to classify HH events into the correct category.
Figure 5.23 now shows a comparison between the four different pair variable datasets using the final model. Improvements over the comparison model are in general minor and differ slightly for each tested dataset. Relative mass and the pair angle between jets both perform poorly at a maximum test accuracy of roughly 60.8%, only barely better than the comparison. Both other variables, angle difference and transverse momentum perform better with the highest accuracy out of these tests at around 61% being achieved by the transverse momentum dataset, while the performance of the angle difference lies between the two extremes at approximately 60.9%. The other performance metrics for these runs show further differences between these various variable datasets. The relative mass dataset shows roughly the same values as the comparison model while the angle difference dataset has similar problems

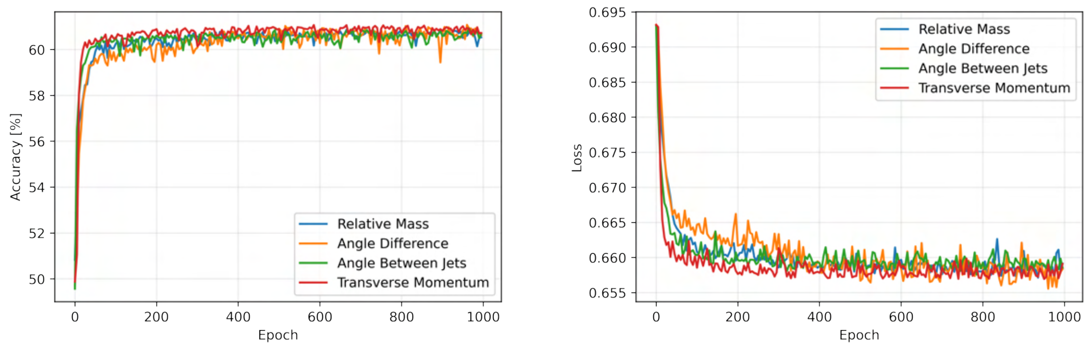

Figure 5.23: Plots for the model performance using different datasets containing one of the pair variables and their respective related jet variables. In general performance is similar and only slightly better than the comparison run using a simple dataset on the initial model.

with its classification achieving an even higher specificity at 75% and a lower recall only achieving 48%. the other two datasets lead to an improvement of recall while reducing the specificity to 60% and 61% respectively compared to the comparison model. As a result no general improvement for these additional performance metrics could be observed for these tested variable datasets.
Results from the other two dataset experiments are shown in figure 5.24, comparing their test performance. While these results are better than the previous ones in general, the differences are again only minor. The full dataset using all variables can easily achieve the same maximum accuracy as the transverse momentum dataset, but no significant improvement beyond that. The jet variable data set however can slightly improve the results of the previous experiments to an accuracy of roughly 61.1% in testing.

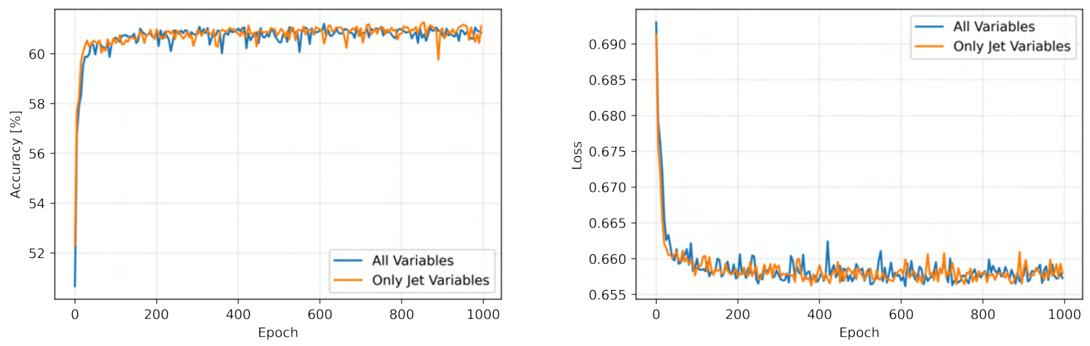

Figure 5.24: Comparison of the two additional data sets using all tested variables or only all used jet variables. Despite the significant difference in the amount of relevant data points provided to the model both models perform equally well.

These results suggest, that multiple preprocessing networks are not optimal for improving model performance and as a result a different approach for implementing the tested pair variables is necessary. Further, an improvement in accuracy and loss was observed for a model adding more variables to the dataset, meaning the model is still capable of learning more features provided the data. Evaluating other performance metrics shows that using both of these datasets leads to a slight improvement of recall while keeping the specificity at the same level as the comparison model. In particular the jet variable dataset performs better with a recall of 58% compared to the recall value of the full variable dataset at 55%. This shows again that a model using all tested variables suffers from its additional networks making it perform worse despite its informational advantage. Precision was unchanged between all conducted model tests at a value slightly above the achieved model accuracies and was therefore irrelevant for an evaluation of model performance.

5.4.2 Jet Pairs

Because only minor improvements were achieved in the previous tests, a slightly altered approach for the dataset and model input set up shall be tested. This final experiment aims to test if using all possible jet pairs of an event instead of a list containing all jets leads to an improvement of the results without major changes to the derived model structure. Setting up the model and dataset in this way has both advantages but also new challenges. An advantage over the previously used model is that no additional preprocessing networks are needed since the variables of all jet pairs contain all of this information already, which makes the model simpler and eliminates the need for networks with variable input size. One big downside of this approach however is that the number of possible pairs becomes quite large for most events in the Monte Carlo simulation, which leads to bigger datasets and more memory and computing time needed when using this model. Because of this, the maximum number of jet pairs is restricted to a maximum of 30 per event, but even for events with more pairs than that the most important pairs will still end up in the dataset since jets created by FastJet are sorted by transverse momentum, meaning the most relevant created jets of an event will always be present in the dataset provided to the model. When applying this model to experimental data the input would need to be configured in a similar manner to make the model usable.

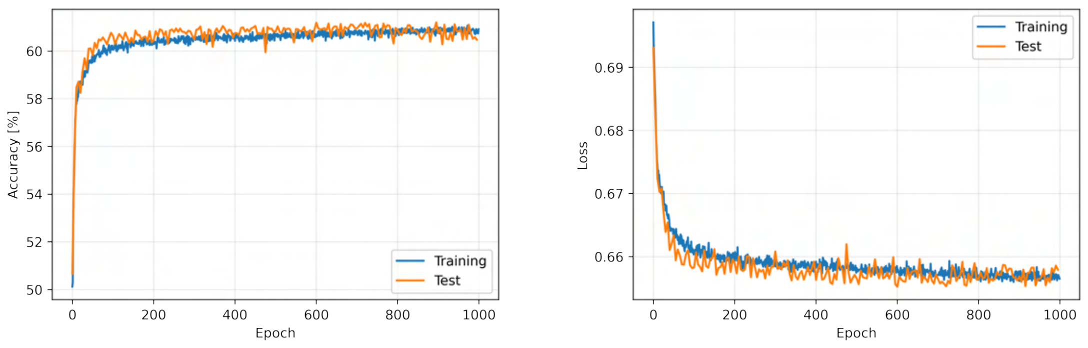

Figure 5.25: Performance of the new network model using only basic variables of jet pairs. Both accuracy (61%) and loss of this model are already comparable to most previous results.

Two experiments were conducted, one only considering the basic variables energy E, momenta p_x, p_y, p_z and particle number as a comparison, to check the effects of this dataset structure change, and a full run containing all variables tested before both for jets and jet pairs. The results for the comparison experiment, are shown in figure 5.25. Even without adding any further variables this model can now already achieve performance similar to most of the previously tested datasets at roughly 61%. This result already shows that the possible classification capability of this newly arranged model seems to be higher, therefore an even better result is expected when adding all tested variables to the dataset in the next experiment.

Performance results for the model after adding all variables to the jet pair dataset are shown in figure 5.26. A massive improvement over the previous experiments is visible and the model outperforms any other experiment conducted significantly. The highest accuracy achieved in

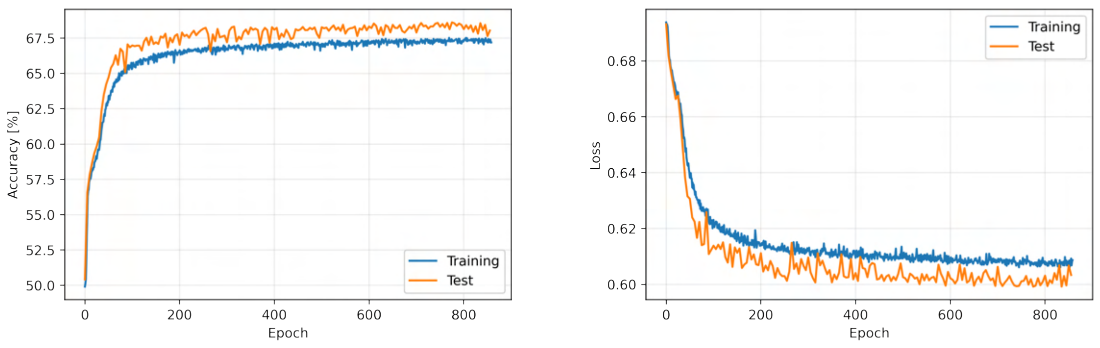

Figure 5.26: Final results for the model performance when using a full variable list for all possible jet pairs as an input. There is a significant improvement compared to the initial model visible leading to a far higher accuracy of around 68.5%. Further, an even stronger over-regularization effect compared to other data set tests exists within this model as can be seen by the test accuracy and loss clearly outperforming the training results.

this run is approximately 68.5%, which is also better than the initial model when using the b jet boolean variable. As a conclusion the model previously used, adding one or multiple additional preprocessing networks, does not seem to be suited for this problem and shifting the focus from singular jets to pairs of them is a significantly better approach for organizing the model input data.

Because of the big success of this change in approach, an even stronger restriction on input accessible to the network could be helpful for even better results, however this comes at the cost of a more work and time intensive preliminary analysis and data selection. One possible approach would be to find a small number of jet pairs for each event believed to be highly likely to be higgs particles through classical or neural network analysis and only using these pairs in the current or a similar model to classify the type of final state. This would also help to make the model simpler and lessen the hardware requirement due to a large reduction in the amount of input data, leading to shorter training times, which would make this analysis more applicable, or alternatively, would enable using larger dataset sizes given the resources. But even without going this extra step improving the current state of the model is still possible. For instance there is still model fine tuning for the parameters possible, networks are still relatively small and shallow limiting their capacity. Additionally the model is overall over-regularized due to the combined regularization effect of a bigger dataset and applied dropout, which is visible by the fact that test performance is better than training. Other alternatives such as deep convolution structure allowing for a large number of input variables are also possible directions for model enhancements.

As a general conclusion, it was shown that distinguishing between HH and HZ final states using neural networks is in principle possible, but a lot of improvements still have to be made in order to create a model capable of showing the necessary performance to allow its usage for analyses in the field of di-higgs production. Further the best possible approach to this issue might still be fairly different from the current model making more research into this topic necessary.

Chapter 6

Conclusion

This work has successfully introduced new variables and clearly shown their ability to discriminate HH and HZ final states. Further, drastic changes to the structure of both the neural network model and its input dataset used in this analysis lead to the desired improvement of final state classification accuracy. This was able to solve prominent problems of the original neural network model and further investigate effects of alternative structures on its performance.
Due to time limitations only relatively general variables were able to be tested, however for the purpose of classification there might still be more specific variables to be found, which are sensitive to differences in particle properties such as spin to enable an even more effective discrimination. It was also discussed that a stronger preselection of input data has the potential to improve the model further. The neural network model used also has the potential to be expanded into an even deeper network to improve performance, which could not be done due to hardware and time limitations. Both of these potential improvements to the model are expected to make even better classification performance of the model achievable.
Future work could therefore focus on finding more useful variables or extensively expanding the data preselection and model structure to allow a further improvement in its classification capabilities. In addition tests so far were conducted on purely simulated data without any noise or other background processes, so additional work testing and optimizing the network model for these circumstances is certainly required.
In general it was proven that the used approach of classifying the HH and HZ final states via a neural network model works and still shows enough room for further improvement to achieve accuracies sufficiently high to make usage for an analysis possible. This approach could then be used for discovery and measurement of higgs pair production with the possibility of discovering new physics along the way.

Bibliography

[1] R. L. Workman et al. "Review of Particle Physics". In: *PTEP* 2022 (2022), p. 083C01. DOI: 10.1093/ptep/ptac097.

[2] W. N. Cottingham and D. A. Greenwood. *An Introduction to the Standard Model of Particle Physics*. 2nd ed. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511791406.

[3] David J Griffiths. *Introduction to elementary particles; 2nd rev. version*. Physics textbook. New York, NY: Wiley, 2008.

[4] Alessandro Bettini. *Introduction to Elementary Particle Physics*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511809019.

[5] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021.

[6] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020.

[7] B. Abi et al. "Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm". In: *Phys. Rev. Lett.* 126 (14 2021), p. 141801. DOI: 10.1103/PhysRevLett.126.141801.

[8] T. Aaltonen et al. "High-precision measurement of the W boson mass with the CDF II detector". In: *Science* 376.6589 (2022), pp. 170–176. DOI: 10.1126/science.abk1781.

[9] J Beacham et al. "Physics beyond colliders at CERN: beyond the Standard Model working group report". In: *Journal of Physics G: Nuclear and Particle Physics* 47.1 (2019), p. 010501. DOI: 10.1088/1361-6471/ab4cd2.

[10] K Hübner. "Fifty years of research at CERN, from past to future: The Accelerators. 50 years at CERN, from past to future. The Accelerators". In: (2006). DOI: 10.5170/CERN-2006-004.1.

[11] Fabiola Gianotti and Gian Francesco Giudice. "A roadmap for the future". In: *Nature Physics* 16.10 (2020), pp. 997–998. ISSN: 1745-2481. DOI: 10.1038/s41567-020-01054-6.

[12] Lyndon R Evans and Philip Bryant. "LHC Machine". In: *JINST* 3 (2008). This report is an abridged version of the LHC Design Report (CERN-2004-003), S08001. 164 p. DOI: 10.1088/1748-0221/3/08/S08001.

[13] David d'Enterria. "Physics at the LHC: a short overview". In: *Journal of Physics: Conference Series* 270 (2011), p. 012001. DOI: 10.1088/1742-6596/270/1/012001.

[14] Simone Amoroso. "Precision measurements at the LHC". In: *PoS* RADCOR2019 (2020), p. 001. DOI: 10.22323/1.375.0001.

[15] Burkhard Schmidt. “The High-Luminosity upgrade of the LHC: Physics and Technology Challenges for the Accelerator and the Experiments”. In: *Journal of Physics: Conference Series* 706.2 (2016), p. 022002. DOI: 10.1088/1742-6596/706/2/022002.

[16] The ATLAS Collaboration et al. “The ATLAS Experiment at the CERN Large Hadron Collider”. In: *Journal of Instrumentation* 3.08 (2008), S08003–S08003. DOI: 10.1088/1748-0221/3/08/s08003.

[17] Laith Alzubaidi et al. “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of Big Data* 8.1 (2021), p. 53. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00444-8.

[18] Kerstin Beer et al. “Training deep quantum neural networks”. In: *Nature Communications* 11.1 (2020), p. 808. ISSN: 2041-1723. DOI: 10.1038/s41467-020-14454-2.

[19] J. Feldman and Raul Rojas. *Neural Networks : A Systematic Introduction.* Berlin, Heidelberg, GERMANY: Springer Berlin / Heidelberg, 1996. ISBN: 9783642610684.

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323.

[21] Bernhard Mehlig. *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers.* Cambridge University Press, 2021. DOI: 10.1017/9781108860604.

[22] Stephan Jokiel. *Klassifizierung von HH Endzuständen mithilfe von Machine Learning Methoden.* 2021.

[23] Simone Alioli, Paolo Nason, Carlo Oleari, and Emanuele Re. “A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX”. In: *JHEP* 06 (2010), p. 043. DOI: 10.1007/JHEP06(2010)043. arXiv: 1002.2581 [hep-ph].

[24] Christian Bierlich et al. *A comprehensive guide to the physics and usage of PYTHIA 8.3.* 2022. DOI: 10.48550/ARXIV.2203.11601.

[25] R. Brun and F. Rademakers. “ROOT: An object oriented data analysis framework”. In: *Nucl. Instrum. Meth. A* 389 (1997). Ed. by M. Werlen and D. Perret-Gallix, pp. 81–86. DOI: 10.1016/S0168-9002(97)00048-X.

[26] Biagio Di Micco, Maxime Gouzevitch, Javier Mazzitelli, and Caterina Vernieri. “Higgs boson potential at colliders: Status and perspectives”. In: *Reviews in Physics* 5 (2020), p. 100045. DOI: 10.1016/j.revip.2020.100045.

[27] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. “FastJet user manual”. In: *The European Physical Journal C* 72.3 (2012). DOI: 10.1140/epjc/s10052-012-1896-2.

[28] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems.* Red Hook, NY, USA: Curran Associates Inc., 2019.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* 2015. DOI: 10.48550/ARXIV.1502.01852.

[30] Lei Huang et al. *Normalization Techniques in Training DNNs: Methodology, Analysis and Application.* 2020. DOI: 10.48550/ARXIV.2009.12836.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition.* 2015. DOI: 10.48550/ARXIV.1512.03385.

[32] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks.* 2015. DOI: 10.48550/ARXIV.1511.08458.

[33] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. *Empirical Evaluation of Rectified Activations in Convolutional Network.* 2015. arXiv: 1505.00853 [cs.LG].

[34] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.* 2016. DOI: 10.48550/ARXIV.1609.04836.

[35] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. DOI: 10.48550/ARXIV.1502.03167.

[36] Anders Krogh and John A. Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Proceedings of the 4th International Conference on Neural Information Processing Systems.* NIPS'91. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1991, pp. 950–957. ISBN: 1558602224.

[37] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors.* 2012. DOI: 10.48550/ARXIV.1207.0580.

Selbständigkeitserklärung

Ich versichere hiermit, die vorliegende Arbeit mit dem Titel

Discrimination of HH and HZ Final States Using Neural Networks

selbständig verfasst zu haben und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Lars Linden

München, den 24. November 2022