Employing the Matrix Element Method with Neural Networks to Search for Higgs Pair-production

Christoph Ames

München 2024

Employing the Matrix Element Method with Neural Networks to Search for Higgs Pair-production

Christoph Ames

Dissertation
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Christoph Ames
aus Ascot, England

München, den 31.7.2024

Erstgutachter: Prof. Dr. Otmar Biebel
Zweitgutachter: Prof. Dr. Thomas Kuhr
Tag der mündlichen Prüfung: 24.09.2024

Zusammenfassung

Neuronale Netze (NN) werden zur Berechnung von Matrixelementgewichten trainiert, um die verfügbare Information eines gemessenen Endzustands im Rahmen der theoretischen Erwartung der Matrix-Element-Methode (MEM) möglichst vollständig zu erfassen und für die Auswertung der Messdaten zu verwenden. Zugleich wird durch den Einsatz von NN die CPU-aufwändige Berechnung der Matrixelementgewichte zur Klassifizierung einzelner Endzustände stark beschleunigt. Dabei werden die Matrixelementvorhersagen des NN als proof-of-principle zunächst zur Klassifizierung und Separierung von HH- und HZ-Ereignissen benutzt. Die HH-Endzuständen beinhalten, unter anderem, Beiträge von Higgs-Selbstwechselwirkungsdiagrammen (tri-H).

Die Matrixelementgewichte werden anhand von simulierten Daten mit zwei verschiedene Matrixelementen berechnet, jeweils für die HH- und die HZ-Endzustände. Die Trennung zwischen den Gewichten is schlechter für die HZ- als die HH-Ereignisse. Zwei NN-Arten werden untersucht: Feed-forward und Convolutional; diese lernen die HH- und HZ-Matrixelementgewichte vorherzusagen. Eine sehr starke Reduzierung der Rechenzeit im Vergleich zur MEM wird erreicht. Beide Netzarten sind erfolgreich bzgl. der Genauigkeit der HH-Matrixelementgewichten, weisen aber Schwierigkeiten bei den HZ-Matrixelementgewichte auf. Um völlig verstehen zu können, warum die Trennung der Gewichte bei HZ-Ereignissen schlechter ist, und warum die NN bei den HZ-Matrixelementgewichten auf Probleme stoßen, wäre eine tiefere Untersuchung der theoretischen Berechnungen der Matrixelemente notwendig, welches nicht im Rahmen dieser Forschung liegt.

Abstract

Neural networks (NN) are trained to calculate matrix element weights, with which information of a measured final-state can be gained via the Matrix Element Method (MEM) theoretical framework. At the same time, the use of NN accelerates the CPU-intensive computations of the matrix element weights, which are used to classify final-states. The predictions of the NN are used as a proof-of-principle to separate Higgs-Higgs (HH) and Higgs-Z (HZ) events. The HH events include final-states coming from super-positions of Higgs self-coupling (tri-H).

The matrix element weights are calculated for two different matrix elements, one for HH and one for HZ final states, using simulated data. The separation between these weights is worse for the HZ data than the HH data. Two types of neural networks, feed-forward and convolutional, are examined and trained to predict the HH and HZ matrix elements weights. The computation times are successfully heavily reduced compared to the standard calculations of the MEM. Both network types are successful in regards to predicting the HH matrix weights, but they struggle with the weights using the HZ matrix element. To fully understand why the weight separation is worse for HZ data and why the networks have more difficulties with the HZ matrix element would require a deeper investigation into the theoretical framework of the matrix elements, which lies beyond the scope of this research.

Contents

Introduction

The field of particle physics focuses on the study of some of the most fundamental aspects of our universe: Which particles exist and how do they interact with one another? Many experiments requiring hundreds and thousands of researchers using some of the largest and most precise machines in the world have been performed to try to answer these questions. This has lead to the construction of a theoretical model known as the Standard Model of Particle Physics (SM). The SM is likely the most well researched physical theory to date — it describes all known elementary matter particles and explains the origin of three of the four known forces. Over the second half of the 20th century, many of the particles predicted by the SM were found, with the final discovery being that of the Higgs boson in 2012 by the ATLAS [1] and CMS [2] detectors. This verified the SM as a self-contained theory that could be used as a very good model for explaining most known phenomena in particle physics.

However, there are still aspects of the SM that have not yet been fully explored. Some of the predicted decay modes have such low cross-sections, that only upper bounds for their branching ratios have been set, since successful measurements have not yet been performed [3]. One such interaction is Higgs self-coupling, where a Higgs boson decays to two further Higgs bosons. The strength of the self-coupling is proportional to the self-coupling parameter λ, which also appears in the Higgs potential and influences its shape [4]. The shape has significant importance, as it describes the past, current and future stability of the universe [5]. Additionally, questions regarding the cosmological constant [6], the phase transition of the early universe [7] and the naturalness problem [8] are tied to the Higgs potential and, therefore, the Higgs self-coupling parameter.

Due to the scarcity of decays containing Higgs self-coupling, the Matrix Element Method (MEM), a calculation-based analysis method, becomes attractive. The MEM calculates the likelihood that a given decay mode took place in an event based on the four-momentum of the measured final-state particles. Background events are separated from signal events via the differences in the resulting assigned probabilities. It is even possible to base an analysis solely on this method, as was done with precision mass measurements for the top quark [9].

The MEM does come with a downside, which is the amount of resources required for it to be performed. A large integration must be done for each event, leading to a very high amount of computational resources being needed. This makes the MEM inaccessible for smaller-scale analyses, and a solution to reduce the computation barrier would be required for this method to become more wide-spread.

Alongside the discovery of the Higgs Boson, the time around 2012 also had other exciting developments, but this time in the field of artificial intelligence: the revolution of deep learning [10]. In 2011 and 2012, *DanNet* [11] and *AlexNet* [12] greatly increased the standard of computer vision [13]. Both of these neural networks consisted of convolutional deep neural networks run on Graphics Processing Units (GPUs). Convolutional networks are commonly used in pattern-searching situations, where it is unclear where the pattern will appear in the dataset, e.g. looking for a face in a picture or a specific word in an audio file. However, the break-through wasn't due to the use of convolutional neural networks, but the implementation of networks on GPUs. This greatly improved the speed at which they could run [11] and therefore increased the scale of the problems they could tackle. Greater computational power and efficiency was becoming more and more important as datasets started to grow larger and more complex. These data sets enabled greater research, as they pushed the limits of Machine Learning (ML). The creation of ImageNet [14] enabled great strides in research of computer vision, and it was this data set that AlexNet was tested on.

While machine learning (ML) has had an impact on many branches of industry, it has also found numerous applications in the natural sciences [15, 16]. High energy particle physics is no exception [17, 18], as experiments in this field have to deal with extremely high volumes of data. For example, during the last year of the collection period Run 2 (2015-2018) at the Large hadron Collider (LHC) at CERN, the High Level Trigger of the ATLAS detector, which is the final layer that decides if an event is interesting for physics analyses, had a readout rate of up to ≈ 1.2 kHz, with each readout containing an average of 1 MB [19]. This highly complicated process has been supported by the use of ML, as have many analyses of the resulting data. A categorised summary of the use of ML in particle physics can be found in "A Living Review of Machine Learning in Particle Physics" [20].

There is still much to be explored in regards to combining ML with particle physics. This work focuses on using neural networks to implement the MEM, which by-passes the need for an integration over each event. Variables describing the final-state of an event are given to a neural network, which in return produces a weight. The decay mode that is used for this analysis contains Higgs self-coupling — a mechanism which is an ideal candidate for the MEM due to its theoretical certainty, yet heavy suppression.

The structure of this work is as follows: In Ch. 1, the underlying theory of the SM (Sec. 1.1), the MEM (Sec. 1.5) and ML (Sec. 1.6) is discussed. Following that, the software used in this work is introduced in Ch. 2. The analysis is presented in Ch. 3, which is split into data generation (Sec. 3.3), the implementation of the MEM (Sec. 3.4), possible additional parameters to support the neural networks (Sec. 3.5) and the application of the neural networks (Sec. 3.6). At the end, a summary of the analysis is given (Ch. 3.6.4).

Chapter 1

Theory

1.1 The Standard Model of Particle Physics: Particle Description

In this section, the theoretical background and particle content of the SM will be discussed. This will contain a mathematical description of the existence of leptons, quarks and bosons, while focusing on the Higgs boson, as the underlying Higgs physics are a key motivation for the decay mode studied in this work. Doing so will involve looking at some of the SM's shortcomings and how a better understanding of the Higgs boson, and more precisely the Higgs coupling parameter and Higgs potential, could shed some light on possible physics beyond the SM.

Sec. 1.1.1 is oriented towards [21].

1.1.1 Leptons, Quarks and Bosons

The SM consists of elementary particles, which can be sorted into three categories: leptons, quarks and bosons. These particles interact with each other via the four known forces: the electromagnetic force, the weak force, the strong force and gravity; the first three of which are mediated by the bosons of the SM. The SM consists of and produces mathematical descriptions for each particle type — it has even been used to predict the existence of some particles before they were experimentally observed, e.g. the top quark [22] and the Higgs boson [23]. After the following general overview of the particle types and the four forces, their mathematical derivations will be discussed.

Fermions

Leptons and quarks can be combined into the category of "fermions", which are defined by their half-integer spin value $S_z = \pm\frac{1}{2}$ and consist of all know matter particles. They also follow the Pauli-Principle and Fermi-Dirac statistics [24]; the latter of which is where they get their name from.

Leptons consist of six flavours (see Tab. 1.1), which themselves can be split into two groups: charged leptons and neutral leptons, also known as neutrinos. They come from the SU(2) symmetry (via the weak force, see Sec. A) and are introduced in doublets: each charged lepton is associated with a neutrino. Both particles in a doublet (also referred to as a generation) have the same weak isospin of $I = 1$, but their values of the third component of the weak isospin differ by the sign: $I_3 = -\frac{1}{2}$ for charged leptons and $I_3 = \frac{1}{2}$ for neutrinos. Therefore, by interacting weakly — specifically, by emitting or absorbing a W boson — they can change into one another by changing their electric charge Q according to the Gell-Mann-Nishijima formula: $Y = 2(Q - I_3)$[1] [24].

Quarks come from the SU(3) gauge symmetry involving the colour charge. The colour charge, which appears in the form of either *red*, *green* or *blue* (or the associated anti-colours *anti-red*, *anti-green* or *anti-blue*), allows multiple quarks of the same flavour to bind together via the strong force. Without the colour charge, the Pauli exclusion principle would seem to be broken for particles such as Δ^{++}, which consists of three strange-type quarks. The configuration of quarks has to be colour neutral (also called colourless or *white*), meaning there must be a red, green and blue charge for particles made of three quarks (known as baryons) or the associated anti-colours for antiquarks (antibaryons). For particles made of a quark and an anti-quark (mesons), there must be a colour charge plus anti-colour charge of the same colour. As a result, individual quarks can never be found by themselves (known as colour confinement [25]) and hadronise[2] when separated. The colour charge is not to be taken literally: quarks don't have colours in the conventional sense, this is simply nomenclature. The name comes from red, green and blue light overlapping to white light.
The quarks themselves also come in six flavours, split into three generations of doublets (see Tab. 1.1). Like with the leptons, the doublets are defined by their weak isospin, where each doublet has an isospin of $I = \frac{1}{2}$ and the upper quarks have a third component of weak isospin $I_3 = \frac{1}{2}$ and the lower quarks $I_3 = -\frac{1}{2}$. Also like the leptons, their electric charge differs by a total value of 1[3]; however, the up-type quarks have a value of $Q = +\frac{2}{3}$, whereas the down-type quarks have $Q = -\frac{1}{3}$ [24, 25].

Bosons and Forces

The strong force is mediated by gluons. It's strength[4] is the greatest of the four forces while having the shortest range (its range is about the size of a nucleus [24]). Gluons are coloured, therefore they are confined to colourless particles and don't freely propagate [25]. Although there are nine colour combinations, there are only eight gluon states[5], due to there being only eight generators for the SU(3) gauge symmetry. They couple only to quarks, other gluons and the Higgs boson [25].

The electromagnetic force is mediated by photons. It's strength is about 10^{-3} times that of the strong force [25], while having an infinite range [24]. Photons couple to charged leptons, quarks and the Higgs boson.

[1] Y is the weak hypercharge.

[2] New quarks are spontaneously created and bind to the original to create a colourless particle.

[3] In units of e

[4] The strength of each force is dependent on the distance and energy scale [25].

[5] Gluon states: $r\bar{g}$, $g\bar{r}$, $r\bar{b}$, $b\bar{g}$, $b\bar{g}$, $g\bar{b}$, $\frac{1}{\sqrt{2}}\left(r\bar{r} - g\bar{g}\right)$, $\frac{1}{\sqrt{6}}\left(r\bar{r} + g\bar{g} - 2b\bar{b}\right)$

	Generation	Flavour	Electric Charge [e]	Mass [GeV]
Leptons	1	Electron e^-	-1	0.51×10^{-3}
		Neutrino ν_e	0	$< 0.8 \times 10^{-9}$
	2	Muon μ^-	-1	0.113
		Neutrino ν_μ	0	$< 0.19 \times 10^{-3}$
	3	Tau τ^-	-1	1.777
		Neutrino ν_τ	0	$< 0.182 \times 10^{-3}$
Quarks	1	Up u	$+2/3$	2.16×10^{-3}
		Down d	$-1/3$	4.67×10^{-3}
	2	Charm c	$+2/3$	1.27
		Strange s	$-1/3$	0.093
	3	Top t	$+2/3$	173
		Bottom b	$-1/3$	4.18

Table 1.1: Basic properties of SM leptons and quarks, in natural units. Antiparticles have the same masses, but opposite charges. Masses are taken from [3].

Type	Electric Charge [e]	Interaction Type			Mass [GeV]
		weak	strong	electromag.	
$W^\pm$	± 1	✓	✗	✗	80.4
Z	0	✓	✗	✗	91.2
γ	0	✗	✗	✓	0
g	0	✗	✓	✗	0

Table 1.2: Basic properties of SM bosons, in natural units. Masses are taken from [3].

***W*$^\pm$ and *Z* bosons** are the intermediate vector bosons that propagate the weak force, which is weaker than the strong force by a factor of about 10^{-8} [25]. They couple to leptons, quarks and the Higgs boson. Quarks are able to change from an up-type quark to a down-type quark and vice-versa via interactions involving a W boson. These interactions are described by the Cabibbo-Kobayashi-Maskawa matrix [26, 27], in which mass eigenstates are related to weak eigenstates:

$$\begin{pmatrix} d' \\ s' \\ b' \end{pmatrix} = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix} \begin{pmatrix} d \\ s \\ b \end{pmatrix} \approx \begin{pmatrix} 0.974 & 0.225 & 0.004 \\ 0.225 & 0.973 & 0.042 \\ 0.009 & 0.041 & 0.999 \end{pmatrix} \begin{pmatrix} d \\ s \\ b \end{pmatrix} \tag{1.1}$$

Gravity has the weakest strength of the four forces, with a strength of around 10^{-37} times that of the strong force. Its range is infinite; however, it isn't clear how gravity is mediated. Gravity is not described by the SM and the origin of its mechanisms is unknown. Gravity can be treated as a field theory, which postulates a gauge boson known as a "graviton", a massless spin-2 particle [28]. Gravitational waves, which were postulated by the theory of general relativity [29], were discovered in 2015 [30].

1.2 The Standard Model of Particle Physics: Brief Mathematical Description

In this section, a brief overview of the Higgs boson in the SM is presented. A detailed description of the mathematical derivation of particles in the SM and how this results in the Higgs boson can be found in App. A. The following mathematical description of the Higgs mechanism follows [25].

In the SM of particle physics, the origin of particles is derived using fields and gauge transformations. Under the Glashow-Salam-Weinberg (GSW) model, the Higgs boson is generated using a $\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ gauge transformation. To do this, a complex vector field is chosen as the following doublet:

$$\phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ \phi_3 + i\phi_4 \end{pmatrix}, \tag{1.2}$$

where ϕ^+ and ϕ^0 are complex scalar fields. ϕ represents a weak isospin doublet and the two components differ by a charge of $+1$. The Higgs boson itself is described by its potential,

$$V(\phi) = \mu^2 \left(\phi^* \phi\right) + \lambda \left(\phi^* \phi\right)^2. \tag{1.3}$$

For this potential to have a finite minimum, λ must be positive, as the $(\phi^*\phi)^2$ term begins to dominate for larger values. μ^2 is not restricted to be positive: choosing it to be negative, $\mu^2 < 0$, causes the minima of the Higgs potential to fulfil

$$\phi^\dagger \phi = \frac{1}{2}\left(\phi_1^2 + \phi_2^2 + \phi_3^2 + \phi_4^2\right) = \frac{v^2}{2} = -\frac{\mu^2}{2\lambda}. \tag{1.4}$$

This is similar to the minimum of the double-well potential, where ϕ is a scalar field. There, the minimum of the double-well potential is

$$\phi_{min} = \pm v = \pm \left| \sqrt{-\frac{\mu^2}{\lambda}} \right|, \tag{1.5}$$

which notably has two possible values. In this instance, where a ϕ is a complex vector field, the result is similar (compare Eq. 1.4 to Eq. 1.5). However, due to ϕ being a complex scalar field, there are infinite possible states that describe the minimum. Therefore, the minimum of the field ϕ can be chosen as

$$\langle 0 | \phi | 0 \rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v \end{pmatrix}. \tag{1.6}$$

By expanding the fields around the minimum, the spontaneous symmetry of the field can be broken, which results in a massive scalar boson and three massless Goldstone bosons, which will give the longitudinal degrees of freedom to the $W^\pm$ and Z bosons. The Higgs doublet then becomes

$$\phi(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + h(x) \end{pmatrix}. \tag{1.7}$$

The field $h(x)$ is known as the Higgs field. The covariate derivative of the $\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ gauge transformation can be chosen such that when it is applied to the field in Eq. 1.7, it results in new terms which represent the $W^{\pm}$ and Z bosons, and the photon. The resulting mass terms are

$$m_W = \frac{1}{2} g_W v \tag{1.8}$$

$$m_Z = \frac{1}{2} v \sqrt{g_W^2 + g'^2} \tag{1.9}$$

$$m_A = 0. \tag{1.10}$$

By defining

$$\frac{g'}{g_W} \equiv \tan(\theta_W), \tag{1.11}$$

the mass term for the Z boson can be rewritten as

$$m_Z = \frac{1}{2} \frac{g_W}{\cos(\theta_W)} v \tag{1.12}$$

from which the simple relation between the masses of the W and Z boson is revealed:

$$\frac{m_W}{m_Z} = \cos(\theta_W). \tag{1.13}$$

Since $m_W = \frac{1}{2} g_W v$, measuring the mass of the W boson and its coupling parameter leads to a value for the vacuum expectation value:

$$v = 246 \text{ GeV}. \tag{1.14}$$

Through the measurement of the Higgs boson's mass (see [1, 2]), using the vacuum expectation value and $m_H = \sqrt{2\lambda v^2}$, a value for the self-coupling parameter λ can be calculated. From this, using $v^2 = -\frac{\mu^2}{\lambda}$, a value for μ can be determined. In the end, the GSW model can be described by four parameters: g', g_W, μ and λ [25].

With this, the SM Higgs boson and its couplings to the W and Z bosons have been determined. Its couplings to fermions and their masses can be derived in a similar way (see [25]).

1.3 The Self-coupling Parameter

There are still many unknown aspects of the Higgs boson, such as origin of electroweak symmetry breaking, the strength of electroweak phase transition or the shape of the

Higgs potential. The later can only be properly determined by measuring the self-coupling parameter, which so-far has only been constrained [31].

The self-coupling parameter λ is important for some theories that look to solve questions that extend beyond the SM. For this, the relation $\kappa_\lambda = \frac{\lambda_{HHH}}{\lambda_{SM}}$ is used to describe whether the current understanding of Higgs theory accurately describes the underlying physics. If $\kappa_\lambda = 1$, then the SM would correctly cover Higgs self-coupling. In the following, a few open topics that involve either the Higgs potential or the self-coupling parameter are briefly discussed.

1.3.1 The Stability of the Universe

The universe has three possible states that describe its long-lived stability: stable, metastable and unstable. Stable means that the current vacuum potential $V(\phi)$ is at a global minimum $V(v)$. Metastable means that there is a ϕ much larger than the v that describes the global minimum, which in turn would mean that our universe is currently only at a local minimum. This implies that it would be possible for a tunnelling effect from the current local minimum to the global minimum; however, the barrier is large, hence metastability. The unstable scenario is the same as the metastable one, but the barrier from our local to the global minimum is small, meaning that there is a non-insignificant chance that the tunnelling could take place during the lifetime of our universe [5].

Where our universe lies relies heavily on the masses of the top quark and the Higgs boson. As can be seen in Fig. 1.1, our universe lies in the metastable region.

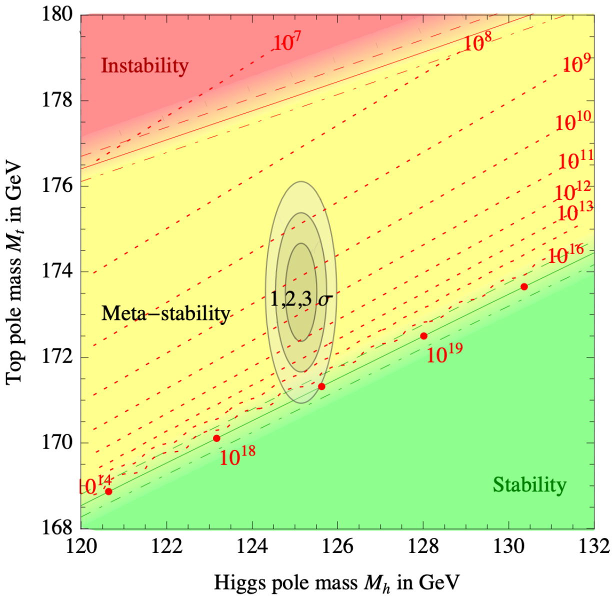


Figure 1.1: Stability of the universe depending on the masses of the Higgs boson, and therefore the self-coupling parameter since $m_H = \sqrt{2\lambda v^2}$, and the top quark. The boundary lines correspond to 1-σ variations of $\alpha_3(M_Z) = 0.1184 \pm 0.0007$, where α_3 is the strong gauge coupling. The powers of 10 (red dashed lines) indicate the energy in GeV for which the metastability or instability occur. Plot taken from [32].

According to our understanding of the Higgs boson and mechanism, the 125 GeV that is measured in a detector is associated with the mass of the Higgs boson. If, however, our theory of the Higgs field is incomplete and $\kappa_\lambda \neq 1$, then it could be that the measured 125 GeV does not correspond to the mass of the Higgs boson. This would have implications on the stability of our universe.

1.3.2 The Phase Transition of the Early Universe

At about 10^{-10} seconds after the Big Bang, the universe had cooled to a critical temperature, where it underwent a phase transition via spontaneous symmetry breaking. But, a question remains: was the phase transition of first-order or second-order? A first-order phase transition would be able to provide an explanation for the matter-antimatter asymmetry of our universe [33], due to the creation of bubbles in the phase with $V(\phi) \neq 0$. Particles scatter with the bubble walls, which can generate CP asymmetries and lead to transitions that generate more baryons that antibaryons [7]. A first-order phase transition would also lead to gravitational waves, due to the turbulent expansion of the bubble walls [7], which may be observable by some experiments [34].

Whether the transition is of first- or second-order depends on the mass of the Higgs boson. The measurement of 125 GeV indicates that the phase transition was of second-order; however, this is only regarding the SM. Beyond the SM, the nature of the electroweak phase transition is sensitive to any new physics which lie near the electroweak scale [34]. A better understanding of the self-coupling constant would therefore allow for the creation of better models for the electroweak phase transition.

1.3.3 The Cosmological Constant

The effective cosmological constant consists of

$$\Lambda_{eff} = \Lambda_B + \kappa\rho_{vac}, \tag{1.15}$$

where Λ_B is a bare value. The additional term can be considered as contributions originating from vacuum fluctuations, where ρ_{vac} is the constant energy density of the vacuum and $\kappa \equiv \frac{8\pi G}{c^4} \equiv \frac{8\pi}{m_{PI}^2} \equiv \frac{1}{M_{PI}^2}$, where m_{PI} is the Planck mass and M_{PI} the reduced Planck mass. While Λ_{eff} is the observable that we can measure, it can be shown that the contributions from fluctuations can be much larger than the value measured for Λ_{eff}. For example, the Higgs potential contributes with $\rho_{H,vac} \sim \frac{1}{2}\lambda v^4 \approx 1.2\times10^8$ GeV^4 at the potential's minimum. The critical energy density of the universe is $\rho_{crit} \approx 3.7 \times 10^{-47}$GeV^4 [6], which is about 10^{55} times smaller than the Higgs contribution [6]. In a quantum field theory, this very large discrepancy between the contributions as calculated by theory and the experimentally measured value isn't a problem, since the large contributions cancel each other out, meaning only the energy differences between the contributions are relevant. This is not the case when gravity is taken into consideration, as the same contributions now need to be renormalised. For more in-depth overviews of the cosmological constant problem, see [6, 35].

[6] In geometerised units. In SI units $\rho_{crit} \approx 1.878 \times 10^{-29} h^2$ g cm^{-2}, where $h = 0.674$ is the scaling factor for the Hubble expansion rate [3].

If it's determined that $\kappa_\lambda \neq 1$ and that, therefore, the theory of the Higgs potential needs to be expanded, then the Higg's contribution to the cosmological constant would change. This could shed some light on where the discrepancies with the cosmological constant are coming from.

1.3.4 The Naturalness Problem

The theory describing the Higgs boson runs into a predicament: the quantum corrections to the Higgs mass m_H are on a much larger scale than the mass itself. The mass can be described as follows:

$$m_H^2 = m_{bare}^2 + \frac{{y_t}^2}{16\pi^2}\Lambda^2 + \delta\mathcal{O}\left(m_{weak}^2\right), \tag{1.16}$$

where m_{bare} is the Higgs boson mass parameter of the unrenormalised Lagrangian, y_t is the top quark Yukawa coupling, Λ is the cut-off value of momentum in the top quark loop of the Higgs boson self-energy and $\delta\mathcal{O}\left(m_{weak}^2\right)$ are all other quantum corrections at the weak scale [8]. Since the SM seems to valid up until scales such as the Planck mass $\Lambda \sim M_{Pl} \sim 10^{18}$ GeV, then m_{bare} has to not only be very large, but also extremely fine-tuned so that it can cancel out the contribution from $\frac{{y_t}^2}{16\pi^2}\Lambda^2$ to generate a Higgs with $m_H = 125$ GeV. No other parameter in the SM, apart from the cosmological constant, has such a seemingly exact fine-tuning, which can be interpreted as a problem referred to as the *hierarchy problem* [8].

Whether this extreme fine-tuning is an actual problem or not is a topic of debate. While it would seem very unlikely that the contributions line-up the way they do, it is possible, and it can be argued that an issue is being created where none exists. While models exist that could remedy the situation, such as SuperSymmetry or Grand Unified Theories, none have found any experimental success [8, 36].

1.4 The Higgs Boson at the Large Hadron Collider

This section presents an overview of the discovery of the Higgs boson at the LHC and the current state of Higgs self-coupling measurements.

Discovery

The discovery of the SM Higgs boson by the ATLAS and CMS experiments in 2012 [1, 2] at the LHC is one of the most important measurements in modern physics and marks both experiments as successful. Both measurements were performed using data from pp collisions recorded at centre-of-mass energies of $\sqrt{s} = 7$ TeV in 2011 and $\sqrt{s} = 8$ TeV in 2012.
For both experiments, the main production mechanisms were gluon-gluon fusion ($gg \to H$), vector-boson fusion ($qq' \to qq'H$), Higgs-strahlung ($qq' \to WH, ZH$) and for the $H \to \gamma\gamma$ analyses, production via a $t\bar{t}$ pair ($q\bar{q}/gg \to H$).

The measurement at the ATLAS experiment was done by combing searches in the decay channels $H \to ZZ^{(*)} \to 4l$, $H \to WW^{(*)} \to e\nu_e\mu\nu_\mu$ in the 8 TeV data with previous results from $H \to ZZ^{(*)}$, $WW^{(*)}$, $b\bar{b}$ and $\tau^+\tau^-$ in the 7 TeV data, and results from improved analyses of the $H \to ZZ^{(*)} \to 4l$ and $H \to \gamma\gamma$ channels in the 7 TeV data. From this, an observation with a significance of 5.9σ of a neutral boson with mass $126 \pm 0.4(\text{stat.}) \pm 0.4(\text{sys.})$ GeV, which is compatible with the production and decay of the SM Higgs boson, was made [1]. This observation was made in the $H \to ZZ^{(*)} \to 4l$ and $H \to \gamma\gamma$ channels, where an excess of events is observed near 126 GeV (see Fig. 1.3). These are supported by the $H \to WW^{(*)} \to e\nu_e\mu\nu_\mu$ channel. Note that the local p-value discussed in these plots is the probability that the background can produce a fluctuation greater or equal to the excess observed in data [1]. The combined results can be seen in Fig. 1.2.

The measurement at the CMS experiment was done by combing searches in the decay channels $H \to ZZ$, W^+W^-, $\gamma\gamma$, $\tau^+\tau^-$ and $b\bar{b}$ with both $\sqrt{s} = 7$ TeV and $\sqrt{s} = 8$ TeV. From this, an observation with a significance of 5.0σ of a neutral boson with mass 125.3 ± 0.4 (stat.) ± 0.5 (sys.) GeV was made. The excess is most significant in the two decay modes with the best mass resolution, ZZ and $\gamma\gamma$. The combined results and excess can be seen in Fig 1.4

Current State of Higgs Self-Coupling Parameter Measurements

Higgs pair production via Gluon-Gluon Fusion (ggF) can be achieved via self-coupling or a box diagram (see Fig. 1.5). The amplitudes of these two production modes interfere destructively (in the SM), which leads to an overall cross-section of $\sigma^{SM}_{ggF}(pp \to HH) = 31.0^{+2.1}_{-7.2}$ fb at $\sqrt{s} = 13$ TeV. This has been calculated at Next-to-Leading Order (NLO) in Quantum Chromodynamics (QCD) with the measured value of the top-quark mass and corrected to Next-to-Next-to-Leading Order (NNLO) including finite top-quark mass effects [37].

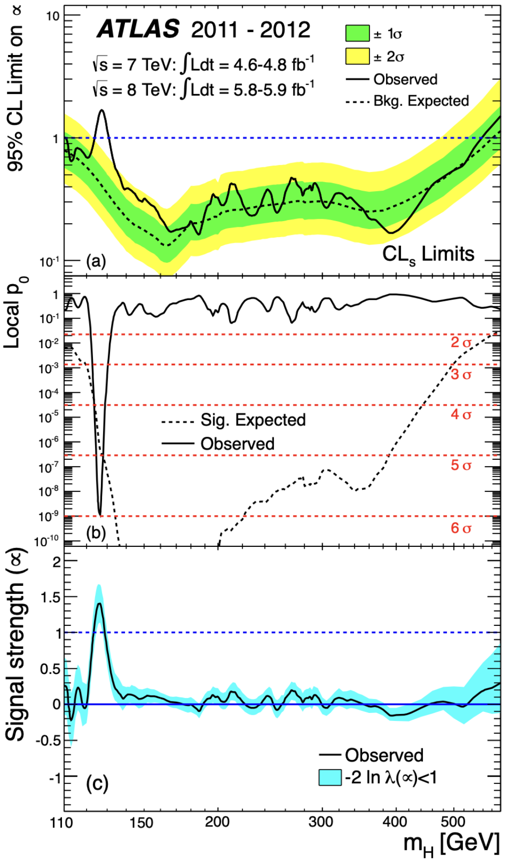


Figure 1.2: Combined search results: **(a)** The observed (solid) 95% Confidence Level (CL) limits on the signal strength and the expectation (dashed) under the background-only hypothesis. **(b)** The observed (solid) p_0 and the expectation (dashed) for a SM Higgs boson signal hypothesis at the given mass. **(c)** The best-fit signal strength. The band indicates the approximate 68% CL interval around the fitted value [1].

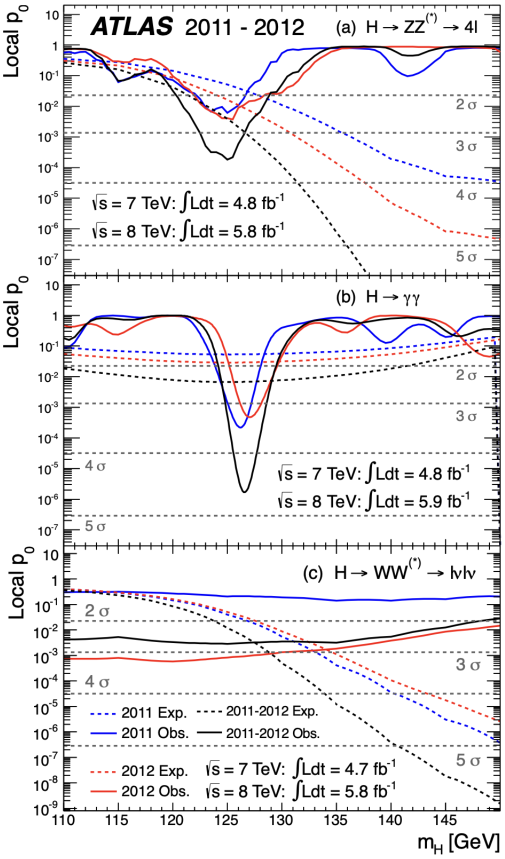
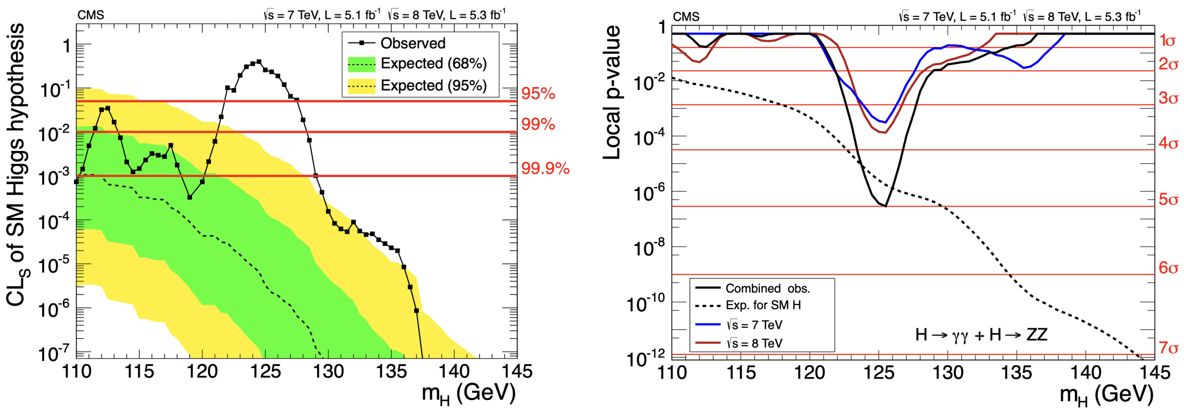


Figure 1.3: The observed local p-value as a function of the hypothesised Higgs boson mass for **(a)** $H \rightarrow ZZ^{(*)} \rightarrow 4l$, **(b)** $H \rightarrow \gamma\gamma$ and **(c)** $H \rightarrow ZZ^{(*)} \rightarrow l\nu l\nu$ channels. The dashed curves show the expected local p_0 under the hypothesis of a SM Higgs boson signal at that mass. The black line represents the combined $\sqrt{s} = 7$ and $\sqrt{s} = 8$ data [1].

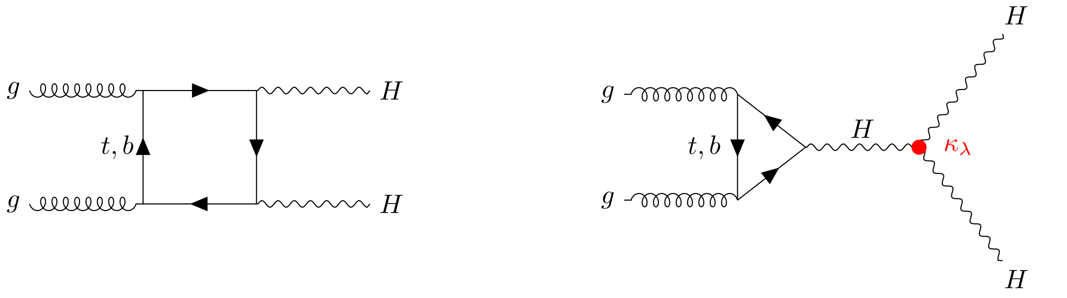
Analyses attempting to measure the Higgs self-coupling parameter have been performed by both the ATLAS [37] and CMS [38] collaborations. In these cases, not just Higgs self-coupling events were taken into consideration, but also general Higgs pair production.

In the ATLAS analysis, studies on Higgs boson pair production to $b\bar{b}b\bar{b}$, $b\bar{b}\tau^+\tau^-$ and $b\bar{b}\gamma\gamma$ decay channels are combined with single Higgs to $\gamma\gamma$, ZZ^*, WW^*, $\tau^+\tau^-$ and $b\bar{b}$ decay channels. The data were recorded at a centre-of-mass energy of $\sqrt{s} = 13$ TeV and correspond to an integrated luminosity of $126 - 139$ fb^{-1}. Additionally, decay channels produced via Vector-Boson Fusion (VBF) with predicted SM cross-section of $\sigma_{VBF}^{SM} = 1.72 \pm 0.4$ fb are analysed.
The value of the signal strength μ_{HH} can be seen in Fig. 1.6b. It is defined as the



(a) **(b)**

Figure 1.4: **(a)** The CL$_S$ (modified frequentist criterion) values for the SM Higgs boson hypothesis. The background-only expectations are represented by their median (dashed) and by the 68% and 95% CL bands [2]. **(b)** The observed local p-value for decay modes with high mass resolutions ($\gamma\gamma$, ZZ). The dashed line shows the expected local p-values for a SM Higgs boson with mass m_H [2].

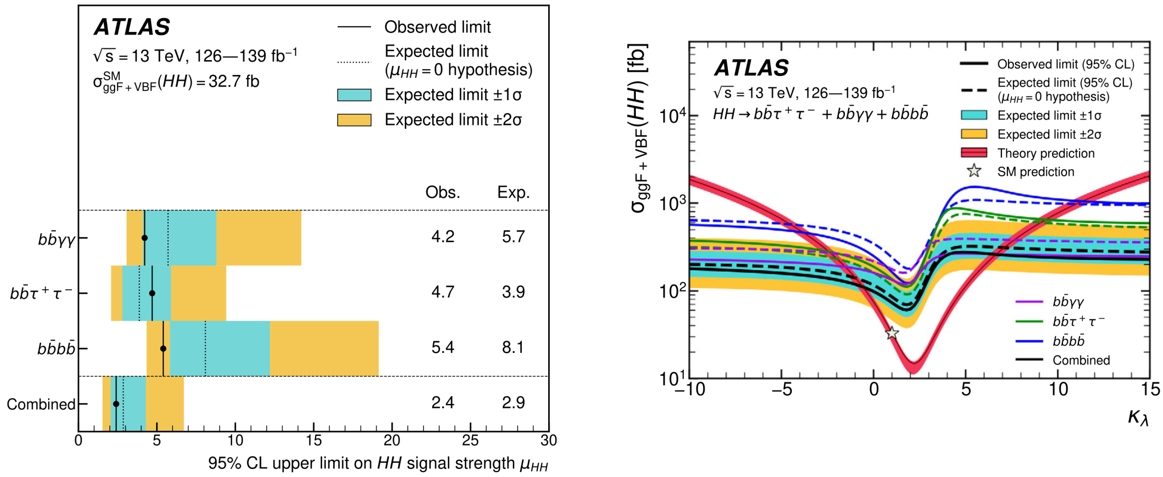


(a) **(b)**

Figure 1.5: Higgs pair production via **(a)** a box diagram and **(b)** self-coupling.

ratio of the Higgs boson pair production cross-section, including only the ggF HH and VBF HH processes, to the SM prediction [37]. As can be seen, the combined observed value is lower than what was expected, apart from the decay to four b quarks. The observed value for κ_λ, which is a multiplier for the SM Higgs self-coupling strength, in relation to the cross-section can be seen in Fig. 1.6a. The observed values, for each individual decay mode and for the combined case, all lie above the SM prediction. These results lead to a 95% CL upper limit of 2.4 on the Higgs pair signal strength and the constraint $-0.6 < \kappa_\lambda < 6.6$, assuming other Higgs boson interactions are as predicted by the SM. When the Higgs pair production decay channels are combined with single Higgs cross-section measurements, the 95% CL limits become tighter, with $-0.4 < \kappa_\lambda < 6.1$ [37].

The CMS analysis studies focus on the Higgs boson pair production to $\gamma\gamma b\bar{b}$ decay channel with a total integrated luminosity of 137 fb^{-1} at a centre-of-mass energy of $\sqrt{s} =$ 13 TeV. For the production, both ggF and VBF are regarded. The $\gamma\gamma b\bar{b}$ final state has a combined branching ratio (BR) of $BR(HH \to \gamma\gamma b\bar{b}) = 2.63 \pm 0.06 \times 10^{-3}$ for a Higgs boson mass of 125 GeV. No significant deviation from the background is observed and the observed 95% CL upper limit for the cross-section $\sigma_{HH} \cdot BR(HH \to \gamma\gamma b\bar{b})$ is 0.67



(a) **(b)**

Figure 1.6: The observed and expected 95% CL upper limits on **(a)** the signal strength for Higgs pair production, where the value $m_H = 125.09$ GeV is assumed when deriving the predicted SM cross-section [37], and **(b)** the combined ggF HH and VBF HH processes as a function of κ_λ. Both plots are taken from [37].

fb. Additionally, a constraint at 95% CL of $-3.3 < \kappa_\lambda < 8.5$ is achieved [38]. In fig. 1.7, in can be seen that the observed values do not overlap with the SM value of $\kappa_\lambda = 1$.

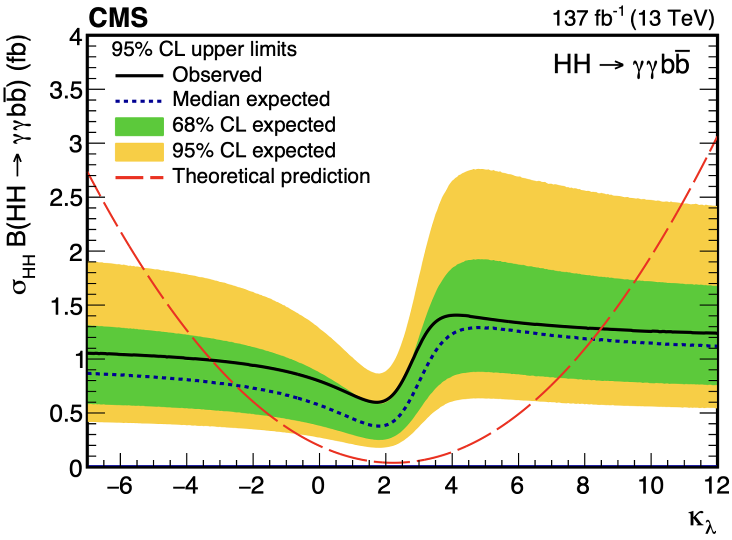


Figure 1.7: The observed and expected 95% CL upper limits on the HH production as a function of κ_λ. Plot taken from [38].

These analyses show that, currently, the capabilities for measuring Higgs self-coupling are not yet available. Instead, the cross-section can only be constrained, which also results in κ_λ not being measurable. This means that more work will have to be done until it is possible to get a better understanding of Higgs physics and how (or if) the Higgs potential plays a role in physics beyond the SM and new analysis methods have to be devised.

1.5 The Matrix Element Method

The Matrix Element Method (MEM) is an analysis tool with which the likelihood that a given event $\mathbf{x}$ stems from a specific theoretical model α can be calculated: $P(\mathbf{x}|\alpha)$. In a particle physics, an *event* usually refers to a single instance of accelerated particles colliding and thereby decaying into other particles, which are measured by a detector. Here, an event $\mathbf{x}$ will specifically refer to a set of experimentally measurable quantities, such as the four-momenta, etc. [39]. In this case, the theoretical model α stands for a specified decay mode. The likelihood $P\left(\mathbf{x}|\alpha\right)$ will be referred to as a weight.

For the calculation of the weight, the event $\mathbf{x}$ is fixed, while all possible momentum configurations $\mathbf{y}$ are integrated over. During the integration, the differential cross section for the decay mode is convolved with Parton Density Functions (PDFs), which describe the probability to find initial-state partons of a given flavour and momentum inside the colliding protons. This is then in-turn convolved with the transfer function $W\left(\mathbf{x},\mathbf{y}\right)$, which describes the probability to reconstruct an event $\mathbf{x}$ in a detector as parton final state $\mathbf{y}$ [40].

The weight of a final state with n_f partons and specific configuration $\mathbf{y}$ is proportional to the differential cross section $\mathrm{d}\sigma_P$ of the corresponding process:

$$\mathrm{d}\sigma_P\left(a_1a_2 \to \mathbf{x};\alpha\right) = \frac{\left(2\pi\right)^4 \left|\mathcal{M}_P\left(a_1a_2 \to \mathbf{y};\alpha\right)\right|^2}{\xi_1\xi_2 s}\mathrm{d}\Phi_{n_f}. \tag{1.17}$$

Here, s is the square of the collider energy, ξ_1 and ξ_2 are the momentum fractions of the colliding particles a_1 and a_2, $\mathcal{M}_P$ is the matrix element for the decay mode and $\mathrm{d}\Phi_{n_f}$ describes the n_f-dimensional phase space over which is integrated [40].

The differential cross section for proton-proton collisions is obtained by convolving the differential cross section in Eq. 1.17 with PDFs:

$$\sigma_P\left(p_1p_2 \to \mathbf{y};\alpha\right) = \int\limits_{\xi_1,\xi_2} \sum_{a_1,a_2} \mathrm{d}\xi_1\mathrm{d}\xi_2\ f^{a_1}_{PDF}(\xi_1)f^{a_2}_{PDF}(\xi_2)\ \mathrm{d}\sigma_P\left(a_1a_2 \to \mathbf{y};\alpha\right). \tag{1.18}$$

The PDFs $f^{a_1}_{PDF}(\xi_1)$ $(f^{a_2}_{PDF}(\xi_2))$ describes the probability to find a parton of given flavour a_1 (a_2) and momentum fraction ξ_1 (ξ_2).

The definition of the transfer function $W(\mathbf{x},\mathbf{y};\boldsymbol{\beta})$ depends parameters describing the detector's response $(\boldsymbol{\beta})$, which includes the parton type. For example, a lepton will have a different transfer function compared to a jet from a quark. As a result, the transfer function can be written as a product of individual single-parton resolution functions [39]:

$$W(\mathbf{x},\mathbf{y};\boldsymbol{\beta}) = \prod_{i=1}^{n} W_i(x^i,y^i;\beta_i). \tag{1.19}$$

Each individual transfer function $W_i(x^i, y^i; \beta_i)$ has to be designed for the particle type it represents, whereby the four-momentum and the particle identification have to be taken into consideration [40].
The differential cross section to observe a given event then becomes

$$\mathrm{d}\sigma_P\left(p_1 p_2 \to \mathbf{x}; \alpha, \boldsymbol{\beta}\right) = \int_y \mathrm{d}\sigma_P(a_1 a_2 \to \mathbf{y}; \alpha) W(\mathbf{x}, \mathbf{y}; \boldsymbol{\beta}). \tag{1.20}$$

By dividing the differential cross section with the total observed cross section, the likelihood $P(\mathbf{x}|\alpha)$ can be obtained [40]:

$$P(\mathbf{x}|\alpha) = \frac{\mathrm{d}\sigma_P\left(p_1 p_2 \to \mathbf{x}; \alpha, \boldsymbol{\beta}\right)}{\sigma_P^{obs}\left(\alpha, \boldsymbol{\beta}\right)}. \tag{1.21}$$

Here, and in the rest of this work, the $\boldsymbol{\beta}$ has been omitted from the description of the likelihood.

1.6 Machine Learning

The goal of using Machine Learning (ML) in artificial intelligence is to recognise patterns, which can be used for data analysis. There are many different types of ML structures and the basic principles will be discussed in this section, which will be restricted to supervised ML with neural networks. Supervised means that the data is labelled, and the network tries to learn the labels.

There are mainly two different kinds of problems that can be tackled with supervised machine learning: classification and regression. In classification problems, the goal is to teach the network to recognise which class the input data represents. A classic example would be teaching a network to be able to differentiate between a picture of a cat or of a dog. In such cases, there can be any number of classes that the neural network has to learn; however, each class is discrete. This means that whether the neural network made a correct assessment or not is definitive.
In regression problems, the goal is to teach the network to estimate a value on a scale. An example would be teaching a network to estimate the price of a house; the difficulty here is that there is no correct or incorrect answer. Instead, how "good" the network's prediction is depends on the challenge that is being tackled.
The base theory of supervised machine learning with neural networks remains the same for classification or regression problems.

The discussion on the theory of machine learning (ML) loosely follows [41].

1.6.1 Feed-forward Deep Neural Networks

The basic building block of a neural network is called a *neuron*. A neuron represents a calculation, where multiple input variables are turned into a single output variable. Neurons are modelled after the neurons of the human brain [42]. Each neuron performs a linear calculation using two types of parameters: a weight $\boldsymbol{\omega}$ and a bias b [7]:

$$y = \boldsymbol{\omega}\mathbf{x} + b \tag{1.22}$$

The weight is a vector with dimension d where each component is assigned to a node of the previous layer: $\boldsymbol{\omega} = (\omega_1, \omega_2, ..., \omega_d)$ (see Fig. 1.9b). The value of each ω_i influences how strongly the output of the associated node in the previous layer will affect the output of the current node. The bias b is a flat term for the entire node.
Once the linear transformation has been performed, an *activation function* $\sigma(y)$ is applied. The goal is to introduce non-linearity to the neuron: without it, the end result would be a combination of linear functions, which would itself be a linear function. This can also mimic the human brain: not all neurons fire all of the time, which can be represented by having an activation function that can result in $\sigma(y) = 0$. These functions can take various forms, with some of the most common being depicted in Fig. 1.8.

[7] The reuse of the term *weight* is not to be confused with its use in the discussion of the matrix element method (Sec.1.5).

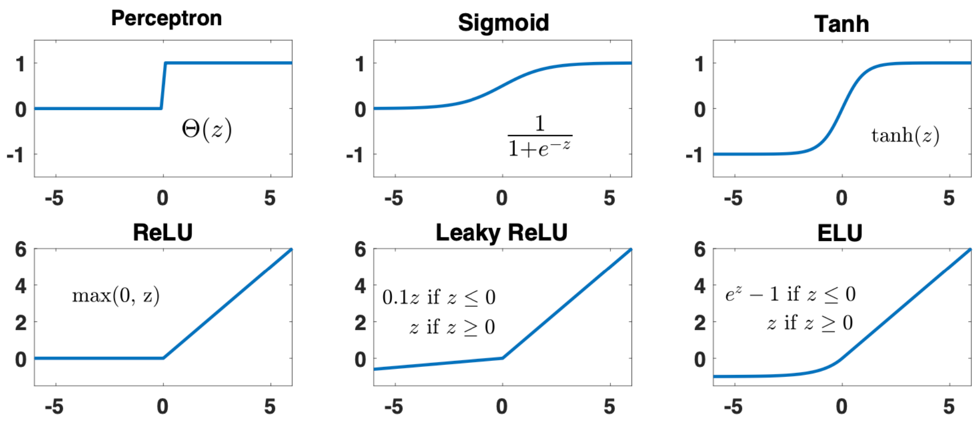


Figure 1.8: Common activation functions for neural networks. Image taken from [41].

The functions in the top row of Fig. 1.8 have gradients that disappear for large positive and negative vales of y — they are referred to as being saturated in these areas. This becomes a problem for backpropagation, which will be discussed later. For this reason, the functions in the lower row are generally preferred in the hidden layers.

A network can consist of any number of layers, where each layer contains multiple neurons (see Fig. 1.9a).

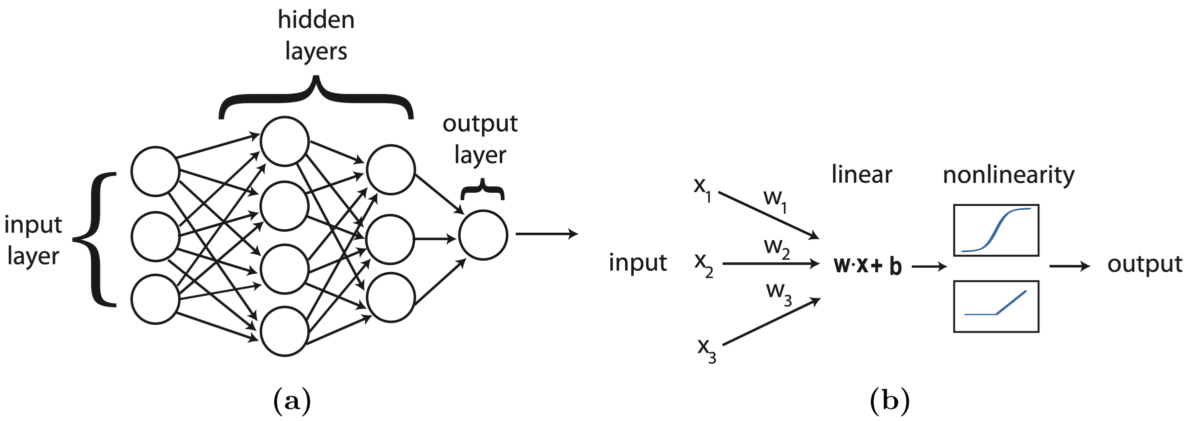


Figure 1.9: Images taken from [41]. **(a)** Basic structure of a neural network. **(b)** Overview of a neuron.

The first layer of neurons is referred to as the input layer, as its input data comes directly from the data set. The final layer is the output layer; the output of this layer is what is used as the result of the neural network. When solving a regression problem, the last layer needs to be able to output a value on a scale. The simplest option is to have the output layer consist of a single neuron without an activation function.
The layers in-between the input and output layers are called *hidden layers*: these layers take the output of neurons of a previous layer as input and their output are used as input for the next layer. How many layers and neurons in each layer are optimal depends on the data set and the challenge that is being tackled. Part of the difficulty of ML is that there are no set rules for choosing an optimal network structure — this must be figured out individually.

1.6.2 The Loss Function

In machine learning, the goal is to tune a model $g(\boldsymbol{\theta})$, with parameters $\boldsymbol{\theta}$, so that it becomes as good as possible at making accurate predictions for a dataset X and then being able to extrapolate. This is achieved by minimising the so-called *loss function* $L\left(X, g(\boldsymbol{\theta})\right)$. How a loss function is defined is up to the user — an example would be the square error

$$L\left(\hat{\boldsymbol{y}}, g(X;\boldsymbol{\theta})\right) = \sum_i \left(\hat{y}_i - g(\mathbf{x}_i, \boldsymbol{\theta})\right)^2, \tag{1.23}$$

where $\hat{\boldsymbol{y}}$ represents the true data and $g(X;\boldsymbol{\theta})$ is the model prediction using a new data point $\mathbf{x}_i$. Iteratively tuning the parameters to minimise the loss function means providing predictions that are closer to the real value.
In linear regression, the loss function can be be written as a sum over all points in a data set [41]:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} l_i(\mathbf{x}_i, \boldsymbol{\theta}). \tag{1.24}$$

Finding the global minimum of the loss function is not necessarily trivial, as the model can have many parameters that influence its shape. Therefore, various techniques and variables have been developed that affect the model's ability to learn, by helping it avoid getting stuck in a local minimum or stranded on a flat region [41].

1.6.3 Gradient Descent

Gradient Descent describes the method of tuning the parameters in the direction for which the gradient of the loss function has the largest negative value, after the parameters have been initialised to some values $\boldsymbol{\theta}_0$:

$$\mathbf{v}_t = \eta_t \nabla_\theta L(\boldsymbol{\theta}_t), \tag{1.25}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t, \tag{1.26}$$

where $\nabla_\theta L(\boldsymbol{\theta}_t)$ is the gradient of $L(\boldsymbol{\theta}_t)$ with regards to $\boldsymbol{\theta}$ at step t. Additionally, a learning rate η_t has been introduced, which controls how large the steps taken in direction of the gradient are [41]. If the learning rate is too small, then the loss function can converge on a local minimum and get stuck. Small learning rates also mean that the model will take longer to train. If it is too large, then the loss function can diverge at every minimum which would cause the model to be come unstable.

Since each data point leads to a new term in the loss function, the calculation of the gradient has to take every term into consideration at the same time. Therefore, the model's parameters are only updated at once the entire gradient has been calculated for the whole dataset, which means that the model doesn't converge very quickly. A slightly different approach is to split the data set: a data set of size n can be divided into *batches* of size M. The gradient is then calculated for the batch and applied to the model's parameters. This means that the gradient is approximated on a subset of the data set, one batch at a time. As a result, the gradient changes to

$$\nabla_\theta L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \nabla_\theta l_i(\mathbf{x}_i, \boldsymbol{\theta}) \to \sum_{i \in B_k} \nabla_\theta l_i(\mathbf{x}_i, \boldsymbol{\theta}). \tag{1.27}$$

where B_k denotes the batch k, with $k = 1, ..., {}^n\!/\!_M$. This method of gradient descend is called Stochastic Gradient Descent (SGD), since the batches consist of randomly chosen data points (which introduces a level of stochasticity). While the accuracy of the gradient does decrease through the introduction of SGD, the model itself converges on the minimum of the loss function quicker [41]. Training a model usually requires many *epochs*, where an epoch refers to a single run through the entire data set. In the following, $\nabla_\theta L(\boldsymbol{\theta})$ will refer to the SGD version.

Momentum

The idea behind momentum is to maintain a downscaled version of the gradient of the previous step when the current step is determined:

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta_t \nabla_\theta L(\boldsymbol{\theta}_t), \tag{1.28}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t. \tag{1.29}$$

where γ is the momentum parameter with $0 \leq \gamma \leq 1$. This allows the gradient to quickly escape flat landscapes and also avoid getting stuck in local minima. An extension to adding momentum is the Nesterov Accelerated Gradient, where the gradient at the current step is calculated using the expected values of the parameters of the next step [41]:

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta_t \nabla_\theta L(\boldsymbol{\theta}_t + \gamma \mathbf{v}_{t-1}), \tag{1.30}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t. \tag{1.31}$$

1.6.4 Overfitting

A general issue with machine learning is the possibility that a model doesn't learn the training data in such a way that it will be able to make accurate predictions with new data — instead, it generates a fit that is finely tuned to the training data, making its predictions on new data unusable. This issue is referred to as *overfitting* and it can occur when a model is too complicated, meaning either the model has too many parameters $\boldsymbol{\theta}$ compared to the size of the data set, or the model puts too much value on certain data points.

One way of combating overfitting is by introducing an L_2 *error*:

$$\hat{\boldsymbol{\omega}}_{Ridge}(\lambda) = \underset{\boldsymbol{\omega}\in\mathbb{R}^d}{\arg\min} \left(||\hat{\mathbf{y}} - g(\mathbf{X}, \omega)||_2^2 + \lambda\, ||\boldsymbol{\omega}||_2^2 \right) , \tag{1.32}$$

For the least squares loss function, this is referred to as *Ridge Regression* [41]. The loss function receives a punishment if individual weights become large in magnitude, i.e. if the model is relying too heavily on a data point, then it will be punished. This will lead to self-correction when weights become too large. The weight decay λ is a hyperparameter of the model, meaning it is not optimised by gradient descent.

It is also possible to actively monitor the model as it is learning, and to stop it early as soon as signs of overfitting become visible.

Another way to keep track of overfitting is to compare the loss of the model after it ran on training data compared to when it ran on the validation data. If no overfitting is taking place, then the values will converge on one another; if there is overfitting, then the loss for the training data will be (significantly) lower than for the validation data.

1.6.5 Standardising the Input Data

A neural network can only be as good as its input data allows it to be. Therefore, it's possible to support the network by restructuring the data in such a way that it eases the network's ability to learn. To do this, a data point is standardised by subtracting the mean of each input variable from the value of the respective variable, and then dividing each variable by its standard deviation:

$$\mathbf{x}_i^{stand} = \frac{\mathbf{x_i} - \bar{\mathbf{x}}_i}{\sigma_{\mathbf{x}_i}}, \tag{1.33}$$

where $\bar{\mathbf{x}}_i$ and $\sigma_{\mathbf{x}_i}$ are the mean value and standard deviation of the input variable x_i. This reshapes each variable to have a standard deviation of 1 around the value 0. By restricting the input values to a similar range, the neural network will have an easier time learning the distribution of the input variables, since the weights aren't initialised to reflect different ranges and scales in the input data.

For regression problems, restructuring the training values of the output variable can be particularly helpful if the values are spread across a large scale. It's easier to learn the boundaries of the possible output values when they are more tightly restricted. To evaluate the network, the predicted values will then have to be appropriately transformed back to their original structure.

1.6.6 Backpropagation

Backpropagation describes the process of updating the parameters $\boldsymbol{\theta}$ of the model, which in this case are the weights $\boldsymbol{\omega}$ and biases b. It relies on using the chain rule, since all layers are connected to one another:

$$a_j^m = \sigma\left(\sum_k \omega_{jk}^m a_k^{m-1} + b_j^m \right) = \sigma(y_j^m). \tag{1.34}$$

This describes the case where the network has M layers with $m = 1, ..., M$. ω^m_{jk} describes the weight that couples the k-th neuron on layer $m-1$ to the j-th neuron on layer m. The bias of the j-th neuron on layer m is b^m_j.
The loss function L depends on the output of the network, which means the output of layer M. This, in turn, is dependent on outputs a^{M-1}_j of the previous layer, which recursively works its way up to the input layer. To understand how the weights and bias should be adjusted at each node, a few definitions are required. First, the error of the j-th neuron on the m-th layer is defined as the change of the loss function in regards to the weighted input y^m_j:

$$\Delta^m_j = \frac{\partial L}{\partial y^m_j} = \frac{\partial L}{\partial a^m_j}\sigma'(y^m_j). \tag{1.35}$$

Here, $\sigma'(y^m_j)$ is the derivative of the activation function $\sigma(\cdot)$ evaluated at the value for y^m_j. For the output layer $m = M$, the derivatives of the loss function and the activation function $\sigma(y)$ need to be known. Second, the error Δ^m_j can also be interpreted as the change of the function with regards to the bias b^m_j:

$$\Delta^m_j = \frac{\partial L}{\partial y^m_j} = \frac{\partial L}{\partial b^m_j}\frac{\partial b^m_j}{\partial y^m_j} = \frac{\partial L}{\partial b^m_j}, \tag{1.36}$$

using $\frac{\partial b^m_j}{\partial y^m_j} = 1$. Third, the chain rule is used to connect the error of layer m with that of $m+1$:

$$\Delta^m_j = \frac{\partial L}{\partial y^m_j} = \sum_k \frac{\partial L}{\partial y^{m+1}_k}\frac{\partial y^{m+1}_k}{\partial y^m_j} = \sum_k \Delta^{m+1}_j \frac{\partial y^{m+1}_k}{\partial y^m_j} = \left(\sum_k \Delta^{m+1}_j \omega^{m+1}_{kj}\right)\sigma'(y^m_j). \tag{1.37}$$

Last, the derivative of the loss function in regards to the weight ω^m_{jk} is

$$\frac{\partial L}{\partial \omega^m_{kj}} = \frac{\partial L}{\partial y^m_j}\frac{y^m_j}{\partial \omega^m_{kj}} = \Delta^m_j a^{m-1}_k. \tag{1.38}$$

Together, these equations can be used to calculate the gradient for all parameters on every node. The steps are as follows:
Once the network has run through a batch, where-by the activations a^m_j are calculated for each neuron, the loss function for that batch is calculated. Then, using Eq. 1.35, the error of the output layer is calculated. Following this, by using Eq. 1.37, the errors Δ^m_j are calculated for each layer and neuron[8]. Finally, using Eq. 1.36 and Eq. 1.38, the errors for each ω^m_{kj} and b^m_j are calculated and the weights and biases are adjusted accordingly [41] (see Sec. 1.6.3).

It's at this point that problems can occur: if the gradients are too large, then the changes to the parameters will explode, since the parameters will receive large changes with every update, which will cause the network to become unstable. On the other

[8] This step is referred to as backpropagation.

side, if the gradients are (vanishingly) small, then the backpropagation will get "stuck" halfway through and the parameters of the beginning layers won't be updated. This will lead to the network not learning. There are, however, methods for combating these problems, e.g. using an activation function that doesn't have a vanishing gradient (e.g. bottom row of Fig. 1.8).

Dropout

Another method to help prevent overfitting is known as *dropout* [43] . The idea is to randomly, temporarily remove neurons from a network, along with their ingoing and outgoing connections (see Fig. 1.10). The dropout is applied to each neuron of a chosen layer individually, with probability p. Neurons that are dropped pass an output value of 0. This results in a collection of thinned-out networks (one for each batch), where each one is slightly different. Doing this on its own would then lead to an issue when running the network on test data, as it would require running all the networks and then averaging their predictions. Instead, an approximate averaging method is used, where the weights are scaled with $\frac{1}{1-p}$ (PyTorch implementation [44]). For the averaging, any training case which does not use a parameter (because it has been dropped) contributes a gradient of zero.

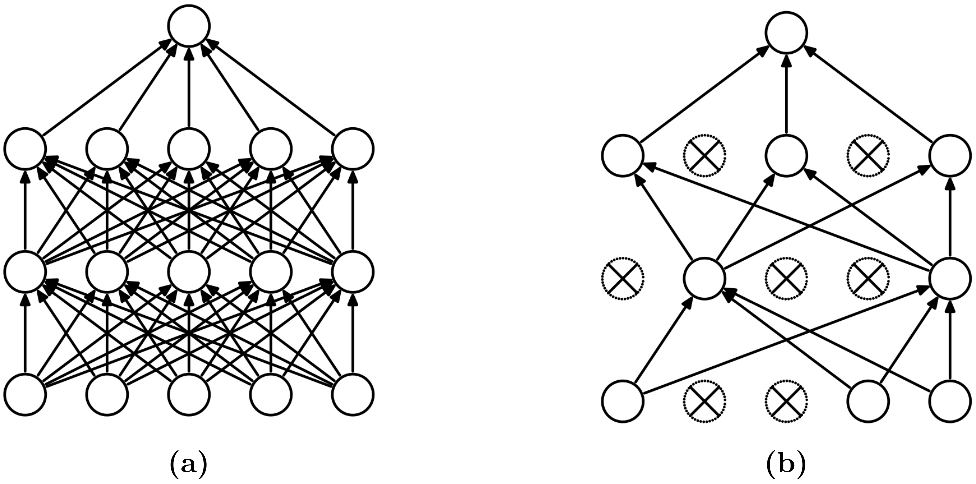


Figure 1.10: Images taken from [43]. **(a)** Fast-forward neural network with two hidden layers. **(b)** Same network, but with dropout applied to each layer (crossed units have been dropped).

Residuals

The idea behind residuals [45] is to pass the output variables of a layer to another future layer other than the one directly following it (see Fig. 1.11). This method was developed to combat an issue where the accuracy of deep neural networks would suddenly start to diminish. It could be shown that this was not a result of overfitting and adding more layers would lead to a higher training error [45]. In the simplest case, the residuals are passed via an identity mapping, meaning the values are passed unchanged. This method can increase accuracy without having to introduce additional hyperparameters or increase computational complexity and can also be used to combat vanishing gradients.

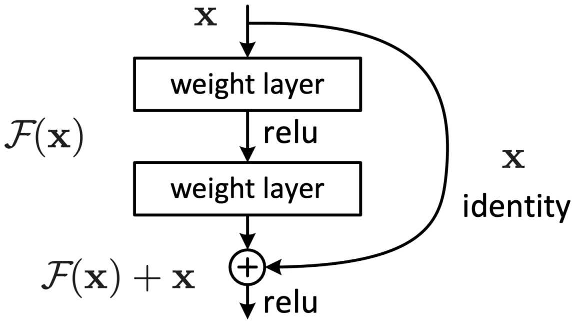


Figure 1.11: Residuals being passed forward to a future hidden layer. Plot taken from [45].

1.6.7 Convolutional Neural Networks

Convolutional Neural Network (CNNs) were designed to take advantage of locality and translational invariance [41]; in other words, recognising a local pattern (locality) anywhere in a data set (translational invariance). This makes CNNs particularly good at image and audio recognition, where smaller structures, such as a face or a word, need to be identified within a larger structure, such as a photo or an audio file.

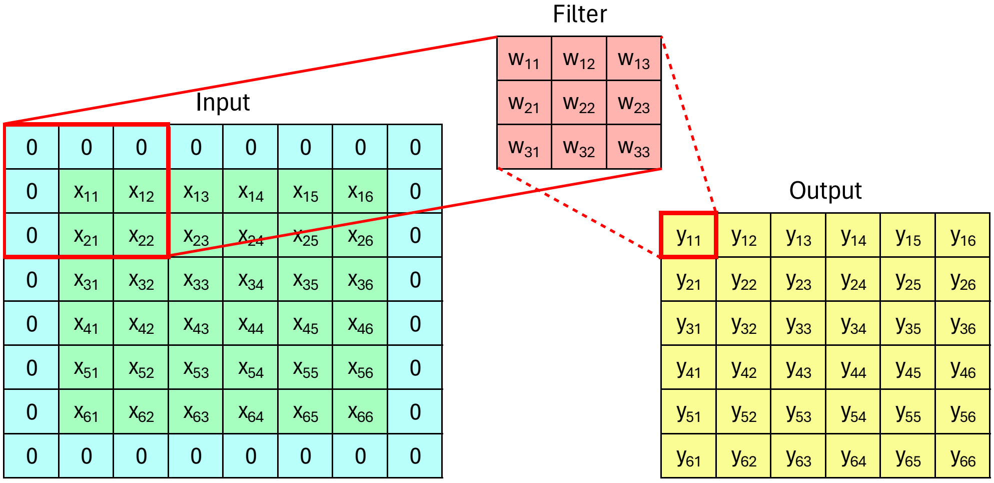
Convolutional Layer

CNNs mostly consist of two types of layers: convolutional layers and pooling layers. In a convolutional layer, the input data, which has the form of a matrix with width W, height H and depth D, is convolved[9] with a filter; the output of which is also a matrix (see Fig. 1.12). The number of variables in a data point is represented by its depth and each variable requires its own *channel*. For example, an $N \times M$ image with pixels containing RGB data would require three channels, one for each colour, and the input matrix would therefore be of size $N \times M \times 3$.
Each channel is convolved with singular matrix, called a *kernel*. Usually, the kernel is of size $K \times K$, where K is the *kernel size*. The result of these convolutions is then summed and the biases associated with the channels are added at the end. This results in an output matrix. A convolutional layer can have multiple kernels, where each kernel results in its own output matrix — this means that the number of output channels that a convolutional layer has is defined by the number of kernels in that layer.
For example, in Fig. 1.12, one step of the convolution between an input matrix with $D = 1$ and a filter with one kernel is depicted. The result of this step is $y_{11} = x_{11}\omega_{22} + x_{12}\omega_{23} + x_{21}\omega_{32} + x_{22}\omega_{33} + b$. If $D = 3$, then y_{11} would consist of the sum of the convolutions of all three channels with the kernel and their biases. If the filter had more than one kernel, then there would be more than one output matrix.

[9] This is referring to the mathematical operation.

There are a couple of aspects with which the output of the convolution can be configured. First, the filters don't have to have the same height and width as the matrix that they are being convolved with. If a filter is larger than $1 \times 1 \times D$, then the resulting matrix will have a smaller height and width than the input matrix; however, by introducing padding (additional rows and columns of 0s, see Fig. 1.12), the output matrix (feature map) can be the same size as the input matrix. Also, the size if the filter influences the number of weights in that convolution layer. In Fig. 1.12, a filter of size $3 \times 3 \times 1$ is depicted.

Another variable of the convolution is the stride s: this defines how many cells over the filter moves during each step of the convolution. In Fig. 1.12, the stride in both width and height is $s_W,\ s_H = 1$, leading to a feature map of the same size as the input (due to one row/column of padding). If $s_W = 2$ and $s_H = 1$, then the feature map would have the size $6 \times 3 \times 1$.
The feature map can then have an activation function applied to it, which will affect the matrix elements individually.



w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

0	0	0	0	0	0	0	0
0	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	0
0	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	0
0	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	0
0	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	0
0	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	0
0	x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	0
0	0	0	0	0	0	0	0

y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}
y_{21}	y_{22}	y_{23}	y_{24}	y_{25}	y_{26}
y_{31}	y_{32}	y_{33}	y_{34}	y_{35}	y_{36}
y_{41}	y_{42}	y_{43}	y_{44}	y_{45}	y_{46}
y_{51}	y_{52}	y_{53}	y_{54}	y_{55}	y_{56}
y_{61}	y_{62}	y_{63}	y_{64}	y_{65}	y_{66}

Figure 1.12: Diagram for a convolutional layer of a Convolutional Neural Network (CNN) with $W,\ H = 6$ and $D = 1$. Input layer (green) with padding (blue) is convolved with a filter (red), resulting in an output matrix of the same size (yellow).

Pooling Layer

The pooling layer reduces the resolution of the feature map. This is commonly achieved by applying a max-pooling function [46], which simply returns the largest entry from the inputs, e.g.:

$$max\left(\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\right) = 7 \qquad (1.39)$$

This results in the first layers learning low-level features such as edges and curves, while later layers learn more abstract features [46] (see Fig. 1.13).

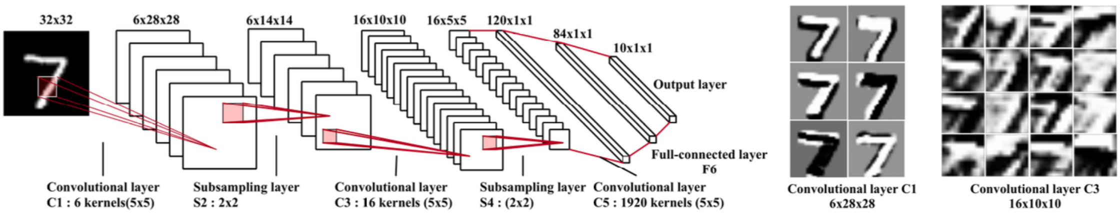


Figure 1.13: CNN with depiction of the feature maps of the first and second convolutional layer, which have a pooling layer in-between them. Image taken from [46].

Fully Connected Layer

Generally, the final layer (or final few layers) of a CNN is a fully connected layer, which outputs the prediction of the neural network. It functions in the same way as the final layer of a feed-forward neural network, where all the inputs of the second-to-last layer are fed to a single neuron without an activation function[10], which then gives the network's predicted value. For this, the output of the layer before the fully connected layer has to be flattened, as it can contain multiple dimensions.

1.6.8 Evaluating a Neural Network

Once a trained neural network has been tested on validation data, its output values have to be evaluated. So far, the loss function has been used as a tool for training the neural network to be as optimal as possible; however, the absolute value of the loss function is difficult to interpret. This means that the loss function can't easily be used to judge whether the accuracy of the network is good enough to solve the challenge it has been tasked with.
This can be addressed by introducing the R^2-parameter[11] [47]:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{1.40}$$

where y_i and $\hat{y}_i$ are the predicted value and the true value for data point i, respectively, and $\bar{y}$ is the mean output value of the true data. This sets $R^2 \leq 1$.
The best-case scenario is $R^2 = 1$, since this means that the predicted values perfectly match the true values. If $R^2 = 0$, then this can be interpreted as the neural network not having learnt very much; it is as good as always predicting the mean value of the data set. A neural network that effectively hasn't learnt[12] will have $R^2 < 0$. Realistically, a neural network that has learned something will have $0 < R^2 < 1$.
An important thing to note about the R^2-parameter is whether a given value is considered good or not is arbitrary. Whether the state of the trained neural network can be considered good or not depends on the problem that is trying to be solved and the data set being used. Therefore, there are no standardised values for the R^2-parameter that should be aimed for. However, it can be used to compare different neural networks

[10] In the case of a regression problem.

[11] The definition for the R^2-parameter can differ depending on the source.

[12] This could be due to multiple factors, e.g. bad weight initialisation, having a structure that is too simple, vanishing gradients, etc.

that are being tested on the same problem, using the same data set, which is useful when there are many different model configurations that are being examined.

Chapter 2

Software and Hardware

In this chapter, the main pieces of software used in this work are presented. They have been grouped by their usage in the analysis: either for generating the simulated event data, for applying the MEM to the data, or for ML. At the end, a brief description of the hardware that was used is given.

2.1 Software

2.1.1 Event Generation

All data that is used in this analysis has been simulated. The first step uses MadGraph 5 [48, 49] to simulate the hard scattering of the initial protons[1]. The showering is then handled by Pythia6[2]. The jets are reconstructed using the package FastJet 3.3.0 [51, 52]. The plotting variables are stored in a ROOT [53] file and are later read-in via Uproot [54] and plotted using Seaborn [55].

2.1.2 Matrix Element Method

MoMEMta [56, 39] is a C++ software package that has been developed to compute the convolution integrations at the core of the MEM. It's built on the same principles as MadWeight [57] in regards to the parametrisation of the phase space, but allows for more freedom for the user in terms of modularity [56]. This is achieved by having every term in the weight calculation (see. Eq. 1.21) be represented by its own configurable module [56].

[1] The following configurations are used:

- $E_{beam} = 2$ TeV
- $b_{w,cutoff} = 33$ to allow off-shell W production and decay in $H \to WW$
- PDF set 90400 from LHAPDF6 [50]
- $min(p_T, jet) = 20$ GeV, $min(p_T, lep) = 10$ GeV

[2] textscPythia tune 340 `AMBT1` was used: `ATLAS_CONF_2010_031`

The transfer functions are assumed to factorise into direction- and momentum-dependent terms for each measured final-state particle. For most applications, the angular variables are assumed to be perfectly reconstructed, and the transfer function is described by a Dirac delta function in such cases. These assumptions may not be valid when final-state objects are close to each other [56].
Unresolved degrees of freedom can appear in the form of invisible particles or objects that are unreconstructed objects. They can be removed by enforcing four-momentum conservation in the initial and final states; additionally, mass constraints on narrow resonances or the measured total transverse momentum can reduce the degrees of freedom further [56].

In most cases, the computation of the weights requires the evaluation of multidimensional integrals via adaptive MC (MC) techniques. A finite number of analytical transformations, called *Blocks*, are used to map the structures in the integrand in an efficient way. The transformations are non-linear, which leads to multiple solutions, for each of which the matrix element, the transfer function and PDFs need to be evaluated [56].
There are two types of Blocks: *Main Blocks* and *Secondary Blocks*. Main Blocks describe changes of variables that enforce conservation of momentum between the initial and final states. Secondary Blocks also do this, but they describe changes that do not remove any degrees of freedom [56]. These Blocks are implemented as modules that can be chained together to describe the chosen physical model. They also compute the jacobian factor associated with the variable changes, which are then multiplied with the integrand [56]. The Blocks that are already implemented in MoMEMta can be seen in Tab. 2.1a and Tab. 2.1b.

MoMEMta handles the assignment of reconstructed final-state objects to partons in the matrix element by computing an average weight over all possible permutations. This is done by introducing a new dimension for the integrand phase space, which decides which assignment should be used for computing the integral. The weight calculation is performed with a set number of evaluations. Therefore, by using an adaptive integration algorithm, which concentrates on the permutations that yield the largest contribution, the accuracy of the calculation is increased [56].
The PDFs are obtained from LHAPDF6 [50] and the integration is done using the CUBA library [58], which contains the `Vegas` algorithm [59] for multidimensional integration, amongst others. The matrix element is constructed in MadWeight and can be fed to MoMEMta via a plug-in [56].

The computation of the weight requires the evaluation of multidimensional integrals via adaptive MC techniques [56]. Points in the phase space, which define variables such as the masses of the W and Higgs bosons, are probed and the weight is iteratively calculated. This process continues until either the weight converges or until a limit on the number of iterations has been reached.

2.1.3 Machine Learning

PYTORCH [44] is a ML library built for PYTHON. It focuses on implementing an easy-to-use interface without sacrificing much in terms of speed and efficiency. Most of its core is written in C++ and can be executed on either Central Processing Unit (CPUs) or GPUs. Operations on GPUs are implemented using CUDA [60], which allows for Python code to be executed onCPUs while tensor operators are run on GPUs.
The weights and biases of the neural networks are visualised using TENSORBOARD [61].

Main block	Topology	Removed variables	New variables
A	$(q_1, q_2) \to p_1 + p_2$	$q_1, q_2, \lvert p_1\rvert, \lvert p_2\rvert$	
B	$(q_1, q_2) \to s_{12}(\to p_1 + p_2)$	q_1, q_2, p_1	s_{12}
C	$(q_1, q_2) \to s_{123} \to p_3 + s_{12}(\to p_1 + p_2)$	$q_1, q_2, p_1, \lvert p_3\rvert$	s_{12}, s_{123}
D	$(q_1, q_2) \to s_{134}(\to p_4 + s_{13}(\to p_1 + p_3)) + s_{256}(\to p_6 + s_{25}(\to p_2 + p_5))$	q_1, q_2, p_1, p_2	$s_{13}, s_{134}, s_{25}, s_{256}$
E	$(q_1, q_2) \to (s_{1234}, y) \to s_{13}(\to p_1 + p_3) + s_{24}(p_2 + p_4)$	q_1, q_2, p_1, p_2	$s_{1234}, y, s_{13}, s_{24}$
F	$(q_1, q_2) \to s_{13}(\to p_1 + p_3) + s_{24}(\to p_2 + p_4)$	p_1, p_2	q_1, q_2, s_{13}, s_{24}
G	$(q_1, q_2) \to s_{13}(\to p_1 + p_2) + s_{34}(\to p_3 + p_4)$	$q_1, q_2, \lvert p_1\rvert, \lvert p_2\rvert, \lvert p_3\rvert, \lvert p_4\rvert$	s_{12}, s_{34}

(a)

Secondary block	Topology	Removed variables	New variables
A	$s_{1234} \to (s_{123} \to s_{12}(\to p_1 + p_2) + p_3) + p_4$	p_1	$s_{1234}, s_{123}, s_{12}$
B	$s_{123} \to s_{12}(\to p_1 + p_2) + p_3$	$\lvert p_1\rvert, \theta_1$	s_{12}, s_{123}
C/D	$s_{12} \to p_1 + p_2$	$\lvert p_1\rvert$	s_{12}
E	$s_{123} \to s_{12}(\to p_1 + p_2) + p_3$	$\lvert p_1\rvert, \lvert p_3\rvert$	s_{12}, s_{123}

(b)

Table 2.1: (a) Main blocks of MoMEMta. Each block performs a specific change of integration variables, and removes four degrees of freedom by enforcing momentum conservation between the initial and final states. The third and four columns show the integration variables that are either removed or added. q_i denotes the Bjorken fractions of the initial-state particles, and p_i the four-momentum of the finial-state particles (in polar coordinates $|p_i|, \theta_i$ and ϕ_i). Off-shell intermediate particles are written as $s_{i...j}(\to p_i...p_j)$, where $p_{i...j}$ are the final-state particles to which they decay. The variable $s_{i...j}$ is defined as $s_{i...j} = (p_i + ... + p_j)^2$. Removing a particle, p_i means removing all three degrees of freedom associated with that particle. Variables that are not explicitly removed are understood to remain present as in the standard polar phase-space parametrisation. Similarly, additional final-state particles not mentioned in the block topology are allowed [56]. **(b)** Secondary blocks of MoMEMta. Each block performs a specific change of integration variables, acting exclusively on final-state particles [56].

2.2 Hardware

Due to the large number of events that need to be processed, the calculation of the event weights with MoMEMta is split into smaller jobs and run in parallel on a local Slurm [62] cluster. The cluster consists of 49 PC nodes with various CPUs and GPUs, meaning that the time needed for the calculation of the weights varies, depending on the node.

Chapter 3

Analysis

In this chapter, the analysis of this work will be presented. It is based on the theory that has been discussed in Ch. 1 and uses the software presented in Ch. 2.

3.1 Overview

The main objective of this work is to reduce the amount of computing resources required for an analysis using the MEM by training a neural network to be able to predict event weights accurately. This requires generating simulated training data, where it is known which events are signal events and which are background. The event weights are then computed for the training data, with which the neural network is then trained. The result is a trained network that can take input data from an event and provide an accurate weight, which can be used for analysis purposes.

In this work, as a proof-of-concept, the first steps for a search for di-Higgs events using the MEM is assisted with neural networks. This consists of separating di-Higgs events containing either Higgs self-coupling or lowest level Higgs pair-production from background events containing Z bosons.

3.2 Decay Models

The signal decay mode that has been chosen for Higgs self-coupling can be seen in Fig. 3.1a. This decay mode contains an off-shell Higgs boson that decays into two on-shell Higgs bosons $H^* \to HH$. One of the on-shell Higgs bosons then decays into two b quarks, where the other decays into an off-shell W^* and an on-shell W boson. Both the case where $W^* \to q\bar{q}$ and $W \to l\nu$, and vice versa, are allowed.
This was chosen as the main signal model as it contains a clean detector signal while also maximising the branching ratio for events with Higgs self-coupling. The Higgs boson has a strong coupling to b quarks, with the branching ratio of $BR(H \to b\bar{b}) = (53 \pm 8)\%$ [3]. However, having both Higgs bosons decay to b quarks would have a low detection efficiency, since due to b-tagging, b quarks have a detection efficiency of $\approx 70 - 75\%$ [63]. This makes the likelihood of correctly identifying all b quarks drops

considerably with every additional b quark.
Having a lepton in the final state helps distinguish signal events from a large amount of background events, as hadron colliders produce a high number of purely hadronic events. Leptons also have a high detection efficiency (see [64], [65]), meaning there is a low chance a lepton will be misidentified. However, the reason both W bosons haven't been chosen to decay leptonically is that the di-lepton case has a small branching ratio: $BR(W \to l\bar{\nu}_l) = (10.86 \pm 0.09)\%$ [3]. Additionally, the two neutrinos can't be separated from each other since they both appear as missing energy and missing transverse momentum in the detector. This can make distinguishing the neutrinos difficult. For these multiple reasons, it was chosen to have one W boson decay hadronically.

In events with two Higgs bosons, with $BR(H \to b\bar{b}) = (53\pm8)\%$, $BR(H \to W^+W^-) = (25.7\pm2.5)\%$, $BR(W \to q\bar{q}) = (67.41\pm0.27)\%$ and $BR(W \to l\bar{\nu}_l) = (10.86\pm0.09)\%$ [3], the resulting branching ratio for the chosen final-state is $\approx 0.36\% - 0.79\%$.
In events with a Z boson and a Higgs boson, with $BR(Z \to b\bar{b}) = (15.12 \pm 0.05)\%$ [3], the resulting branching ratio for the chosen final-state is $\approx 0.117\%$–0.125%.

The combination of the decay modes for Higgs pair-production and for Higgs self-coupling constitute the signal data and can be seen in Fig. 3.1. The hard-scattering processes have been automatically selected by MadGraph [48], whereby the lowest-order Feynman diagrams possible for the processes are chosen. MadGraph always generates the mode in Fig. 3.1b together with the mode in Fig. 3.1a using an effective coupling between Higgs and gluons (indicated by the black point). The box-diagram via t/b quark loops isn't selected, since it's of higher order and involves an explicit calculation of the loop. Moreover, the depicted Feynman diagrams represent the HH production cross section, eventually taking higher-order effects into account by k-factors, while destructive interference with the box-diagram becomes prominent only at energies close to the HH production threshold. However, this analysis focuses on separating HH final-states from HZ final-states, whereby interferences between various HH Feynman diagrams close to the HH production threshold can be ignored. A separate analysis is being done to study how the interference between self-coupling and the box-diagram via t/b quark loops impacts the data [66].
While the hard scattering process that generates the on-shell Higgs bosons is different in each decay mode, the showering of the Higgs bosons is the same. Signal events will also be referred to as HH events.
The only background decay mode considered in this analysis is depicted in Fig. 3.2, where an off-shell Z boson (Z^*) emits an on-shell Higgs boson. Unlike the signal decay modes, this decay mode is produced via quark-antiquark annihilation as opposed to ggF. This is because MadGraph chooses the lowest order Feynman diagrams possible for the processes and for the Z boson events this results in a tree-level diagram.
The showering of the background decay mode is the same as for the signal modes. Background events will also be referred to as HZ events. A common background that would need to be considered in a full analysis for detecting Higgs pair-production, or specifically Higgs self-coupling, are events containing $t\bar{t}$. Top quarks can decay via $t \to bW$, meaning events with top-quark pairs can create the same final-state as Higgs pair-production. However, due to the limited scope of this analysis, such background events are not taken into consideration in this work.

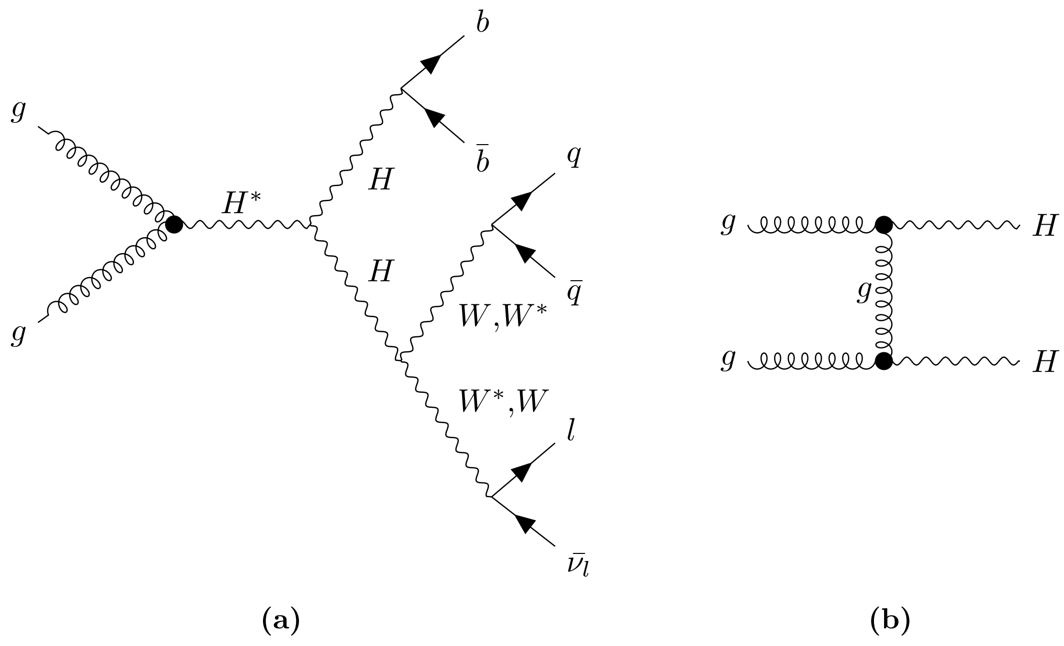


Figure 3.1: Lowest order signal decay modes with **(a)** Higgs self-coupling (hard scattering and shower) and **(b)** Higgs pair-production (hard scattering), in which the Higgs bosons decay in the same way as in the case of self-coupling. The black points represent effective couplings between Higgs and gluons. A $*$ denotes an off-shell particle.

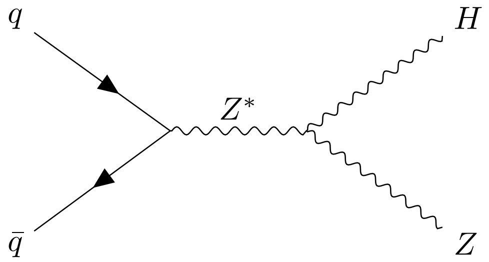


Figure 3.2: Lowest-order background event in which a off-shell Z boson (Z^*) emits a Higgs boson. The Z^* boson is produced via quark-antiquark annihilation.

3.3 Event Generation and Reconstruction

The following section describes how the simulated events are generated and the final-state particle are reconstructed into jets[1]. Additionally, selection criteria that are used to filter events after reconstruction are presented.

Two sets of hard scattering events are generated using MadGraph 5 [48], one for the di-Higgs modes and one for $Z^* \to ZH$, where each set consists of 10^7 events. In these events, both the Higgs and Z bosons are limited to decay to either b quarks or W bosons. This means that not every generated event contains scattering according to either the signal or background modes and need to be filtered out.

[1] A jet is a collection of final-state particles, described by a cone, that is treated as a single particle. This is necessary for describing quarks and gluons, since they hadronise. The cone is described by a radius parameter R. "Reconstruction" describes the process of creating the jets.

Particle Recognition

For real data measured in a detector, highly complicated algorithms are used to assign particles to jets. It's therefore beneficial to use these algorithms when working on an analysis with simulated data to get a better understanding of how the analysis would work with real data. However, the analysis in this work does not have access to such algorithms. For this reason, the full information that the simulation provides is used to assist the assignment of jets to particles. Full information, in this sense, means knowing the four-momentum of all intermediate and final-state particles, and all present decay chains.

The PYTHIA 6 function `TPythia6::Pylist(2)` can be used to visualise all information in an event (see Tab. 3.1). This returns a list that contains hundreds of lines, where each line corresponds to a particle in the event. The columns are to be understood as follows [67]:

- *I* gives each line an index number.
- *particle/jet* describes the particle type for that index. A particle surrounded by exclamation marks indicates that that particle has been created by another generator and has been read into PYTHIA. A particle with parenthesis around it, e.g. (W^+), is a particle that has fragmented.
- In some cases, the *particle/jet* can also have an `A`, `I` or `V`. This indicates the beginning, `A`, or end, `V`, of a string/cluster parton system.
- *K(I,1)* and *K(I,2)* describe the state of the particle and the Particle Identification Number (Particle ID) respectively. The state that is relevant for assigning the partons to jets is `1`, as this describes final-state particles. The Particle ID describes a particle via its MC numbering scheme [3], or, if a particle is made of quarks, its constituents.
- *K(I,3), K(I,4), K(I,5)* refer to the indices of the mother particle and the two daughter particles, respectively. This makes it possible to follow decay chains.
- *P(I,1)–P(I,3), P(I,4), P(I,5)* describe the three-momentum, the energy and the mass of that index's particle.

PYTHIA provides the function `GetK(I,X)`, with $X \in [1, 5]$, which retrieves the appropriate information for the particle at index I. Using this, a script has been written that finds the final-state particles in the event and is able to associate them with the lepton or quark that they come from. This approach is done top-down, e.g. first a Higgs boson is found, then the b quarks that the Higgs boson decayed into, then the final-state particles that come from the hadronisation of the b quark. Once jets have been reconstructed (see Sec: 3.3), it will be possible to know jet contains partons from which original particle. This removes the possibility of assigning a jet to an original particle incorrectly.
In cases where colour reconnection occurs during the hadronisation process, particles are described as a string system, which is indicated by a line having `(string)` in the *particle/jet* column. In such cases, the angular differences of the three-momentum of

I	particle/jet	K(I,1)	K(I,2)	K(I,3)	K(I,4)	K(I,5)	P(I,1)	P(I,2)	P(I,3)	P(I,4)	P(I,5)
1	!p+!	21	2212	0	0	0	0.00000	0.00000	6999.99994	7000.00000	0.93827
2	!p+!	21	2212	0	0	0	0.00000	0.00000	-6999.99994	7000.00000	0.93827
3	!g!	21	21	1	0	0	0.24358	2.69425	193.71676	193.73565	0.00000
4	!dbar!	21	-1	2	0	0	-0.92442	0.76562	-497.54493	497.54638	0.00000
5	!g!	21	21	3	0	0	8.91980	-9.54714	91.32358	92.25349	0.00000
6	!g!	21	21	4	0	0	-6.53213	-3.81120	-201.36118	201.50315	0.00000
7	!h0!	21	25	5	0	0	-13.65932	2.67597	-0.12636	125.76747	124.99481
8	!h0!	21	25	5	0	0	16.04699	-16.03432	-109.91125	167.98918	125.00110
9	!b!	21	5	7	0	0	-1.49180	-59.19027	13.90868	61.00209	4.70000
10	!bbar!	21	-5	7	0	0	-12.16751	61.86625	-14.03504	64.76537	4.70000
11	!W+!	21	24	8	0	0	5.20279	-18.76637	-1.35530	33.62548	27.37865
12	!W-!	21	-24	8	0	0	10.84419	2.73205	-108.55595	134.36369	78.38461
13	!u!	21	2	11	0	0	15.03870	-6.91318	-5.89127	17.56877	0.00000
14	!dbar!	21	-1	11	0	0	-9.83590	-11.85319	4.53597	16.05672	0.00000
15	!mu-!	21	13	12	0	0	-33.42602	8.33897	-43.90010	55.80373	0.00000
16	!nu_mubar!	21	-14	12	0	0	44.27021	-5.60692	-64.65585	78.55996	0.00000
17	(h0)	11	25	7	56	65	-13.65932	2.67597	-0.12636	125.76747	124.99481
18	(h0)	11	25	8	19	20	16.04699	-16.03432	-109.91125	167.98918	125.00110
19	(W+)	11	24	11	66	69	5.20279	-18.76637	-1.35530	33.62548	27.37865
20	(W-)	11	-24	12	21	22	10.84419	2.73205	-108.55595	134.36369	78.38461
21	mu-	1	13	15	0	0	-33.42602	8.33897	-43.90010	55.80373	0.00000
22	nu_mubar	1	-14	16	0	0	44.27021	-5.60692	-64.65585	78.55996	0.00000
23	(u) A	12	2	1	74	74	0.05385	-1.55950	2194.43591	2194.43647	0.00000
24	(g) I	12	21	3	74	74	-0.06334	0.96097	2.67277	2.84098	0.00000
25	(g) I	12	21	3	74	74	-0.65036	0.16382	1.97245	2.08335	0.00000

Table 3.1: First lines of a `TPythia6::Pylist(2)` output for an example event, formatted as a table.

the mother particle of the string and the three-momentum of the particles in the string are determined. The particle in the string for which the angular difference is the smallest is chosen as the particle with which the chain continues. This avoids problems in situations where, for example, during the hadronisation of a b quark a second b quark is created.

Jet Reconstruction

The jets are reconstructed using FastJet 3.3.0. For this, the k_T algorithm [68, 69] is used, along with the recombination scheme “E-scheme”, in which the four-momentum of the particles are summed during their merger. The clustering strategy “Best” has been chosen, which automatically chooses the quickest strategy based on the parameters of FastJet and on the information of the given event. With this configuration, inclusive jets with radius $R = 0.4$ and a minimum transverse momentum $p_T^{min} = 5.0$ GeV are reconstructed.

Since this analysis doesn’t have access to particle tagging algorithms, the correct jets have to be selected by hand. This is done by first combing through the jets of an event and evaluating which jets contain partons that come from the original particle or its hadronisation process. For example, to find the jets belonging to a b quark, the Pythia indices of the particles in every jet are checked and compared to the indices of the final-state particles that have been associated with that b quark. This results in a list of jets that are considered for assignment to that particle.

The final assignment is done by comparing the ΔR between each jet and the original particle, which is defined as

$$\Delta R = \sqrt{(\Delta\phi)^2 + (\Delta\eta)^2}, \tag{3.1}$$

where $\Delta\phi$ is the difference between the azimuthal angle[2] ϕ of the jet and the original particle before hadronisation, and $\Delta\eta$ is the difference between the pseudorapidities[3] η. The jet with the smallest ΔR is then assigned to the original particle, as its direction is most in-line with that of the original particle.

In Fig. 3.3a, the number of jets that contain partons from the same b quark is depicted. As can be seen, in most events each b quark from a Higgs or Z boson generally only has one jet that contains its partons, meaning that the jet assignment is straight-forward. In cases where a single jet is the optimal jet for more than one particle (e.g. the b quarks are close to one another and are formed into a single jet), one of the particles will be assigned the second-most optimal jet. If there is no second-most optimal jet, then that particle doesn't have any jet assigned to it.

If a b quark from an H/Z boson does not have a unique jet associated with it, then the event is filtered out. The same is done for quarks and leptons coming from the W bosons. If an event is kept, then the assigned jets are pruned with $z_{cut} = 0.10$ and $R_{factor} = 1$. Pruning is defined as follows [70]:

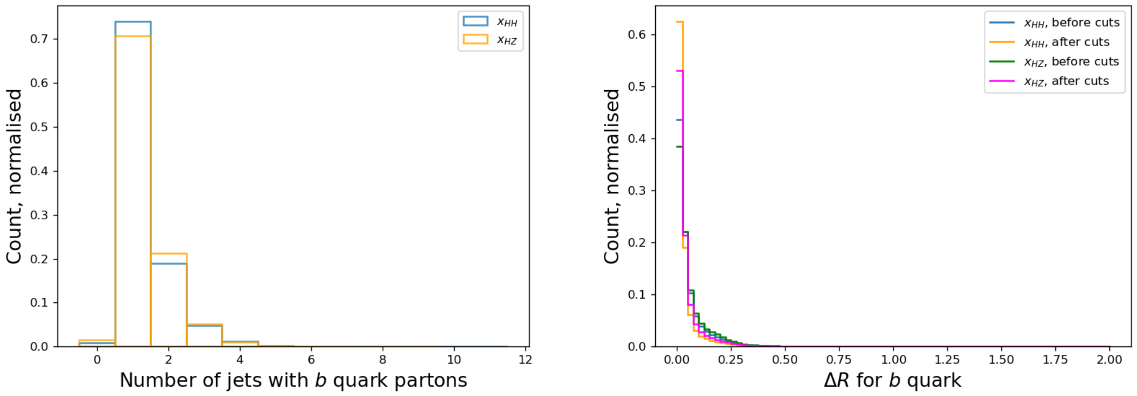
1. Form a list of all partons in a jet.
2. Calculate the distance $\rho_{i,j}$ between all pairs of partons i and j. Additional, determine the beam distance ρ_i for each parton i.
3. If either $\rho_{i,j} > \frac{2R_{factor}m_{jet}}{p_{T,jet}}$ or $\frac{min(p_{T,i},p_{T,j})}{z_{cut}} < p_{T,i+j}$, remove the parton with smaller p_T.

The goal of pruning is to improve the mass resolution of jets reconstructing heavy particles and to reduce QCD background [70] (i.e. background created through hadronisation). Then, all events that contain b quark jets that are too close to the beam axis ($\eta_b > 2.5$) are also filtered, as they would be difficult to detect in a general purpose detector such as ATLAS or CMS, since the detectors can't be built too close to the beam pipe. Finally, the energy values of the reconstructed jets E_{jet} are compared to those of the original particle E_{orig}: if for any jet $|E_{jet} - E_{orig}| > 10.0$ GeV, then the event is discarded. After applying these filters, 499,300 HH and 439,857 HZ events remain. The distributions of ΔR for the data before and after the cuts can be seen in Fig. 3.3b.

The resulting data, which is as input for the software that performs the MEM weight calculation, is entirely at generator level. Detector simulation or smearing of the four-momenta is not applied at this stage.

[2] The angle in the plane that is transverse to the beam axis.

[3] The pseudorapidity is calculated from the polar angle θ using $\eta = -ln\left(tan(\frac{\theta}{2})\right)$.



(a) **(b)**

Figure 3.3: **(a)** Number of jets that contain a parton from one specific b quark. **(b)** ΔR of reconstructed b quark jets, for both signal and background. The "cuts" refer to removing all events where a b quark has $\eta_b > 2.5$, or a reconstructed particle or jet has $|E_{jet} - E_{orig}| > 10.0$ GeV.

3.4 Weight Calculation

3.4.1 Overview

The weight of each event in calculated using MoMEMta (see Sec. 2.1.2). This is done using a modified version of *Block D*, since no combination of the preconfigured main and secondary blocks can be used to describe the complete decay topology.

A depiction of the original *Block D* can be seen in Fig. 3.4a. It's defined by having four visible and two invisible[4] particles.

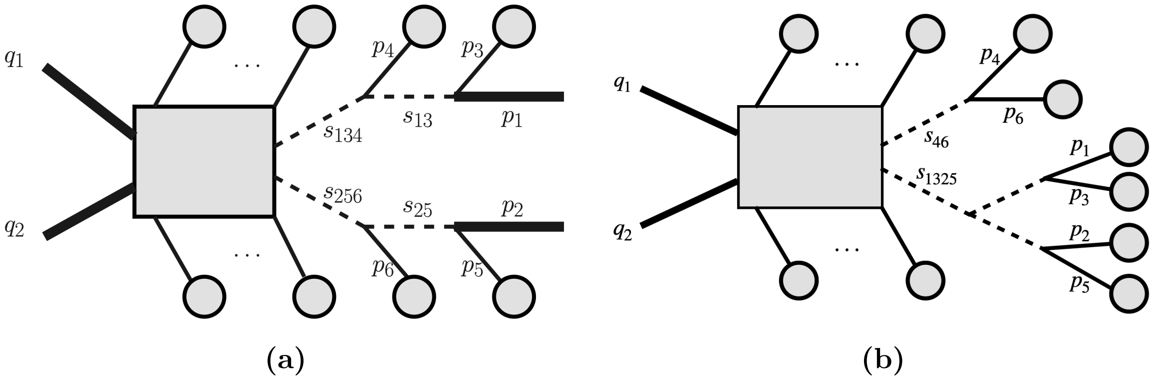


Figure 3.4: Visual depiction of Block D and Block HH. q_1 and q_2 are the Bjorken fractions of the ingoing particles and $s_{i...j}$ refers to $(p_i+...+p_j)^2$. **(a)** Block D: p_1 and p_2 denote the four-momenta of invisible particles, where $p_{3...6}$ are visible particles. Image taken from [39]. **(b)** Block HH: $p_{1...6}$ denote the four-momenta of visible particles.

The HH events contain only a single invisible particle, the neutrino (antineutrino) ν_l ($\bar{\nu}_l$), due to only one W boson decaying leptonically. This means that the four-momenta p_1 and p_2 can both be given explicitly — p_1, because it's a visible particle, and p_2, because it's described by the missing four-momentum in an event. Additionally, the block is restructured. The new block, named *Block HH* (see Fig. 3.4b), can simply take the four-momentum as input vectors and it needs to perform only two variable transformations for the Higgs bosons: s_{46} and s_{1325}.

Since Higgs self-coupling can only be described as a superposition with Higgs pair production, the matrix elements for either process can't be created separately. Instead, there is only one matrix element that contains both processes. This leads to both self-coupling and pair production being signal events when this matrix element is used to calculate the weight. To be able to conduct an analysis where a search for just Higgs self-coupling is performed, a better understanding for how self-coupling can be separated from pair production is necessary. Since the matrix element contains both modes, it isn't clear if the MEM will be able to separate self-coupling from other pair-production modes. This is because the destructive overlap of the wave functions will be affected by any tweaking of the matrix element (e.g. changing the strength of the self-coupling parameter), which would change the phase space of the decay.

[4] In this sense, "invisible" means that the particle can't be directly measured in a detector.

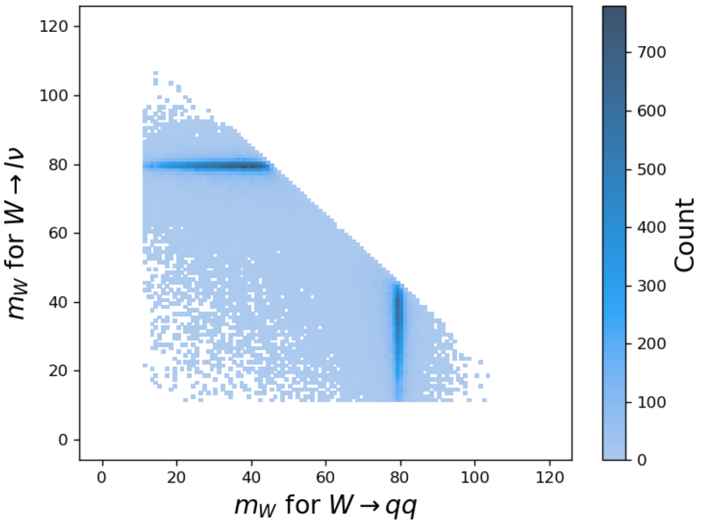


Figure 3.5: Distribution of the W boson masses, for both HH and HZ events.

3.4.2 Configuration

The computation of the weights requires configuring. The masses of the on-shell Higgs bosons are described by a Breit-Wigner distribution, with a peak at $m_H = 125.00$ GeV and a width of $\Gamma = 0.0032$ GeV. Normally, the masses of the W bosons would also be described by a Breit-Wigner distribution; however, this causes issues during the calculation of the weight. Since the Higgs boson's rest mass is much lower than the combined rest mass of two W bosons, one of the W bosons is required to be off-shell — its mass is expected to be around 45 GeV. This can be seen in Fig. 3.5, where the dark regions indicate that, in most cases, one W boson is around $m_W = 80$ GeV and the other is around $m_W = 35 - 45$ GeV. Due to the W boson's Breit-Wigner distribution having a small width, it's extremely unlikely that a MC generator would quickly find the area of the phase space that would produce such an off-shell W boson. When using a Breit-Wigner distribution for the W boson, MoMEMta is unable to converge on a value for the weight and fails due to reaching its iteration limit.

To circumnavigate this issue, the weight of the W bosons is chosen to be described by a flat distribution, meaning that it's just as likely to generate a W boson at 80 GeV than one at 20 GeV. This results in MoMEMta becoming able to converge on a weight within its iteration limits. This does, however, have a side effect that needs to be taken into consideration: as discussed in Sec. 1.5, the calculation of a weight is a result of a calculation of cross-sections. The quicker MoMEMta is able to converge on a weight for a given matrix element, the larger the phase space for that matrix element must be. In other words, there must be more acceptable configurations of particle masses and four-momenta for the intermediate and final-state particles for the given matrix element. Having a larger phase space must mean that the decay can happen more often, which is represented by a larger cross-section. Since the weight is proportional to the cross-section, then making changes that affect the cross-section also affects the weight.

To counter this side effect, an analysis that wants to measure Higgs self-coupling with off-shell W bosons would require a scaling factor for the weight to compensate for the fact that the W bosons are artificially being described with a flat distribution. This has not been investigated in this analysis, since the goal of this work is to implement the MEM on neural networks to reduce computation time, not accurately calculate weights as they would be expected to be when measuring real data. Whether the weights that the neural network trains with are off by a scaling factor will not have an effect on the network's ability to learn and therefore does not conflict with the objective of this analysis.

The PDFs used are CT10NLO [71], which is a general purpose next-to-leading-order set and is set to the energy scale of the SM Higgs boson mass of 125.0 GeV. The matrix element is generated with MadWeight, which is a functionality of MadGraph_aMC@NLO. For this, the model `Higgs_Effective_Couplings_UFO` is imported and the matrix element is produced with [57]:

```
./bin/mg5_aMC
import model Higgs_Effective_Couplings_UFO/
generate p p > h > hh, (h > bb~ ), (h > w+w-, w+ > jj, w- > l-vl~)
```

A plug-in of MoMEMta allows for the matrix element to easily be imported.

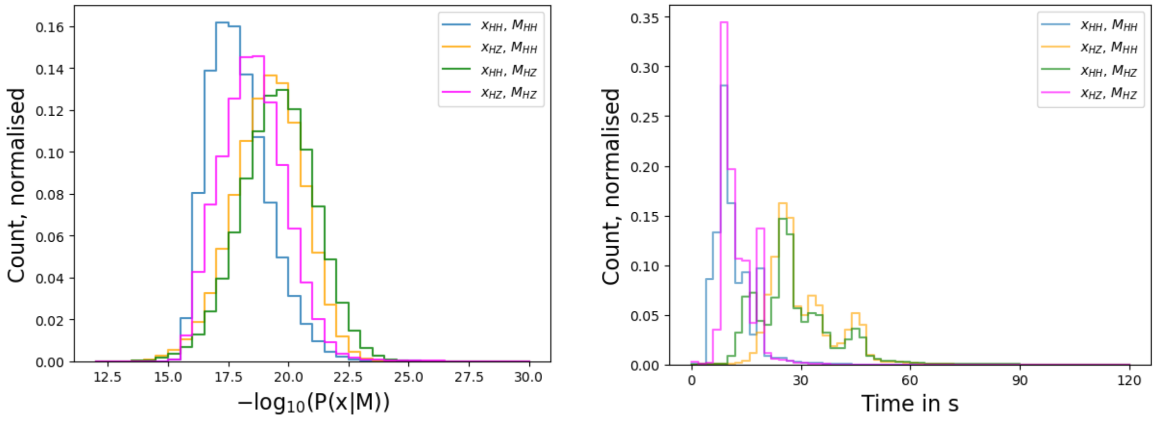
A Gaussian transfer function is used to smear the final-state particles, except for the neutrino. For b quarks, a standard deviation of $\sigma = 0.10$ is used, whereas for the quarks that come from a W boson and for the charged lepton a standard deviation of $\sigma = 0.05$ has been implemented.

The integration by CUBA uses the `Vegas` routine. It's implemented with a batch size of 50,000 and a relative accuracy of 0.01, and ran on four cores. Due to the reduction of the phase space dimensions by MoMEMta, only six integration variables are required. The iteration limit is set to 500,000; an error is returned if the weight hasn't converged within this limit. The jacobian has been deactivated for the computation of the weights. Including the jacobian drastically increases the computation time and it results in weights that are smaller by multiple orders of magnitude. It's unclear what is causing this: the calculation of the determinant of the jacobian is, for the most part, a linear equation, so its inclusion essentially scales the weight of the event by a constant. MoMEMta needs to be studied more closely to understand where such unexpected effects are coming from. Since the end goal of this analysis is to train a neural network to produce accurate weights, it's justifiable to exclude the jacobian, since whether a linear scaling factor is included, or not, will not have a noticeable impact on the network's performance.

3.4.3 Results from MoMEMta

In total, the weight calculation is performed twice: once where the weight is calculated with the self-coupling matrix element (M_{HH}) and once where matrix element for the background decay mode with Z bosons is used (M_{HZ}). The HH data set is referred to as x_{HH} and the HZ data set as x_{HZ}. In Fig. 3.6b, the time for each integration can be seen, where the colours represent the combinations of data and matrix elements that are used. The integration is significantly quicker when the data aligns with the matrix element (blue and magenta) compared to when they don't (orange and green) — this is the expected result. It's noticeable that in both cases where the data and

matrix element align (blue and magenta), the distributions of the integration times are very similar; the combination x_{HH} and M_{HH} seems to be slightly quicker, but not by much. Additionally, the cases where data and matrix element do not align (orange and green) have slower integration times. The distributions are very similar, but the case with x_{HH} data (green) tends to be slightly quicker. These slight differences could indicate that the phase space of the x_{HH} events fits into the acceptable phase space of the matrix element M_{HZ} better than the other way around; however the differences aren't significant enough to confidently be able to make any definitive statements and further studies will be required in the future.
The mean integration time for M_{HH} is ≈ 20.4 sec and for M_{HZ} it's ≈ 20.9 sec. With 499,300 HH and 439,857 HZ events, this leads to a total computation time of 3.66×10^7 sec, or $\approx 1.02 \times 10^4$ hours. The calculations were performed by running individual jobs of 1000 events, distributed onto 30 nodes of a local SLURM cluster, which resulted in each node needing around 340 hours of computation time.

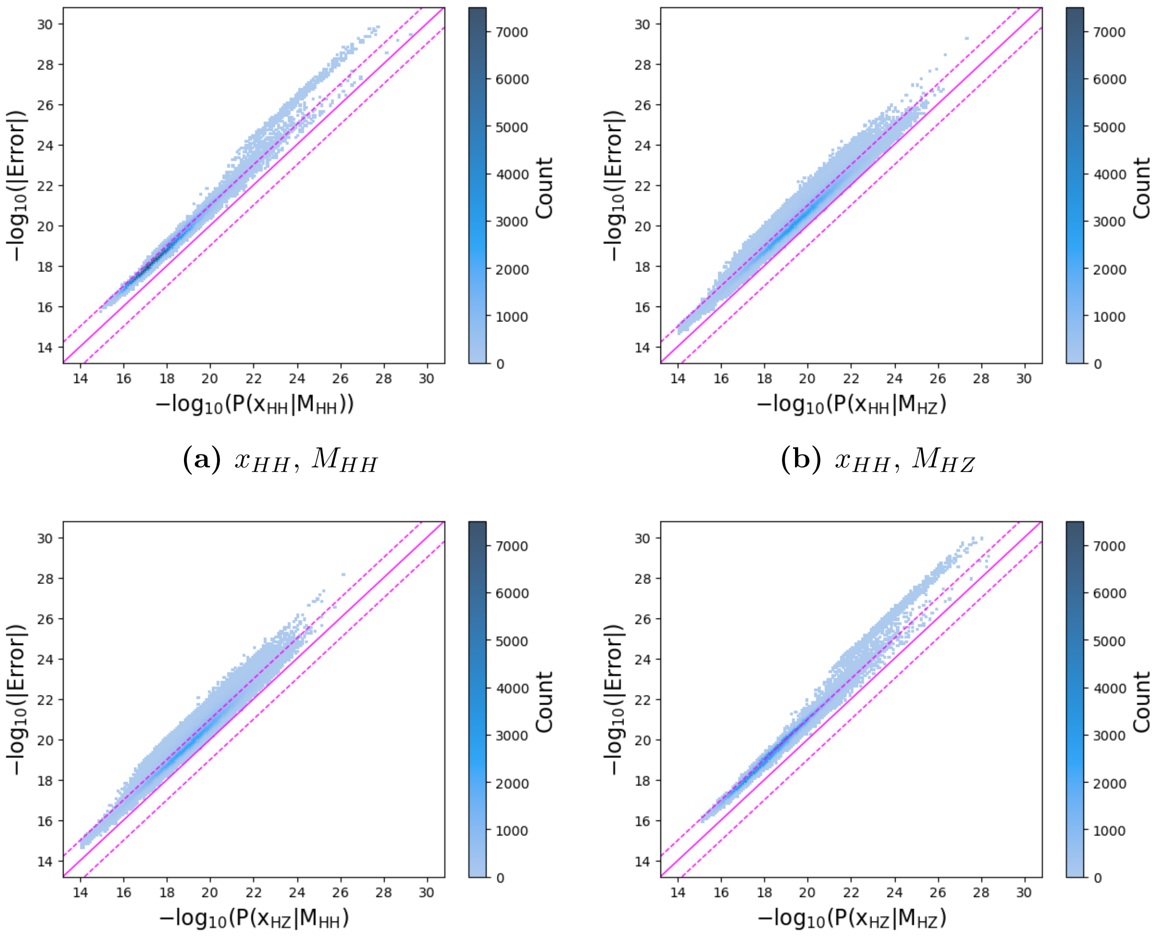


(a) **(b)**

Figure 3.6: The legends indicates the combinations of input data and matrix elements that are used in the calculations. **(a)** Weights for x_{HH} and x_{HZ} events in negative log scale (smaller values represent larger weights). **(b)** Calculation time for a single weight.

The distributions for the weights for both x_{HH} and x_{HZ} can be seen in Fig. 3.6a, computed with both M_{HH} and M_{HZ}. Note that the weights are depicted using a negative log scale, meaning that smaller values represent larger weights. While the peaks of the distributions for the x_{HH} data with M_{HH} (blue) and M_{HZ} (green) are separated by almost two orders of magnitude, the distributions as a whole have considerable overlap. The distributions for the x_{HZ} events, however, are much closer together. While the peaks for M_{HH} (orange) and M_{HZ} (magenta) themselves are very close to overlapping, but don't, most of the bodies do. This means that when given a x_{HZ} event, MoMEMta struggles more to determine whether or not this event is a Higgs pair event or an event with a Z boson, compared to identifying a x_{HH} event correctly.
Additionally, it can be seen that the combination of x_{HH} with M_{HH} results in larger weights than the combination x_{HZ} with M_{HZ}. This means that the phase space of an HH event is more distinct to the matrix element M_{HH} compared to the equivalent case for HZ events.

The error for each weight calculation can be seen in Fig. 3.7. While the errors for each configuration of data and matrix element are smaller than the weights themselves, the



(a) x_{HH}, M_{HH}

(b) x_{HH}, M_{HZ}

(c) x_{HZ}, M_{HH}

(d) x_{HZ}, M_{HZ}

Figure 3.7: The values of the weights and their associated errors (absolute value). The full magenta line indicates where an error has the same order of magnitude as the weight, the dashed lines indicate where there is a difference of one order of magnitude.

weights are slightly smaller when the data and the matrix element align. This can be seen by the distance of the distributions from the line through origin being a bit larger in Fig. 3.7a and Fig. 3.7d compared to Fig. 3.7b and Fig. 3.7b. The mean values for the difference between the weights and their errors are $\delta(x_{HH}, M_{HH}) \approx -0.812$, $\delta(x_{HH}, M_{HZ}) \approx -0.737$, $\delta(x_{HZ}, M_{HH}) \approx -0.796$ and $\delta(x_{HH}, M_{HH}) \approx -0.905$. This means that the error is smaller[5] when data and matrix element are aligned compared to when they are not aligned. This result is not surprising.

MoMEMta also provides a χ^2 test for each weight, the results of which are depicted in Fig. 3.8. The distributions for each case are very similar, although there is a slight tendency for weights calculated with M_{HZ} to have marginally larger values. The errors and χ^2 tests show that while there is a slight difference in MoMEMta's accuracy to be seen when comparing calculations performed with either M_{HH} or M_{HZ}, it isn't significant.

To get a better understanding of the time required to perform the weight integration, the integration times are compared to the weights produced by the integration (see

[5] If the error is smaller than the weight, then applying $-log_{10}$ to the errors and the weights will return a larger value for the error than the weight.

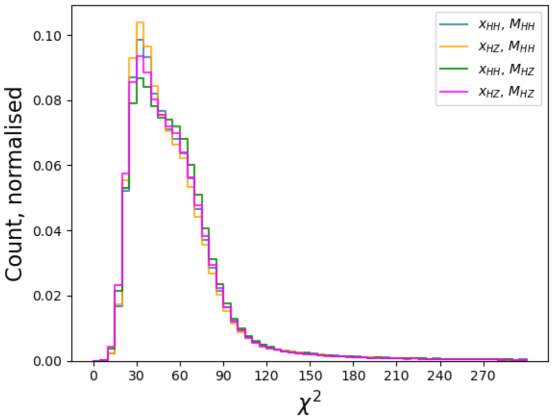


Figure 3.8: χ^2 test as provided by MoMEMta for each weight calculation.

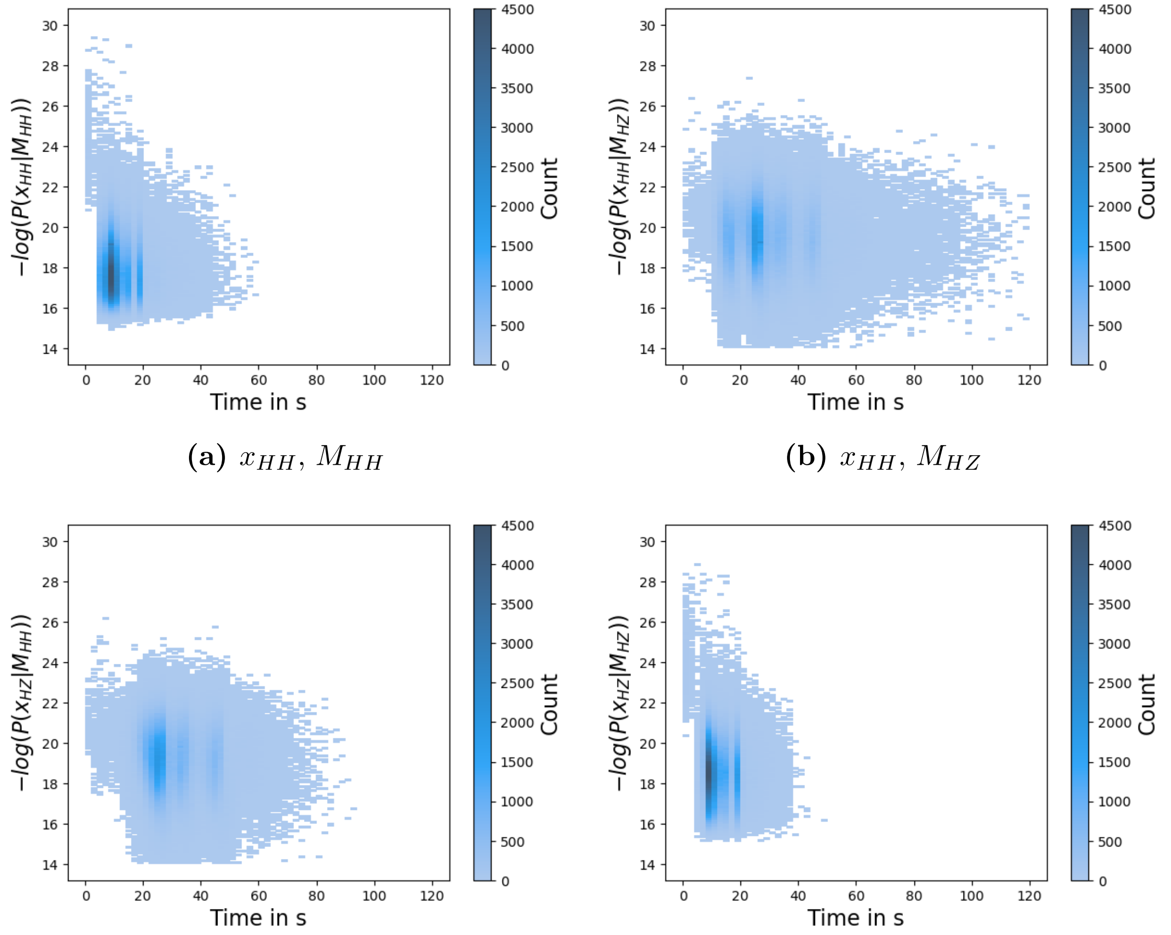
Fig. 3.9). For all four configurations, there are specific integration times that are preferred. The patterns in the cases of x_{HH} (Fig. 3.9a, 3.9b) have a similar structure with four preferred times, but the spacing is different: for M_{HH} the preferred times are much closer to one another, for M_{HZ} they are spaced out. The same can be said for x_{HZ}, only here there are three preferred integration times. While this behaviour could already be seen in Fig. 3.6b, it's easier to distinguish here.
The better the final-state momenta fit the matrix element, the quicker the integration time and the larger the weight is intuitively expected to be. However, this isn't the case: the values of the weights don't have a strong correlation to the integration time. This can be seen by the "height" of each bar: e.g. in Fig. 3.9a, the weights for small integration times are spread relatively evenly between 16 and 19. This shows that the integration time is influenced by an internal structure of MoMEMta. What causes some of the preferred integration times to be longer than the rest is unknown. This could be influenced by MoMEMta struggling to decide which W boson is on-shell and which off-shell, but this is speculation.

For each event, a discriminant introduced by [56],

$$D_{\pm}(x) = \frac{P(x|M_{HH}) - P(x|M_{HZ})}{P(x|M_{HH}) + P(x|M_{HZ})}, \tag{3.2}$$

is defined, which can be used to separate x_{HH} from x_{HZ}. As can be seen in Fig. 3.10a, for x_{HH}, most events lie between $0.95 < D_{\pm}(x_{HH}) < 1$, — this indicates that for x_{HH}, on a per event basis, the weight calculated with M_{HH} is generally significantly larger than the weight calculated with M_{HZ}.

x_{HZ} has its peak at $-1 < D_{\pm}(x_{HZ}) < -0.95$, which allows for a good separation between the x_{HH} and x_{HZ}. The distribution for x_{HZ} is not as heavily focused at the peak, but instead the tail is larger. Also, the secondary peak at $0.95 < D_{\pm}(x_{HZ}) < 1.0$ is larger than the secondary peak of x_{HH} at $-1 < D_{\pm}(x_{HH}) < -0.95$. This indicates that the computation of the weights for x_{HZ} is less likely to return a large difference between the M_{HH} and M_{HZ} weights. The split between x_{HH} and x_{HZ} is only possible because simulated data are being used. When using measured data from an experiment, where it isn't know if an event contains an HH or HZ decay, the discriminant will (ideally) result in a distribution that looks like a combination of the two seen in



(a) x_{HH}, M_{HH}

(b) x_{HH}, M_{HZ}

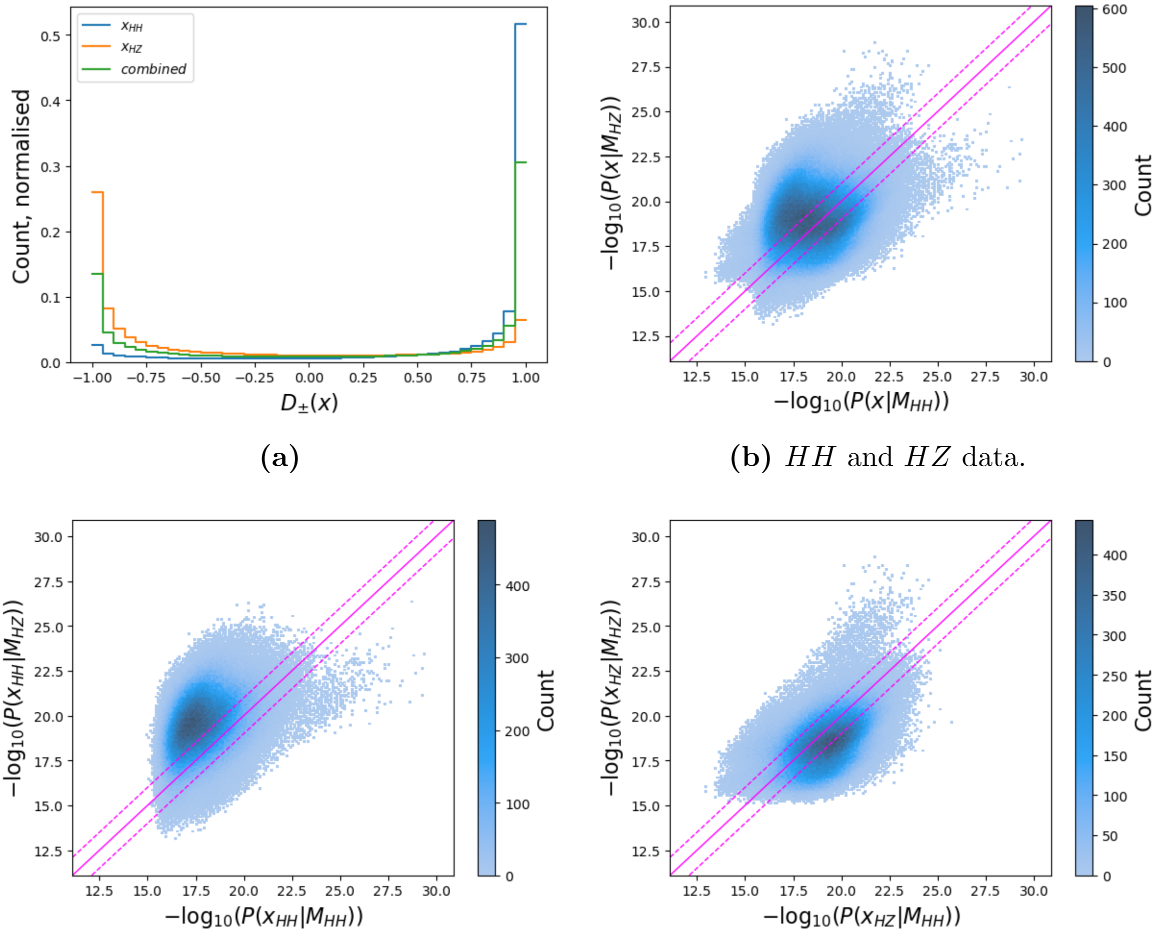
(c) x_{HZ}, M_{HH}

(d) x_{HZ}, M_{HZ}

Figure 3.9: Relation between the integration times and the values of the weights. Each plotx contains the same number of events.

Fig. 3.10a (green). If the neural networks are able to successfully predict the weights, then the resulting discriminant of the predicted weights will resemble the combined distribution.

When splitting the data into x_{HH} and x_{HZ}, it can be seen in Fig. 3.10c and Fig. 3.10d that the events have larger weights when the data align with the matrix element $(P(x_{HH}|M_{HH}, P(x_{HZ}|M_{HZ})$ compared to when they don't $(P(x_{HH}|M_{HZ}, P(x_{HZ}|M_{HH})$. However, the shapes are not as well defined as was hoped for: in an ideal case, every event that has a low value for one weight would have a high value for the other. This would result in a shape resembling a line from the top left to the bottom right. Additionally, there would be a large distance between the distributions and the line through origin (magenta line), as the line though origin indicates that the weights calculated with M_{HH} and M_{HZ} are of equal value. Instead of the ideal case, the distributions in both plots form blob-like shapes, which means that MoMEMta sometimes struggles to appropriately calculate a high weight when matrix element aligns with the data, and a low weight when they don't. For both x_{HH} and x_{HZ}, the distributions cross over the line through origin, meaning that there are some cases where MoMEMta incorrectly assigns a higher weight to the wrong matrix element. This is noticeably worse for x_{HZ}, meaning that MoMEMta is more likely to consider a background event as a signal



(a)

(b) HH and HZ data.

(c) HH data

(d) HZ data

Figure 3.10: **(a)** Discriminant for HH and HZ data. Comparison of the weight values calculated with M_{HH} (x-axes) and M_{HZ} (y-axes) for **(b)** combined x_{HH} and x_{HZ}, **(c)** just x_{HH} and **(d)** just x_{HZ}. The full magenta line indicates where an error has the same order of magnitude as the weight, the dashes lines indicate where the values of the weights are separated by one order of magnitude.

event than vice versa. This explains why the discriminant for x_{HH} has a larger peak and smaller tail compared to x_{HZ} in Fig. 3.10a.
The separation into x_{HH} and x_{HZ} is only possible due to the use of simulated data. The combined case results in the distribution seen in Fig. 3.10b.

When regarding how the integrations times, weight errors and χ^2 tests don't show much difference between x_{HH} and x_{HZ}, it's surprising to see that x_{HZ} results in less of a distinction between M_{HH} and M_{HZ} compared to x_{HH}. The reason for this must come from another source, and since the value of the weight is defined by the four-momentum of the final-state particles in an event, it makes sense to analyse the connection between the four-momentum and the weights.
The four-momenta of the six final-state particles result in 24 input parameters for the MEM. Due to the difficulty of visually depicting the influence of all of these parameters in a coherent way, just the b quarks that come from a Higgs or a Z boson are regarded, since this is where any strong differences between x_{HH} and x_{HZ} are most likely to be

visible. Specifically, the differences in the individual four-momentum parameters are studied: $E^{b_1} - E^{b_2}$, $p_i^{b_1} - p_i^{b_2}$ where $i = x, y, z$. The indices of b_1 and b_2 refer to the b quark from the Higgs or Z boson that was either first or second assigned a jet. The idea is to understand how strongly the orientation of the b quarks with respect to one another affects the weights. Not all events are used to study these parameters; instead, two subsections of x_{HH} and x_{HZ} are regarded. For this, a parameter is introduced:

$$\Delta P = -log_{10}(P(x|M_{sig})) - (-log_{10}(P(x|M_{back}))) \,. \tag{3.3}$$

Here, M_{sig} refers to the matrix element that aligns with the data and M_{back} to the matrix element that doesn't[6]. In cases where MoMEMta is able to correctly assign the weights according to the data (i.e. the top-left corner of Fig. 3.10c and the bottom-right corner of Fig. 3.10d), ΔP will have a negative number due to applying $-log_{10}$[7]. Therefore, the larger the negative value, the better. If MoMEMta is unsuccessful in correctly assigning weights, then ΔP will be positive.

In Fig. 3.11, the data has been split into $\Delta P < -1$ (Fig. 3.11a–3.11c, 3.11g–3.11i), which indicate events where there is a good separation, and $\Delta P > 1$ (Fig. 3.11d–3.11f, 3.11j–3.11l), which indicate events where there is a bad separation. It should be noted that the four individual cases have a different number of entries. While this does make some structures more clearly visible than others, each case has enough entries that observations can be made. The energy difference $E^{b_1} - E^{b_2}$ doesn't result in any visible influences on the weight and is therefore excluded from further analysis.

When regarding x_{HH}, $\Delta P < -1$, it can be seen that there are distinct shapes for the distributions of $p_x^{b_1} - p_x^{b_2}$, $p_y^{b_1} - p_y^{b_2}$ and $p_z^{b_1} - p_z^{b_2}$ (Fig. 3.11a). $p_x^{b_1} - p_x^{b_2}$ and $p_y^{b_1} - p_y^{b_2}$ adopt a flat peak with a slight dip in the middle, where as $p_z^{b_1} - p_z^{b_2}$ takes on a more peak-like distribution, but with a wider base. While this already indicates a ring-like structure, it can clearly be seen that there is a ring-like preference when comparing $p_x^{b_1} - p_x^{b_2}$ with $p_y^{b_1} - p_y^{b_2}$ (Fig. 3.11b). This means that there is a condition of $(p_x^{b_1} - p_x^{b_2})^2 + (p_y^{b_1} - p_y^{b_2})^2 \approx$ 110^2 – 120^2 GeV^2 that greatly increases the likelihood that MoMEMta will assign a large weight to x_{HH} for M_{HH} and a small weight for M_{HZ}. A similar type of ring-like structure can be seen when comparing $p_x^{b_1} - p_x^{b_2}$ with $p_z^{b_1} - p_z^{b_2}$ (Fig. 3.11c); however, it is not as well defined. This indicates that the relative orientation of the b quarks in the x-y-plain, i.e. the azimuth angle between the quarks, has a bigger impact on the weights than their orientation along the z-axis[8]. For MoMEMta to be able to correctly assign weights, it's important that the b quarks have a strong separation in the x-y-plain.

This ring-like structure is not seen for x_{HH}, $\Delta P > 1$, where the separation of the M_{HH} and M_{HZ} weights is bad. Here, $p_x^{b_1} - p_x^{b_2}$ and $p_y^{b_1} - p_y^{b_2}$ result in a relatively even distribution in a circle-like shape, with a slightly higher density in the middle (Fig. 3.11e). This circle has a radius of about 90 GeV and resembles the missing inside-area of the ring in Fig. 3.11b. These relations can also be seen for $p_x^{b_1} - p_x^{b_2}$ and $p_z^{b_1} - p_z^{b_2}$: the shape in Fig: 3.11f looks like the inside-area of the ring in Fig. 3.11c.

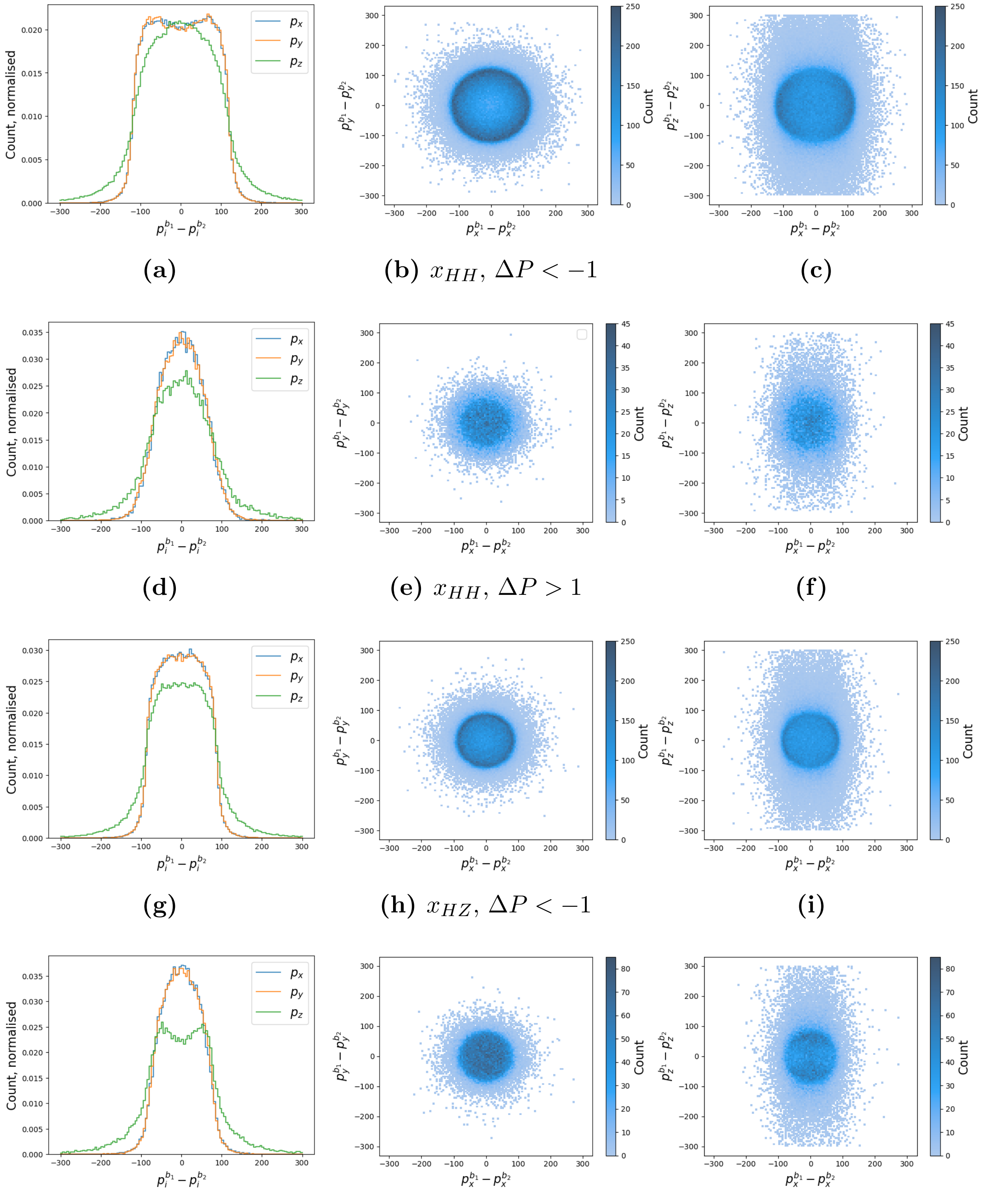
[6] E.g., $M_{sig} = M_{HH}$ for x_{HH}.

[7] E.g., if $P(x_{HH}|M_{HH}) = 10^{-19}$ and $P(x_{HH}|M_{HZ}) = 10^{-21}$, then $\Delta P = -2$

[8] The z-axis is the beam/collision axis.

For x_{HZ}, $\Delta P < -1$ similar structures can be seen compared to the x_{HH} case, but they are less well defined. For Fig. 3.11h and Fig. 3.11i, the rings aren't as hollow. This is reflected in the histograms, as $p_x^{b_1} - p_x^{b_2}$ and $p_y^{b_1} - p_y^{b_2}$ don't have flat distributions. However, $p_z^{b_1} - p_z^{b_2}$ has a flatter peak with a slightly more box-like distribution as compared to x_{HH}. Additionally, the condition for the x-y-plain is slightly different: the preference is for $(p_x^{b_1} - p_x^{b_2})^2 + (p_y^{b_1} - p_y^{b_2})^2 \approx 70^2$ – 80^2 GeV^2. This difference is most-likely due to the Z boson having a smaller mass (91 GeV) than the Higgs boson (125 GeV), since both conditions correlate strongly to the bosons' masses.
For x_{HZ}, $\Delta P > 1$, the circle-like distribution is more clearly visible than in the x_{HH} case; however, this is partially due to the difference in the number of entries. What is unique to x_{HZ}, $\Delta P > 1$, is the strong dip in the peak of the histogram of $p_z^{b_1} - p_z^{b_2}$. This means that for x_{HH}, events where the b quarks have a small separation in their momentum along the z-axis are less likely to result in MoMEMta doing a bad job of calculating the weights compared to when the b quarks are separated by up to around 80 GeV on the z-axis.

Therefore, a very good way to greatly increase the likelihood of an event to correctly be assigned a high weight for M_{sig} and a low weight for M_{back}, there needs to be a large separation between the two b quarks that come from the Higgs or Z boson in the x-y-plain. This requirement isn't quite as strong for x_{HZ}, which can at least partially explain why the distribution in Fig. 3.10d crosses over the line through origin more than Fig. 3.10c.



(a) **(b)** x_{HH}, $\Delta P < -1$ **(c)**

(d) **(e)** x_{HH}, $\Delta P > 1$ **(f)**

(g) **(h)** x_{HZ}, $\Delta P < -1$ **(i)**

(j) **(k)** x_{HZ}, $\Delta P > 1$ **(l)**

Figure 3.11: Comparing parameters to identify differences between events with large and events with small ΔP. **(a)(b)(c)** x_{HH}, $\Delta P < -1$, contains 328,841 events; **(d)(e)(f)** x_{HH}, $\Delta P > 1$, contains 27,714 events; **(g)(h)(i)** x_{HZ}, $\Delta P < -1$, contains 183,642 events; **(j)(k)(l)** x_{HZ}, $\Delta P > 1$, contains 57,191 events.

3.5 Additional Parameters to Support the Neural Network

The job of a neural network is to take training input and output parameters and to learn to recognise a pattern between them. The network itself has no concept for what the parameters represent — it just looks for correlations between numbers. This means that the input parameters of a neural network can be anything that is associated with the data points. Concretely, this means that for this analysis, the input parameters of a neural network don't have to be the four-momenta of each final-state particle, as it is with the MEM. The only condition is that the chosen input parameters come from the same event for which the weights have been computed. However, for this analysis, the only parameters that are used to train the networks are the four-momenta for the final-state particles, without additional parameters. This was chosen to understand how well the MEM can be replicated by a neural network at a base level: since the MEM only uses the four-momenta, a neural network should, in theory, be able to predict accurate weights without needing additional support. Nevertheless, possible additional parameters are discussed in the following that can potentially be used in to increase the capabilities of the neural networks.

Each input parameter should ideally not have any overlap with other parameters, as this maximises the information that each parameter delivers individually. However, it can be helpful to provide parameters that do have overlap with others if that overlap is greatly obfuscated, as this can speed-up the network's learning process. A good understanding of the subject is required to be able to choose parameters for which any differences between the data points become the most visible.

Spin Differences

The main difference between the HH and HZ events is that, for HH events, a Higgs boson decays to two b quarks, whereas for HZ events the b quarks come from a Z boson. There are two obvious differences between Higgs and Z bosons that can be exploited: the differences in mass and spin. Regarding the latter, the Higgs and Z bosons being a scalar and a vector particle, respectively, has an impact on the angles with which their decay products are emitted. This difference should be especially visible in the rest frame of the mother particle. For a scalar particle, it would be expected that the decay products won't have a preferred angle; instead, a flat distribution for the decay angle is expected. For a vector particle, a preference is expected.

The difference in the angles between the Higgs/Z boson and a b quark Lorentz-boosted into the rest frame of the Higgs/Z boson can be seen in Fig. 3.12. There are multiple version for the implementation of the lorentz boost: using either `Fortran77`, `C++` with `ROOT`, `C++` by hand and `Python 3`. The reason for this is that `C++` and `Python 3` returned unexpected results: first, x_{HH} and x_{HZ} have very similar distributions which should not be the case, and second, neither of the distributions resemble theoretical predictions. x_{HH} should have a flat distribution and x_{HZ} is expected to have a peak at $0, 2\pi$. Neither is the case: the absolute value of the angle has a peak at $\frac{\pi}{2}$ for both cases.

Since this is using both reconstructed jets and truth data[9], it can be concluded that the issue doesn't lay in the jet reconstruction. As an additional check, the Higgs bosons have been boosted back into their own rest-frame, to see if there is a problem with the boosting function. In all cases, this return three momenta of 0 and masses around the particles' masses, which show that the boosts have been correctly implemented. Therefore, there is no obvious issue that causes these results. However, the `Fortran77` implementation, which also uses truth data, shows the expected results. This means that the problem doesn't lay in the generated data[10].
Since the `Fortran77` implementation shows that the data has been generated correctly and studying the four-momenta of the final-state particles reveals that they contain values that make sense (which are then used for MEM weight calculations), it's assumed that the data is safe to use for this analysis and that issue must bafflingly lay somewhere along the plotting process.

Further studies have been done to understand how the difference in spin can be used to separate HH and HZ events. One such parameter is a modified version of the Ellis-Karliner angle [72], where the angle description has been expanded to include particle masses. This results in a significant difference between HH and HZ events.

Rotationally Symmetric Input Parameters

A second set of input parameters can be constructed from the four-momenta: rotationally symmetric parameters. These consist of the energy E, the transverse momentum p_T, the pseudorapidity $\eta = -ln(tan(\frac{\theta}{2}))$[11] and the azimuth angle ϕ. These parameters describe the four-momenta in such a way that the rotational orientation in a detector is not present in p_T or η, and is solely described by ϕ. The idea is that rotationally symmetric parameters will make it easier for a neural network to learn that the rotational orientation of the entire topology is not important. Instead, what is important is the topology of the jets in relation to one-another.

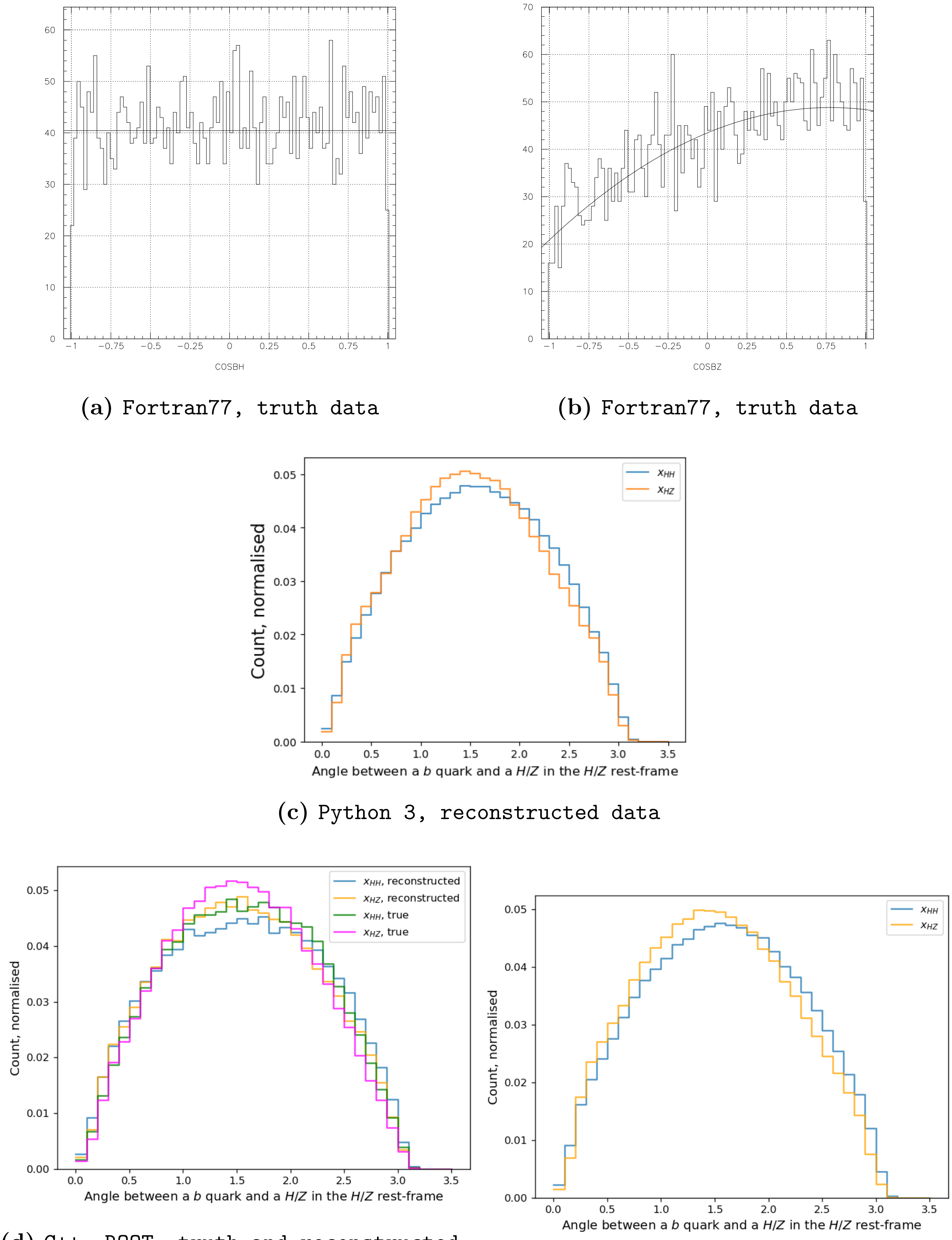
Other Parameters

Additionally, parameters that have been studied in Sec. 3.4.3 can be used as input: $(p_x^{b_1} - p_x^{b_2})^2 + (p_y^{b_1} - p_y^{b_2})^2$ and $(p_x^{b_1} - p_x^{b_2})^2 + (p_z^{b_1} - p_z^{b_2})^2$ have been shown to have preferred values in cases where the weights had a separation of at least one order of magnitude. These parameter would therefore help teach the neural network under which circumstances one weight should be larger than the other. This information is already included in the four-momentum and three parameters are therefore introducing nothing but overlap, but it's possible that this information is too obscured and that the network's performance would therefore be boosted by their inclusion.

[9] The four-momenta of the Higgs boson and the b quarks are directly read-out from `Pythia`.

[10] It was briefly considered that the spin information might not be passed from MadGraph to Pythia, leading to the same results for the Higgs and Z boson, but this turned out not to be the case.

[11] Where θ is the polar angle.



(a) `Fortran77, truth data`

(b) `Fortran77, truth data`

(c) `Python 3, reconstructed data`

(d) `C++, ROOT, truth and reconstructed data`

(e) `C++, by hand, reconstructed data`

Figure 3.12: Comparison of the angle between one b quark and Higgs/Z in the Higgs/Z rest-frame, determined in different programming languages: **(a)**,**(b)** `Fortran77`, boost implemented by hand; **(c)** `Python3`, boost implemented by hand; **(d)**,**(e)** `C++`, boost implemented with `ROOT`/by hand.

3.6 Machine Learning Analysis

The main goal of this analysis is to train a neural network to be able to accurately predict the weight of an event for a given matrix element to reduce the computing time needed for the MEM. Ideally, the distribution of the discriminant in Fig. 3.10a will be replicated using the results of the neural networks.
This analysis focuses on two different types of neural networks: feed-forward and convolutional. Feed-forward neural networks are a very common type of network that aren't specialised on solving any specific type of problem. Convolutional neural networks have been designed to excel at recognising isolated structures anywhere within a set of data; for example, finding a face in a picture. While there are other types of neural networks, such as graph neural networks, that would be interesting to examine, due to time constraints, only these two types are taken into consideration.

3.6.1 Data Preparation

The networks use just the four-momenta of the final-state particles as input parameters, to understand how well the MEM can be replicated by a neural network at a base level. The data sets are normalised in three steps: first, each input parameter a is divided by a factor f that aims to scale the values closer to 0. For the four momentum, this factor is $f = 1000$ for each parameter. Then, the mean $\bar{a}_f$ of each parameter is determined and subsequently subtracted from each parameter, respectively. Lastly, the standard deviation σ_{a_f} for each parameter is determined and each parameter is divided by its standard deviation:

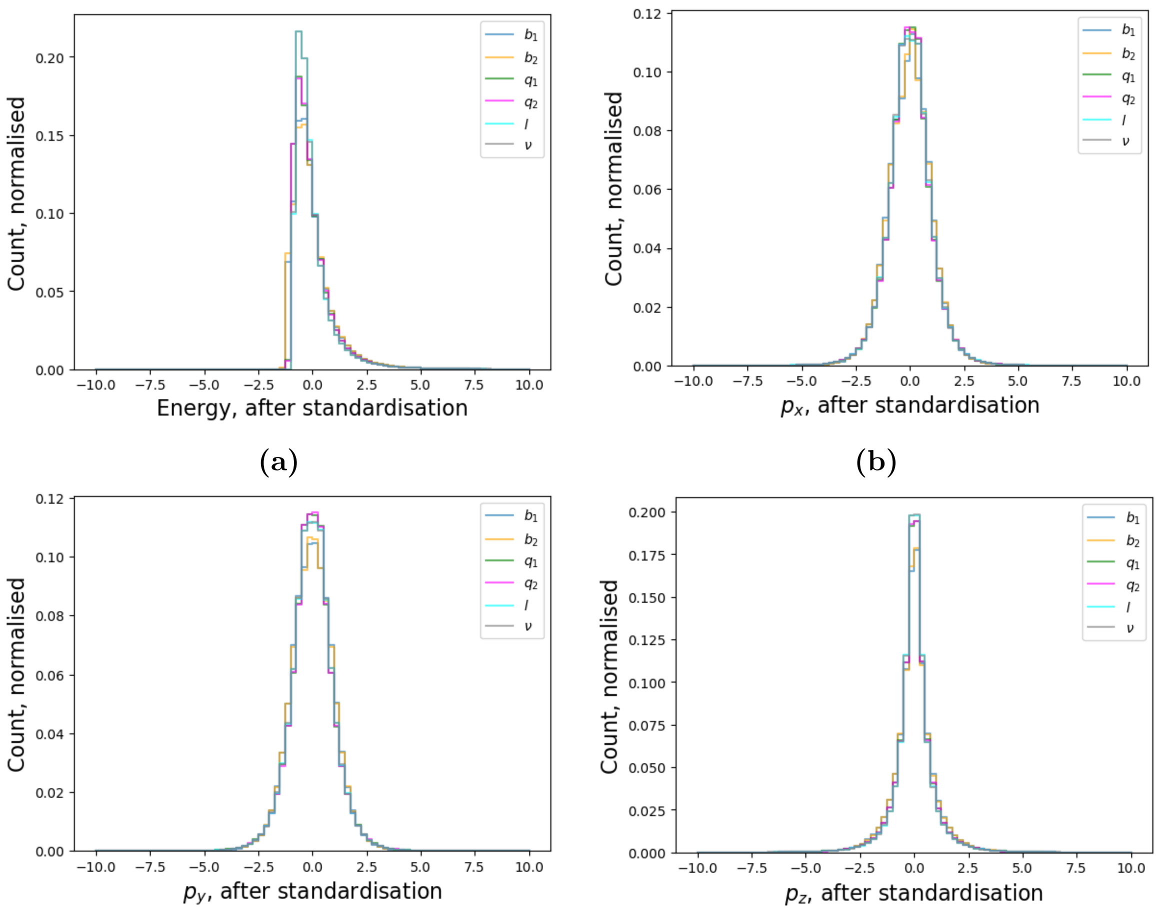
$$a_f = \frac{a}{f} \tag{3.4}$$

$$a_{norm} = \frac{a_f - \bar{a}_f}{\sigma_{a_f}}. \tag{3.5}$$

This leads to distributions that have a mean value of 0 with a standard deviation of 1. The normalised four-momentum data can be seen in Fig. 3.13. The upper and lower bounds of the distributions are close to 0. The largest visible differences are between the quarks b_1 (dark blue) and b_2 (orange). This likely comes from the jet assignment (see Sec. 3.3): b_1 and b_2 are always assigned different jets, where b_1 describes the b quark that gets a jet assigned to it first. If both b_1 and b_2 have the same optimal jet, then this will lead to b_2 being assigned to its secondary jet[12]. The secondary jet will generally have a lower energy, which explains the considerably lower peak for the energy of the b_2 quark.

In total, the data consist of 439,857 HH and 439,857 HZ events. The number of HH events has been reduced to have an even split between the number of signal and background events. Of this data, each network is trained using an individually randomised sub-sample consisting of 791,742 events (90% of the total data) and the networks are tested on 87,972 validation events (the remaining 10% of the total data)

[12] Cases in which b_1 and b_2 both have the same optimal jet and b_2 doesn't have a second jet have been filtered out.



(a) **(b)**

(c) **(d)**

Figure 3.13: Normalised distributions of for both x_{HH} and x_{HZ} using the four-momentum: **(a)** Energy, **(b)** p_x, **(c)** p_y, **(d)** p_z.

every epoch. Due to the random sampling, both the training and validation data are most likely to not have an exactly even 50–50 split between signal and background; however, testing the same network multiple times results in the same performance each time, so the potentially uneven split doesn't cause issues.

In the following analysis of various neural network structures, the value of the R^2-parameter is only shown for the validation data, not the training data. The networks are tuned so that the difference between the values of the R^2-parameter for the training data and validation data is no larger than 1. This means that none of the networks have considerable issues with overfitting, unless stated otherwise.

3.6.2 Feed-forward Neural Network

Optimising a neural network requires a large amount of testing of the various hyperparameters that influence its ability to learn. The feed-forward networks that are presented in this section use the following hyperparameters:

- batch size = 32

- learning rate = 10^{-4}
- momentum=0.99
- weight decay = 10^{-8}
- Nesterov = True
- activation function: LeakyReLU
- negative slope: 0.1
- optimisation: SGD

These values have been reached by testing a range of values on multiple network structures. They consistently lead to a network either being the best-performing or one of the best-performing networks among those that were tested for a given structure. Due to this consistency, the decision has been made to keep these values set and to focus on getting a better understanding of how the structures of the networks influence their abilities to learn.
The first question to be asked is how many layers and nodes are necessary: is a shallow, but wide[13] network better, or a deep, but narrow[14] network? To answer this, three networks are looked at, each with the aforementioned hyperparameters apart form the number of layers and nodes. The first is labelled *shallow+wide*. This has one hidden layer and the input and hidden layers each have 600 nodes. The second is the *mixed* network, which has 320 nodes on the input layer, but the three hidden layers each have consecutively less nodes. The third network, labelled *deep+narrow*[15], has six hidden layers, where every layer, including the input layer, each has 32 nodes. To help reduce overfitting, dropout of 0.1 is applied to the last hidden layer in each network. Their structures are depicted in Tab. 3.2.

The activation function LeakyReLU is chosen, because it's less likely to lead to vanishing gradients, where the gradients of the first layers tend to 0. This is due to LeakyReLU returning a non-zero value for a negative input, while also not converging on an output value for very large negative or positive input values (see Fig. 1.8). The SGD optimisation has been shown to have improved results for generalisation compared to adaptive optimisation methods [73], so even though it's slower, it has been chosen for its better performance. While the shallow+wide and mixed network are able to achieve similar results, as can be seen in Fig. 3.14, the deep neural network performs significantly worse (see Fig. 3.14).

It's possible that the networks suffer from vanishing gradients. To check if this is the case, the gradients of the input layer and last hidden layer of the mixed and of the deep+narrow network are examined. The gradients of the weights and biases of the input layer are generally larger than those of the final hidden layer, which fortunately means that the network doesn't suffer from the issue of vanishing gradients. The weight

[13] Small number of layers with a high number of nodes on each layer.

[14] High number of layers with a small number of nodes on each layer.

[15] Generally speaking, six hidden layers isn't very deep for a neural network. The use of "deep" here is simply to be able to distinguish this network from the other two.

Layer	Network 1	Network 2	Network 3
Input	600	320	32
1. Hidden	600	160	32
2. Hidden	-	80	32
3. Hidden	-	40	32
4. Hidden	-	-	32
5. Hidden	-	-	32
6. Hidden	-	-	32
Dropout	0.1	0.1	0.1
Output	1	1	1

Table 3.2: Structures of three networks that were tested to find out whether a shallow+wide (Network 1) or deep+narrow (Network 3) network is better. Network 2 is a mixture of both, where it starts out wide, but then the number of nodes is reduced with each layer. Dropout of 0.1 is applied to the last hidden layer in each network.

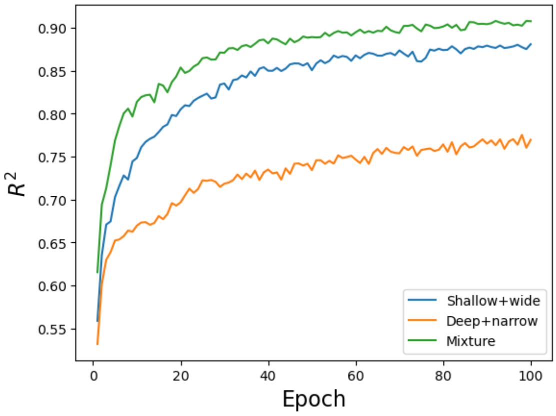


Figure 3.14: R^2 parameter for the deep+narrow, shallow+wide, and mixed networks.

distribution broadens significantly over numerous epochs for both networks, but moreso for the dee+narrow network. In addition, for both networks, the distributions for the values of the weights remains stable in the later epochs, indicating that the network is able to learn steadily. This correlates with the values of the R^2-parameter maintaining steady values for each network. The weights of the input layer and the gradients of the input layer and last hidden layer can be seen in App. B.1) for the mixed network and App. B.2) for the deep+narrow network.

Residuals

Although there isn't an issue with vanishing gradients, methods that can improve the backpropagation are introduced. The first is implementing residuals (see Sec. 1.6.6). Residuals require that the layer that is receiving the residuals has the same number of neurons as the layer from which the residuals come. This isn't an issue with the deep+narrow network, as each layer has 32 neurons. Here, residuals have been introduced to the second, fourth and sixth hidden layers (see Fig. 3.15b). Adding residuals isn't possible for the mixed neural network as is, since none of the layers contain the same number of neurons as any other layer. To be able to include residuals, an additional layer has to be introduced (see Fig. 3.15a). Since the shallow network has only one hidden layer, adding residuals isn't worthwhile.

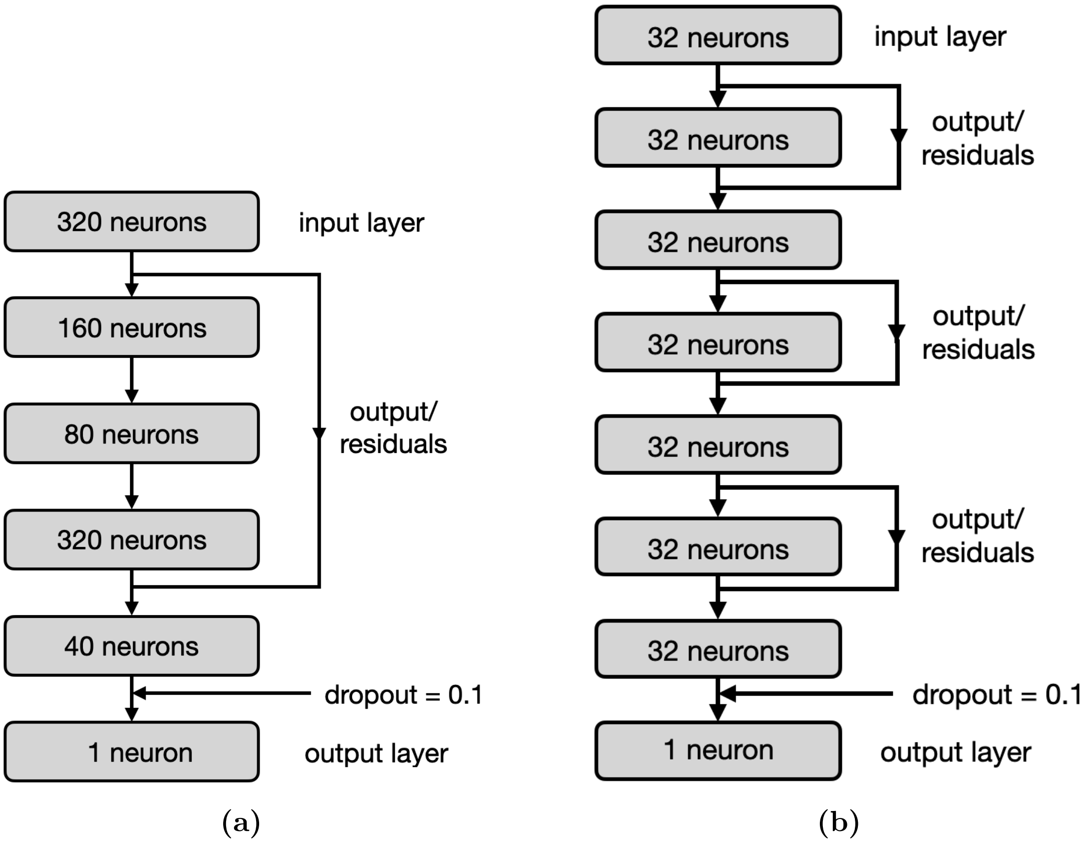


Figure 3.15: Structure of **(a)** mixed and **(a)** deep+narrow neural networks after residuals have been introduced. Each layer, apart from the output layers, is using the activation function LeakyReLU.

Introducing the residuals doesn't have much of an impact on either network regarding the R^2 parameter (Fig. 3.16). This isn't unexpected, as the networks already had non-zero gradients through-out the network across all epochs. The weights of the input layer spread out less with the residuals than they do without. This implies that the network puts less value on certain input variables and instead learns from all of them equally. While this isn't reflected in the R^2-parameter values of the deep+narrow or mixed networks, it's nevertheless a better outcome. The same can be said for the mixed network, but in this case, the weights aren't as spread out before residuals are introduced. The weights and gradients of the input layer for the deep+narrow residual network and the mixed residual network can be seen in App. B.3.

Fig. B.3d shows the weights for the input layer of the deep+narrow residual network. The weights of the input layer spread out less with the residuals than they do without (compare with Fig: B.2c). This implies that the network puts less value on certain input variables and instead learns from all of them equally. While this isn't reflected in the R^2-parameter values of the deep+narrow or mixed networks, it's nevertheless a better outcome. The same can be said for the mixed network, but in this case, the weights aren't as spread out before residuals are introduced (compare Fig. B.3b with Fig. B.1c).

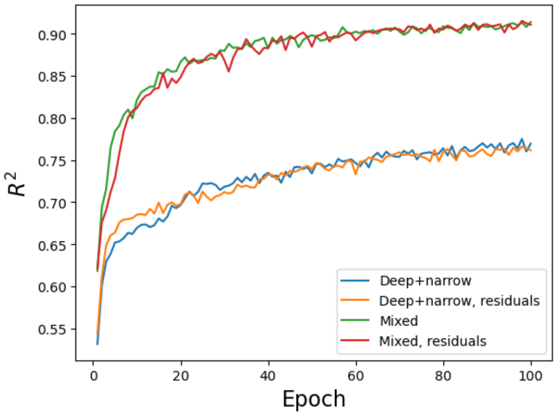


Figure 3.16: Comparison for the R^2-parameter for the deep+narrow and the mixed network once residuals have been introduced.

Weight Regularisation

The second method used for improving gradients is weight regularisation. By changing the values that the weights are first set to, the network's capability to learn can be greatly altered. For this, two weighting schemes are compared. The first is using a *Xavier uniform* distribution[16] [74], where the weights are initialised from a uniform distribution with

$$W_j \sim U\left[-\sqrt{\frac{6}{n_j + n_{j+1}}}, \sqrt{\frac{6}{n_j + n_{j+1}}}\right], \tag{3.6}$$

with n_j being the number of input paths and n_{j+1} being the number of output paths of that layer. Biases are initialised to 0. The PyTorch implementation of Xavier initialisation has an additional parameter: *gain*. Gain is used to scale the initialised weights based on the activation function that is used. For LeakyReLU, the optimal gain is $\sqrt{\frac{2}{1+ns^2}}$, where ns is the negative slope.

The second is using a *Kaiming* distribution[17] [75], where the weights are initialised from a uniform distribution with[18]

$$W_j \sim U\left[-\sqrt{\frac{3}{n_x}}, \sqrt{\frac{3}{n_x}}\right], \tag{3.7}$$

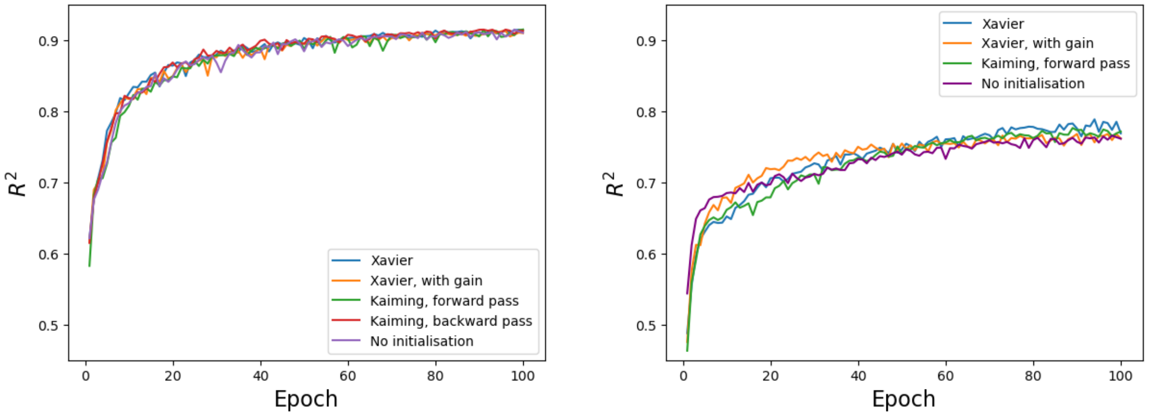
where n_x can either be the number of input paths (n_j, for forward propagation) or the number of output paths (n_{j+1}, for backward propagation) and the biases are initialised to 0. In this comparison, the backward propagation is used. In Fig. 3.17b, the R^2 values for the deep+shallow residual neural network with both the Xavier and Kaiming distributions can be seen. While there are more differences between the networks with weight initialisation and those without compared to the mixed network, they aren't noteworthy. The residual deep+narrow neural network doesn't have an entry for the

[16] Also known as *Glorot initialisation*

[17] Also known as *He initialisation*

[18] This is the PyTorch implementation. Originally, it used $\sqrt{\frac{2}{n_x}}$

Kaiming weight initialisation with backward pass, as this leads to the network failing during the first epoch.



(a)

(b)

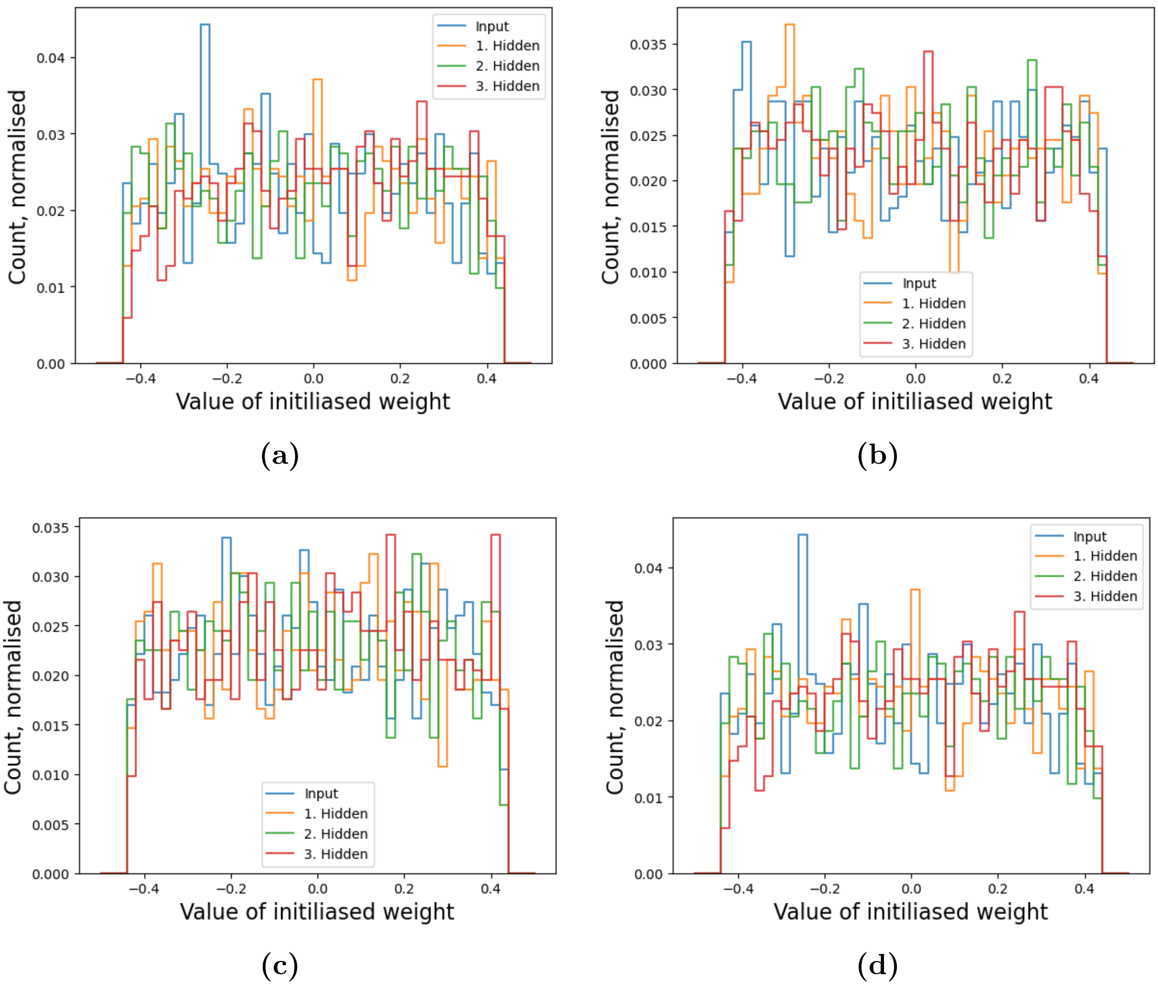
Figure 3.17: R^2-parameter for **(a)** the residual mixture neural networks and **(b)** the residual deep+narrow neural networks with weight initialisation.

Introducing both residuals and weight initialisation doesn't lead to a significant improvement for the mixed network, which so far has been the best performing network. Fortunately, the network is already performing rather successfully, which means that it isn't reliant on further improvement methods to be able to achieve an acceptable result. However, to understand why the weight initialisations aren't having any significant effects, the weights for the different initialisations are depicted in Fig. 3.18. Only the weight initialisation of the deep-narrow network is shown, since this network has the same number of weights for each layer and is therefore easier to depict. For the sake of visual clarity, only the input layer and the first three hidden layers are depicted.

All four cases have essentially the same initialised weights. This doesn't make sense: since the network has 32 nodes at each layer and therefore 32 input/output channels, the limits for the Xavier initialisation should be $[-\frac{\sqrt{6}}{8}, \frac{\sqrt{6}}{8}] \approx [-0.3, 0.3]$ without gain and $[-0.42, 0.42]$[19] with gain. However, both instances of Xavier initialisation result in distributions with limits of around $[-0.42, 0.42]$. For the Kaiming initialisation, the forward and backward passes should both[20] have limits of $\approx [-0.31, 0.31]$, but they, too, have distributions with limits of around $[-0.42, 0.42]$. Yet, for the backward pass, and only the backward pass, the network always fails during the first epoch, which indicates that there is some difference in the weight initialisation that causes this different behaviour. The technical implementation of the initialisation has been thoroughly checked and no errors have been found. This is an issue that can't be investigated within the scope of this analysis and will have to be studied in future work. Due to these behaviours not being understood, weight initialisation is not implemented in further networks.

[19] Using a negative slope of 0.1, the gain is ≈ 1.41.

[20] Since each layer has the same number of input/output channels.



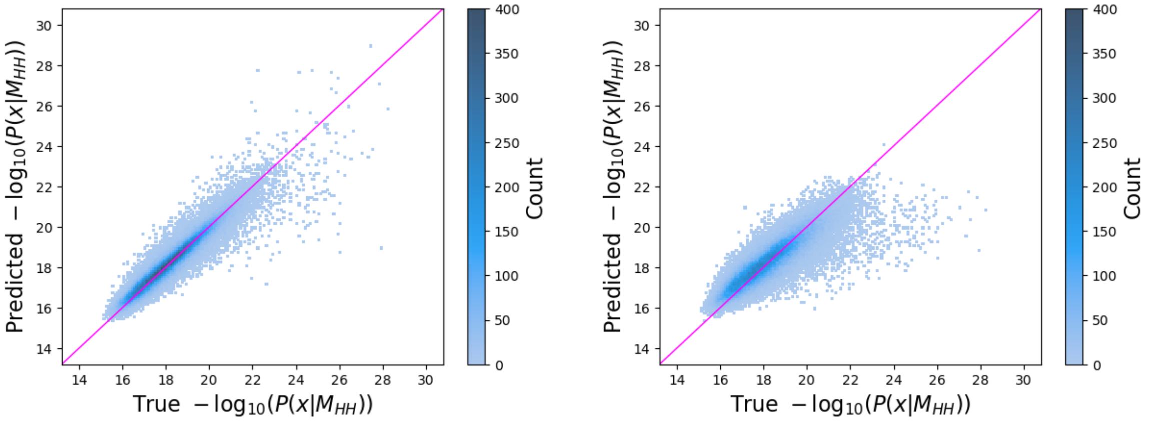
(a) **(b)** **(c)** **(d)**

Figure 3.18: Weight initialisation for the first four layers of the residual deep+narrow neural networks with **(a)** Xavier, without gain; **(b)** Xavier, gain; **(c)** Kaiming, forward pass; **(d)** Kaiming, backward pass.

Understanding the R^2-parameter

While the R^2 parameter itself allows for comparisons between networks, it doesn't actually say anything about whether the network's results are good enough to overcome the challenge being tackled. To understand the results, it's easiest to look at what the predicted values are and compare them to the training data. Fig. 3.19 shows the results for both the best performing deep+narrow network (Fig. 3.19b) and one of the best performing mixed networks (Fig. 3.19a). The mixed network achieves $R^2 \approx 0.91$ and it can be seen that its distribution closely resembles a line through origin. This is particularly important as the weights are being represented in a log scale, so being off of the line through origin can quickly mean that the predicted weight is off by an order of magnitude. For large weights (small values in the plot), the network tends to predict slightly larger weights than MoMEMta calculated. For smaller weights, the network is more accurate. The best deep+narrow network achieves $R^2 \approx 0.76$ — quite a bit less than the mixed network. This difference can also be seen in the distribution of the weights: while it does still resemble a line though origin, it's much broader. The densest part has a width of about an order of magnitude, and the same effect where large weights are predicted to be larger than what MoMEMta provided, but it's much

stronger. The network isn't accurate enough for performing further studies.
This means that for this work, neural networks need to achieve $R^2 > 0.90$ for them to be considered successful and usable. Anything below that won't have the necessary accuracy to reliably predict the weights of the events.



(a)

(b)

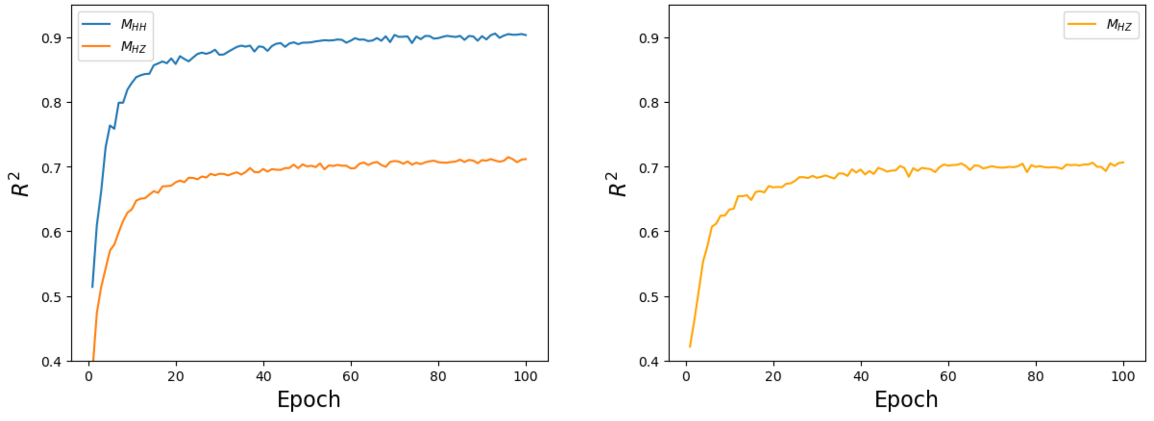
Figure 3.19: Comparison of the MEM weights, on a negative log scale, as calculated by MoMEMTa (x-axis) with those as predicted by **(a)** the residual mixture neural network and **(b)** the residual deep+narrow neural network, both without weight initialisation.

Expanding the Network to two Outputs

In Sec. 3.4.3, a discriminant is defined that allows for a separation between signal and background data. To be able to calculate such a discriminant with the neural network's predictions, the neural network needs to be extended to also provide a prediction for a negative log scale of the MEM weight calculated with the HZ matrix element.This means that it now outputs predictions for both $-log_{10}(P(x|M_{HH}))$ and $-log_{10}(P(x|M_{HZ}))$. To do this, a simple change is applied to the structure of the network: the output layer is given a second node. The input data is the exact same, as the MEM weights for both M_{HH} and M_{HZ} are calculated for the same events. For each event, the network is now given both the weight calculated with M_{HH} and with M_{HZ}, whereas before it was just training with the weight from M_{HH}.

Since the input data hasn't changed and the values of $-log_{10}(P(x|M_{HH}))$ have a similar distribution to that of $-log_{10}(P(x|M_{HZ}))$ (Fig. 3.6a), it can be expected that the network should be able to achieve similarly successful performances for both matrix elements. For this combined network, two R^2-parameters are calculated, one for each output. The goal is to achieve as high a value as possible for both outputs; however, the values of the R^2-parameters should ideally be as close together as possible. Should the values be far part, then the prediction for either HH or HZ will be significantly more accurate than the other — this would lead to the discriminant losing meaningfulness, even if one of the values has a high accuracy.

The results can be seen in Fig. 3.20a and the network clearly does not meet the expectations: while it achieves $R^2 \approx 0.90$ for M_{HH} weights, it performs significantly worse on M_{HZ} weights, with $R^2 \approx 0.70$. The network is able to show near identical success with M_{HH} compared to the previous network that only predicted the weight for M_{HH}. In Fig. 3.21a the discriminant[21] for the neural network with two outputs can be seen. There is significantly less separation between positive and negative values compared to the combined graph in Fig. 3.10a: using the network's predictions, $\approx 35\%$ of the events have a value larger than 0.95, whereas for MoMEMta it is $\approx 30\%$. Additionally, there are hardly any events with a value smaller than -0.95, which is significantly worse than for MoMEMta. By comparing the values that the M_{HH} and M_{HZ} weights of individual events have to one another (Fig. 3.21b), it can be seen that the network almost always predicts the M_{HH} weight to be larger, regardless of the data. This is a stark contrast to the output of MoMEMta (Fig. 3.10b). This could partially be explained by once again looking at the MoMEMta data: for x_{HH}, the weight calculated with M_{HH} is significantly larger than that for M_{HZ} ($\approx 84\%$). For x_{HZ}, while it's usually the case that M_{HZ} results in a larger weight than M_{HH} ($\approx 68\%$); for almost a third of the events it's the other way around. This means that in most events ($\approx 60\%$ of total) the weight for M_{HH} is larger than for M_{HZ}, creating a bias. This could lead the neural network to lean towards predicting a higher weight to M_{HH} than M_{HZ}, even when that isn't correct. However, a neural network should be able to recognise such a bias and predict accordingly, so this doesn't satisfyingly explain the results.



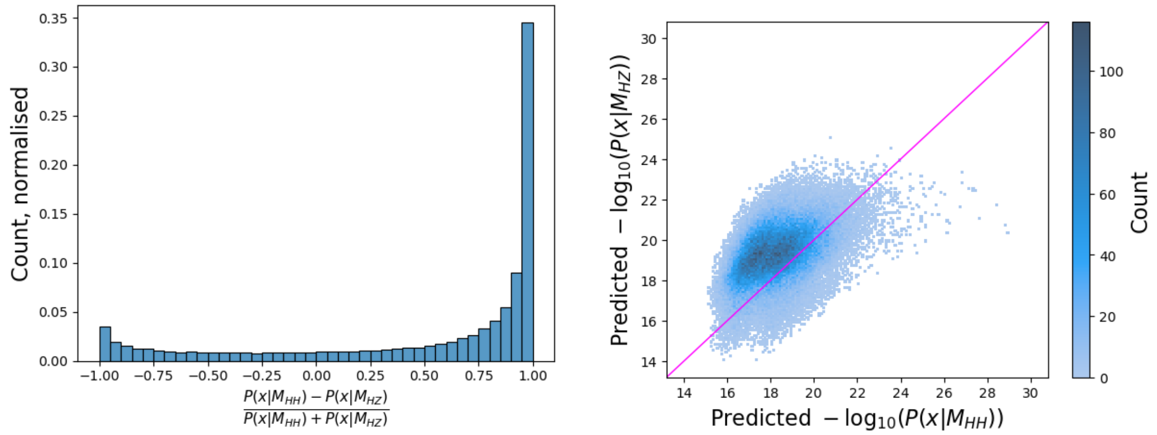
(a) **(b)**

Figure 3.20: **(a)** R^2 parameter values for the combined M_{HH} and M_{HZ} network, split into the individual outputs. **(b)** R^2 parameter value for single-output network with M_{HZ}.

This raises a few questions: first, does the low R^2 value have to do with having one network predicting both weights, or is there an issue with predicting the weights for M_{HZ} itself? To test this, the exact same (single output) network, as was used in Fig. 3.19a, is run, except now it's being trained with $-log_{10}(P(x|M_{HZ}))$. The result of this can be seen in Fig, 3.20b. The network with one output achieves the same low R^2 values for M_{HZ} as the network that predicts both weights.

This leads to the next question: do these low values simply come from the neural network structure not being optimal for the M_{HZ} weights? This question will be

[21] Defined in Eq. 3.2.



(a)

(b)

Figure 3.21: **(a)** Discriminant for the weights predicted by the feed-forward network with dual output (same as in Fig. 3.20a). **(b)** Comparison of the weights predicted for each event. The x-axis describes the prediction trained with M_{HH}, the y-axis that with M_{HZ}. The magenta line indicates the line through origin, where the network predicts both weights to be of equal value.

answered later, when a convolutional neural network is used to predict the weights. Lastly, there is one more possibility that could explain these results: does the weight calculated with M_{HZ} have more variance in its relation to the input variables compared to the M_{HH} weight? In other words, do events with very similar values for the M_{HZ} weights have more fluctuations in the phase space compared to events with very similar values for the M_{HH} weights? If so, then it would be harder for a neural network to find patterns between the input data and the M_{HZ} weight compared to the M_{HH} weight, which would lead to the predictions for the M_{HZ} weight being worse.

Answering this last question is not that easy. The phase space consists of 24 parameters, so it can be assumed that understanding how fluctuations affect the M_{HZ} weight requires not just looking at the parameters individually, but also their relations to one another. Finding relations between 24 parameters is difficult, so to simplify the task, only the four-momentum of a single b quark is regarded[22]. The idea is to compare events that have very similar weights, specifically $P(x|M) = W_M \pm 0.2$, where W_M is a value for a weight that can be chose arbitrarily. For these events, the four-momentum parameters of the single b quark are compared pair-wise. This will give an idea of what the phase space of one b quark looks like around W_M. This is not just done for M_{HZ} ($W_{M_{HZ}}$), but also for M_{HH} ($W_{M_{HH}}$), so that their fluctuations can be compared.
An issue becomes clear when looking at distributions for for different values of $W_{M_{HZ}}$: the shape of the distributions of the four-momentum parameters for the selected events are dependant on the value of $W_{M_{HZ}}$. Events with $W_M = 16.0$ (large weight) will show different behaviours compared to events with $W_M = 22.0$ (small weight). This by itself makes it difficult to understand the fluctuations for M_{HZ}, since the choice of $W_{M_{HZ}}$ influences how much fluctuation is present (events with smaller weights have more fluctuations).

[22] This is once again based on the assumption that any larger differences between M_{HH} and M_{HZ} will be found in the b quarks.

Additionally, it isn't easy to compare the behaviours of M_{HH} and M_{HZ}, since their weight distributions are different (Fig. 3.6a). As a result, $W_{M_{HH}}$ and $W_{M_{HZ}}$ require different values to find regions in the respective phase spaces that represent comparable behaviours. But, this also isn't so simple: does it make more sense to compare events that represent the peaks of the M_{HH} and M_{HZ} weights distributions, or is it better to compare events that have the largest weights for either M_{HH} or M_{HZ}? What will give more useful information, and how should this information be interpreted?

Ultimately, trying to effectively understand how fluctuations in the phase space affect the MEM weights requires a more in-depth analysis, which, unfortunately, cannot be done within the scope of this work due to time constraints.

Time Comparison

The reason for using neural networks to perform an analysis using the matrix element method is to reduce computing resources. As was previously discussed in Sec. 3.4.3, the total calculation time for MoMEMta is $\approx 1.02 \times 10^4$ hours. Together, the training and verification process, which ran on a single node of the Slurm cluster, required ≈ 803 min 32 sec, which is ≈ 13.4 hours. This means that calculation of the MEM weights with MoMEMta took about 761 times longer than the neural network needed for training and validation. This shows how much time and computing resources can be saved by using a neural network on a large dataset. It isn't possible to escape requiring an MEM calculation with software such as MoMEMta, since the training data for the neural network still needs to be generated. That being said, the measured data of an experiment consists of vastly more events than the training data, which means that the MEM will have to be performed on much fewer events when using a neural network. So, while it has been shown that training a neural network to predict MEM weights can successfully be used to greatly reduce computational resources and time, the issue still remains that the neural networks currently struggle to predict the weights calculated with M_{HZ}.

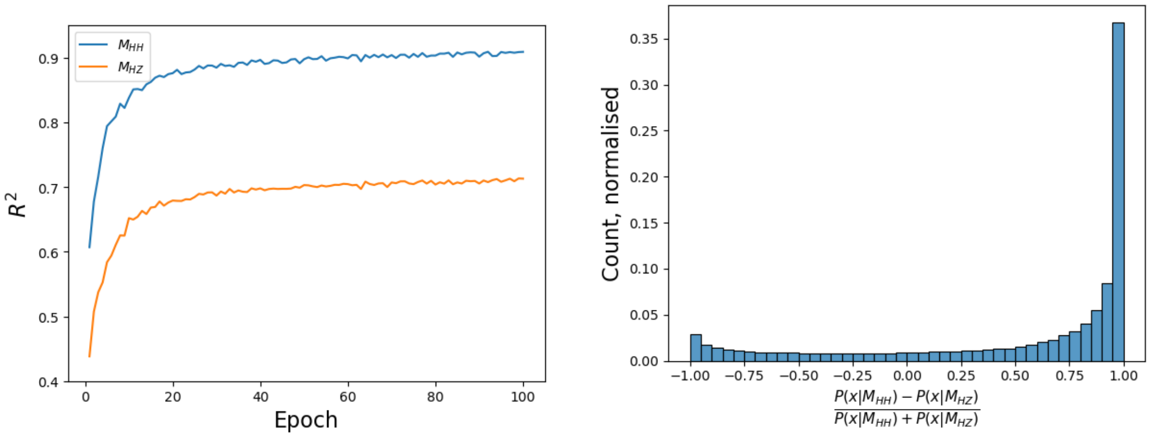
Additional Input and Rotationally Symmetric Parameters

A part of this analysis is to only use the four-momentum of the final-state particles as input parameters to study how well the MEM can be replicated by a neural network when given the same input. However, as was discussed in Sec. 3.5, a neural network isn't limited to the four-momenta as input parameters. In Sec. 3.4.3 two parameters are presented that distinguish between events where the values of the M_{HH} and M_{HZ} weights align with expectations[23] and events where they do not. These parameters are $(p_x^{b_1} - p_x^{b_1})^2 + (p_y^{b_1} - p_y^{b_1})^2$ and $(p_x^{b_1} - p_x^{b_1})^2 + (p_z^{b_1} - p_z^{b_1})^2$. Although they contain information that can already be found in the four-momenta, it isn't clear if this information is obfuscated from the neural network. Therefore, to test whether they can be used for improvement, the best performing feed-forward network, that was used in Fig. 3.21, has been run again with the additional parameters in the input (leading to a total number of 26 input parameters). The result can be seen in Fig. 3.22.

The performance of the network is virtually the same as without the additional parameters. This means that the network must already be able to get all the information

[23] Large weights for when data and matrix element align, small weights when they do not.

that can come from these additional parameters through the basic four-momenta. This isn't that surprising, since the MEM calculates the weights with just the four-momenta, so, in theory, everything a network should need is already provided. Since neither the rotationally symmetric input parameters nor these additional parameters are able to give the network an edge, it can be concluded that parameters that would be able to increase the network's performance will have to share very little information with the four-momenta.



(a) **(b)**

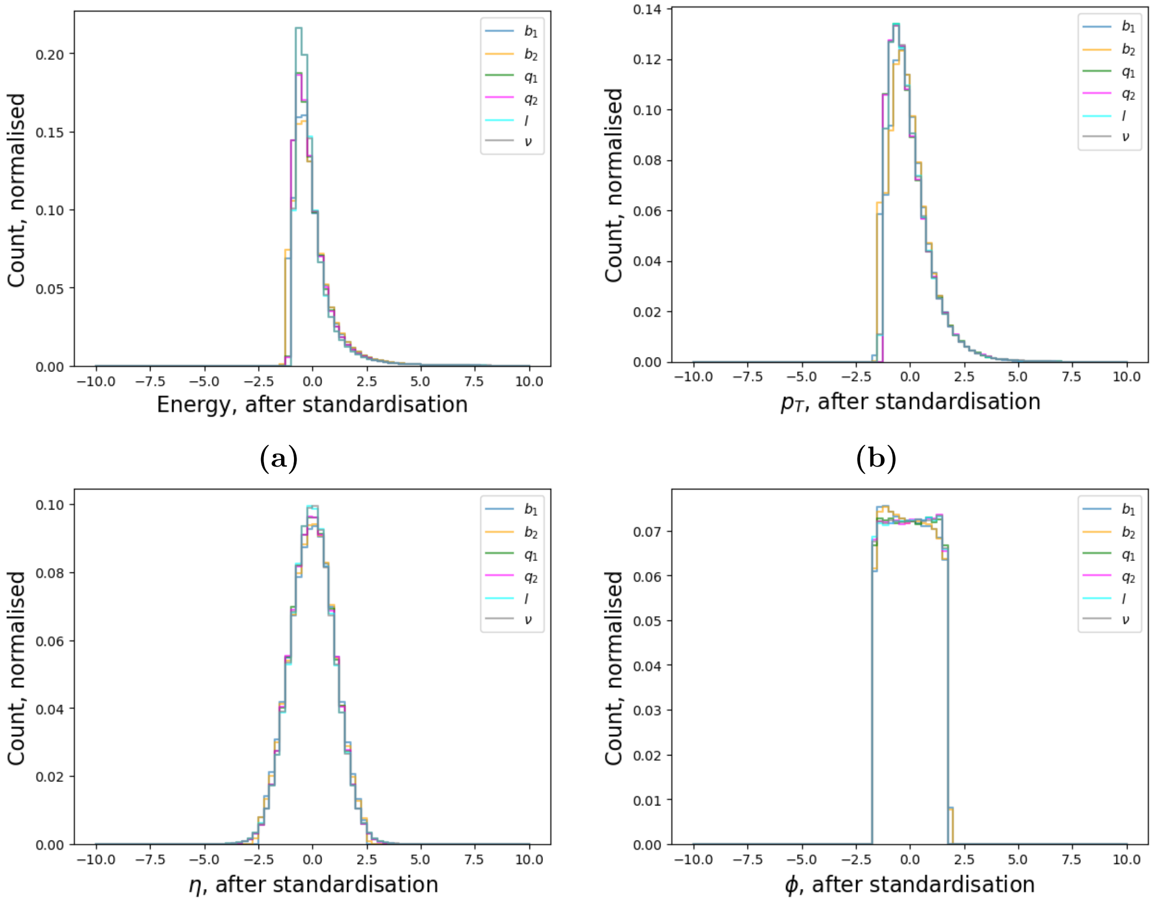
Figure 3.22: Results of the feed-forward network with 26 input parameters: **(a)** value of the R^2-parameter and **(b)** the discriminant.

A network is tested using the rotationally symmetric input parameters, the normalised versions of which can be seen in Fig. 3.23. The scaling factors for the normalisation are as follows: for the energy $f_E = 1000$, for the transverse momentum and pseudorapidity parameter $f_{p_T,\eta} = 100$, and for the polar angle $f_\theta = 10$. Again, the distributions of b_1 and b_2 stand-out the most. Apart from the energy, which is the same as for the four-momentum data, differences between the particles can mostly be seen in the azimuth angle ϕ: while all other particles have a flat distribution, the two b quarks have uneven distributions which favour negative values.

The same network as in Fig. 3.21 is run with the rotationally symmetric input parameters; the results can be seen in Fig. 3.24. With $R^2 = 0.74$ for M_{HH} and $R^2 = 0.58$ for M_{HZ}, this network performs significantly worse than for the four-momenta. It's a bit surprising that the difference in performance is so drastic, as rotationally symmetric parameters remove unnecessary information, while still precisely describing the particles' positions in relation to each other. There is no current explanation for these results.

3.6.3 Convolutional Neural Network

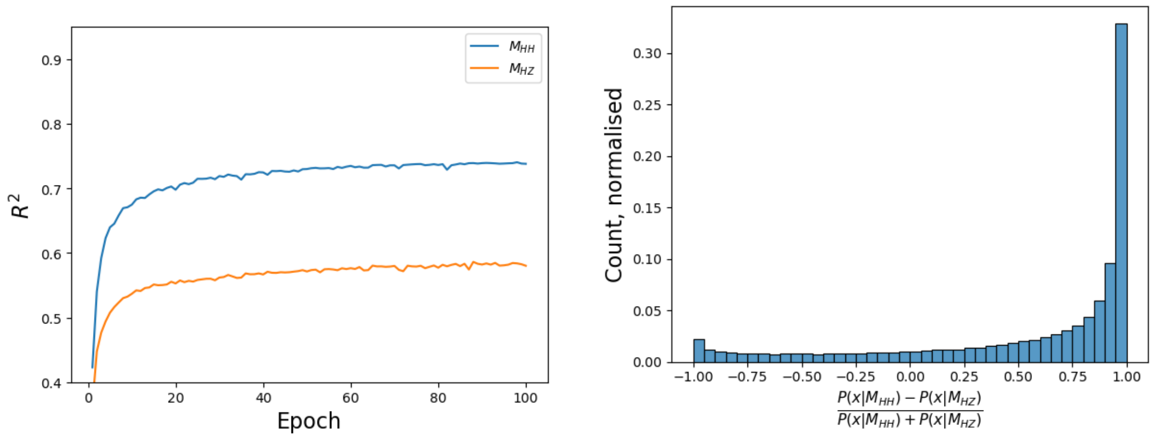
CNNs excel at recognising isolated structures within a larger set of data points. Since the four-momenta of the jets are related to each other, there could be structural patterns that a CNN might be able to pick-up. For this, the inputs variables are structured to resemble an image: each row contains the four-momentum of a single final-state particle and each column contains the same four-momentum parameter (see Tab. 3.3).



(a) (b)

(c) (d)

Figure 3.23: Normalised distributions for both x_{HH} and x_{HZ} which construct the rotationally symmetric data set. **(a)** Energy, **(b)** Transverse momentum, **(c)** Pseudorapidity, **(d)** Azimuth angle.



(a) (b)

Figure 3.24: Results of the feed-forward network with rotationally symmetric input parameters: **(a)** value of the R^2-parameter and **(b)** the discriminant.

Since the placement of the jets in the matrix is the same for every event, this means that identifiable structures will always appear in the same places. Therefore, the full advantage that a CNN can have over other types won't be given an opportunity to shine, but that doesn't mean that a convolutional network can't have improved capability over a feed-forward network.

E^{b_1}	$p_x^{b_1}$	$p_y^{b_1}$	$p_z^{b_1}$
E^{b_2}	$p_x^{b_2}$	$p_y^{b_2}$	$p_z^{b_2}$
E^{q_1}	$p_x^{q_1}$	$p_y^{q_1}$	$p_z^{q_1}$
E^{q_2}	$p_x^{q_2}$	$p_y^{q_2}$	$p_z^{q_2}$
E^{lep}	p_x^{lep}	p_y^{lep}	p_z^{lep}
E^{ν}	p_x^{ν}	p_y^{ν}	p_z^{ν}

Table 3.3: Structure of the input data for the CNN. All variables are saved as a matrix, where each row is designated to a final-state particle. The indices on the quarks indicate that the order of the matrix relates to the event reconstruction: the particle that had a jet assigned to it first is placed before the other in the matrix (see Sec. 3.3).

Like for the feed-forward network, the number of layers for a CNN needs to be tuned. Unlike a feed-forward network, which just uses linear layers, there are three types of layers: convolutional layers, pool layers and linear layers. To understand how strongly the convolutional and pool layers affect the network's performance, multiple set-ups are tested (see Tab. 3.4). All convolutional layers have 10 in-channels and 10 out-channels. Due to the shape of the input matrix being 6×4, there can only be up to two pool layers, as the pool layer has a stride of 2, which reduces each dimension by half. In the cases where there is only one pooling layer, the output has to be flattened for the linear layer, as it requires the dimensions $X \times 1 \times 1$[24]. The final linear layer exists without an activation function so that the output can achieve any value, allowing for the prediction of the MEM weight.

Network A	**Network B**	**Network C**	**Network D**
Conv. (input)	Conv. (input)	Conv. (input)	Conv. (input)
Conv.	Conv.	Pool	Conv.
Pool	Conv.	Conv.	Pool
-	Conv.	Pool	Conv.
-	Pool	-	Conv.
-	-	-	Pool
Linear	Linear	Linear	Linear
Linear (output)	Linear (output)	Linear (output)	Linear (output)

Table 3.4: Structures of CNNs that are used to get a better understanding of the impact on convolutional and pool layers.

The results of each network can be seen in Fig. 3.25a, where each network has the following configuration:

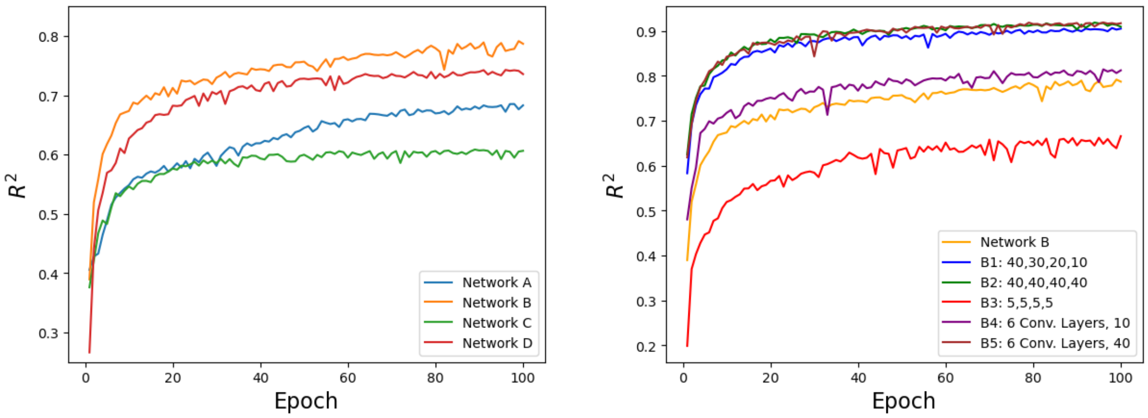
[24] X can be any integer, within reason.

- batch size: 32
- convolution in-channels: 10
- convolution out-channels: 10
- convolution kernel size: 3
- convolution stride: 1
- convolution padding: "same"
- pool kernel size: 2
- pool stride: 2
- nodes on linear layer: 40
- learning rate: 10^{-5}
- momentum: 0.995
- weight decay: 10^{-8}
- dampening: 0
- Nesterov: True
- activation function: LeakyReLU
- negative slope = 0.1
- optimisation: SGD

There are significant differences in the performances of the networks: first, having two pooling layers reduces the network's capability. This can been seen by comparing Network A (shallow, single pool, blue) to Network C (shallow, double pool, green) and by comparing Network B (deep, single pool, orange) to Network D (deep, double pool, red). Second, having more convolutional layers (i.e. having a deeper network) increases the performance.

To get a better understanding out how the number of channels at each layer affects the network's success, multiple versions of Network B with different numbers of channels are compared (see Fig. 3.25b). For Network B1, the number of channels is increased to 40 in the first layer and decreases in steps of 10, leading to the last layer having 10 channels again. This is comparable to the best performing feed-forward network, which has a high number of nodes in the first layer, which then decreases. For Network B2, the number of channels is increased to 40 in each layer and for Network B3, they are decreased to 5 in each layer. Network B4 and B5 are different: since Fig. 3.25a shows that increasing the number of convolutional layers improves the network's capability, for Network B4 and B5 the number of convolutional layers has been increase to 6, where each layer has 10 channels for B4 and 40 channels for B5.
The three best performing networks are: B1, with a decreasing number of channels per

layer (blue), B2, with 40 channels at all four layers (green) and B6, with 40 channels at all six layers (brown). All three have around the same performance, which is significantly better than the rest. This indicates that the most important factor is having a large number of channels at the input layer, which resembles the same findings as for the fast-forward network (see Fig. 3.14). In the rest of the analysis, B2, with 40 channels and four layers, is used, as it has a slightly better performance than B1 and a quicker training time than B6.

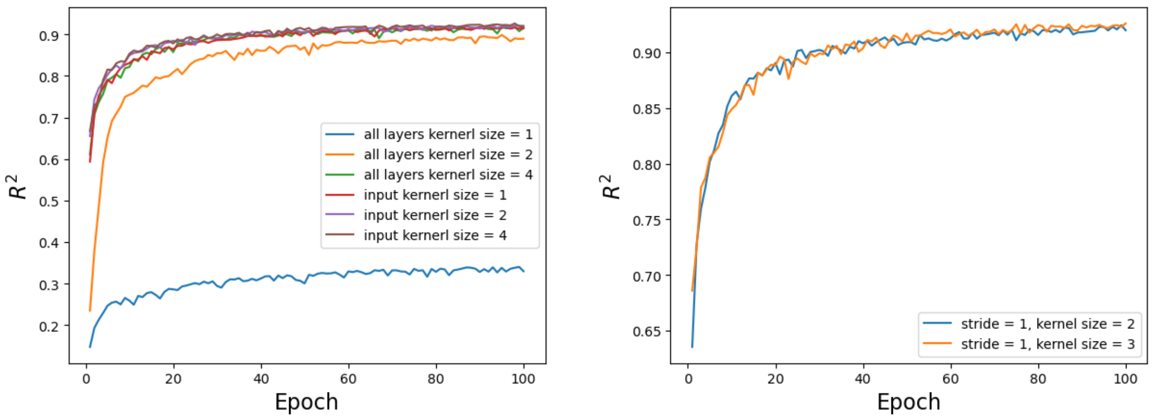


(a) **(b)**

Figure 3.25: **(a)** R^2 values for the **CNNs!** described in Tab. 3.4. **(b)** Different configurations for Network B.

The next hyperparameters that are compared are the kernel sizes of both the convolutional layers and the pool layer, and the stride of the pool layer. Changing the stride of the convolutional layer to a value other than 1 requires a padding other than ”same”, which would then affect the rest of the network’s structure. Therefore, the stride of the convolutional layer is left untouched. For the kernel size of the convolution, two approaches are taken: adjusting the kernel size of just the input layer and adjusting it for all layers. This is to study how much of an effect just the input layer has, since, as has previously been shown, the input layer seems to have by far the most impact on the network’s performance. Up until now, the kernel size of all convolutional layers has been 3. First, networks are run where just the input layers have kernel sizes of 1, 2 or 4, whereas all the other layers keep their kernel size of 3. Then, networks where convolutional layers have a kernel size of 1, 2 or 4, are tested. The results can be seen in Fig. 3.26a.
The best performing networks have $R^2 \approx 0.90$, but it isn’t clear how the configurations impact the networks: the network where each convolutional layer has a kernel size of 4 performs just as well as the network where the first layer has a kernel size of 1 and the other convolutional layers have a kernel size of 3. Even though there structure are quite different, they have the same performance. There doesn’t seem to be a “correct” choice that causes a network to perform better than all others. However, the the wrong choice of kernel size can have a very negative impact on the network (e.g. all layers having a kernel size of 1).

After comparing kernels sizes, a couple of networks are tested in which the stride of the pool layer is set to 1 (up until here it has been 2). For this, the network with a kernel size of 2 in the input layer and 3 in the other convolutional layers is used as a base.



(a) **(b)**

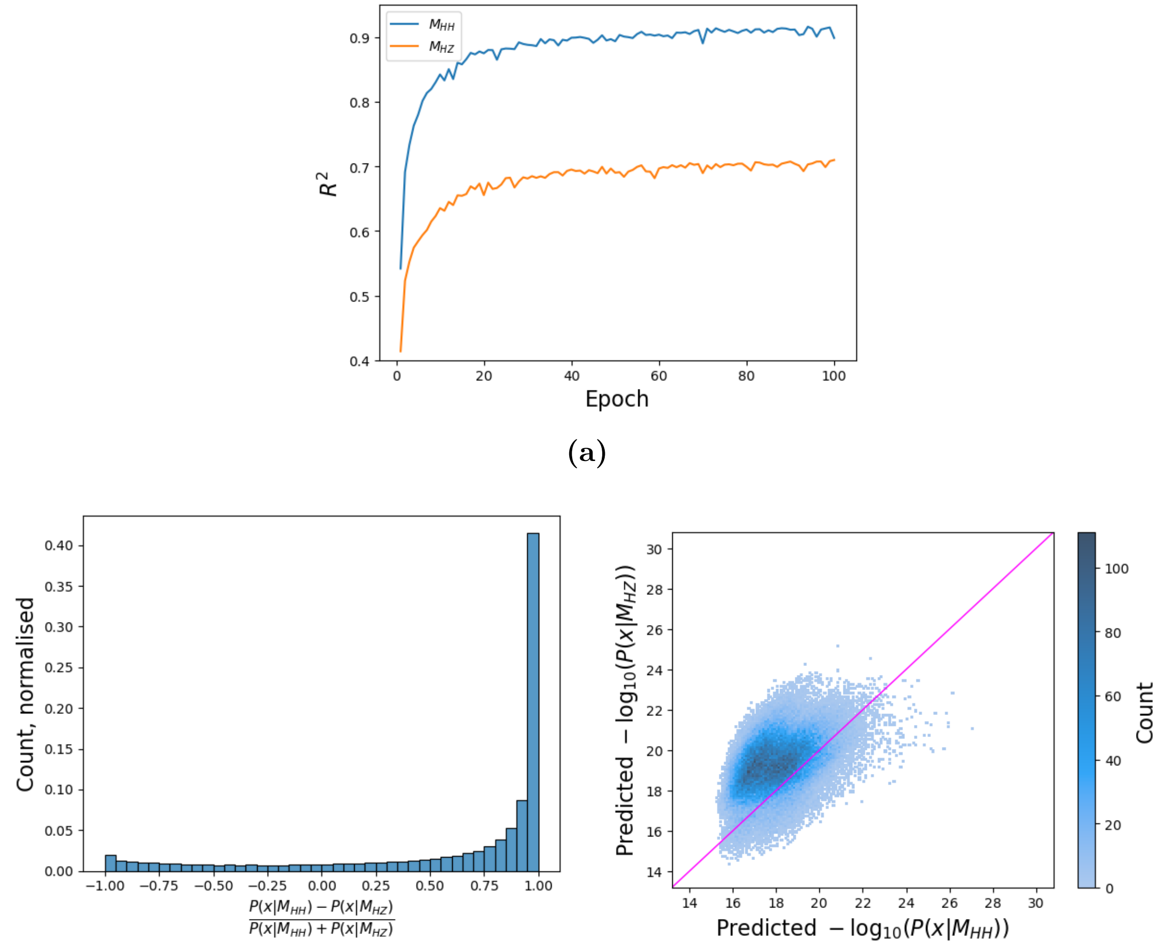
Figure 3.26: R^2 parameter values for networks with adjusted **(a)** convolution kernel size **(b)** strides for both the convolutional layers as the pool layer.

This network, like the others so far, has a convolutional stride of 1 and a pool stride of 2. Changing the pool stride to 1 impacts the number of output parameters: when the output of the pool layer is flattened, a stride of 1 results in 600 output channels (each with only a single value), which is passed to the first linear layer. In comparison, when the pool stride is 2, the flattened output contains 240 output channels. One of the two networks that are tested has a pool kernel size of 2, the other has a pool kernel size of 3. The results can be seen in Fig. 3.26b. The best performing networks are those with a pool stride of 1, with $R^2 \approx 0.93$. The likely reason that this performs better is that a small stride maintains more information. The idea behind having a stride of 2 is specifically to reduce information, keeping that which is deemed most relevant and making the job of the linear layer easier. However, it seems having more information is better, as even the information that is less relevant still contains something that is useful to the network.

Expanding the Network to two Outputs

In the following, the network that is used has a kernel size of 2 in the input layer and 3 for the other convolutional layers, with the pool layer having a kernel size of 3 and a stride of 1. The network is trained with both M_{HH} and M_{HZ} weights and is therefore expanded to two output variables, such as was done with the feed-forward network. The results can be seen in Fig. 3.27. For the M_{HZ} weights, the network achieves $R^2 \approx 0.71$. For the M_{HH} weights, it achieves $R^2 \approx 0.90$ for the validation data; however, for the training data it achieves $R^2 \approx 0.92$. As previously stated, in this analysis, networks that have a difference between the R^2_{train} and R^2_{val} that is smaller than 1 are considered to not overfit. This means that the network overfits slightly on the M_{HH} weights when it trains with both M_{HH} and M_{HZ} weights, which it didn't do when it just trained with the M_{HH} weights.
The shape of the discriminant's distribution (Fig. 3.27a) is heavily skewed compared to the discriminant from the MoMEMta weights. Around 42% of events have a value above 0.95 and a peak for values less than -0.95 is barley visible. In Fig. 3.27b, it can be seen that for most events, the predicted value for M_{HH} is larger than that of M_{HZ};

in many cases even close to two orders of magnitude. This shows that the network still has a lot of room for improvement, since while it is picking-up that the M_{HH} weight is generally larger, it's over-valuing this fact and it isn't learning when the M_{HZ} weight should be larger.



(a)

(b) **(c)**

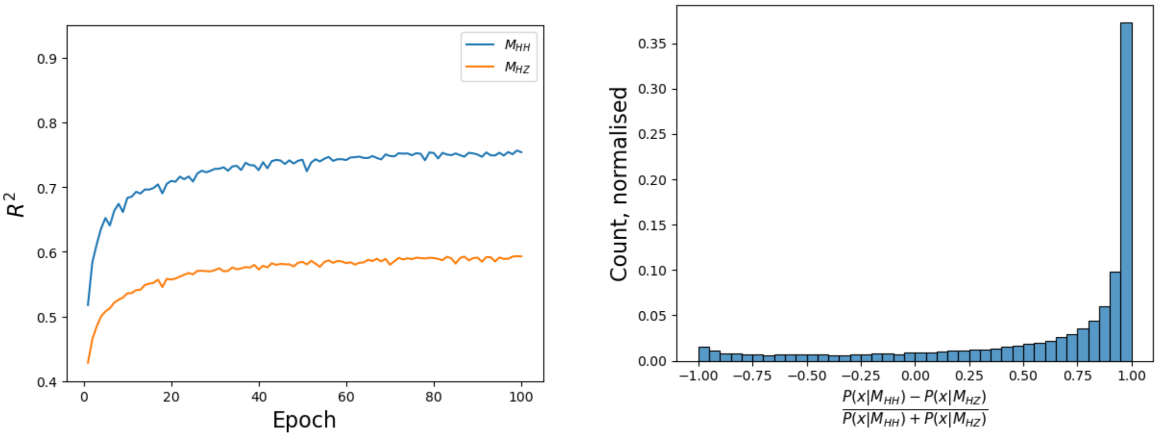
Figure 3.27: **(a)** Value of the R^2 parameter for M_{HH} and M_{HZ} weights predicted by the convolutional network with dual output. **(b)** Discriminant for the weights. **(c)** Comparison of the weights predicted for each event. The x-axis describes the prediction trained with M_{HH}, the y-axis that with M_{HZ}. The magenta line indicates the line through origin, where the network predicts an event to be equally likely for either matrix element.

Time Comparison

The time required to train and validate the CNN depicted in Fig. 3.27 is ≈ 23.1 hours. This is about 442 times quicker than MoMEMta needed to calculate the weights and about 1.72 times slower than the best-performing feed-forward neural network. While it's noticeably slower than the feed-forward network, it's still much, much quicker than running MoMEMta and therefore a success in reducing the computation time.

Rotationally Symmetric Parameters

Due to the convolutional neural network having a 6 × 4 × 1 input matrix, it isn't possible to add the two additional parameters (Sec. 3.4.3) to the four-momenta as input parameters. However, the convolutional network can take the rotationally symmetric parameters; their scaling is the same as described in Sec. 3.6.2. The results can be seen in Fig. 3.28 and with $R^2 = 0.75$ for M_{HH} and $R^2 = 0.59$ for M_{HZ}, the results are significantly worse than for the four-momenta.



(a) **(b)**

Figure 3.28: Results of the CNN with rotationally symmetric input parameters: **(a)** value of the R^2-parameter and **(b)** the discriminant.

3.6.4 Discussion and Outlook

The fact that both the feed-forward and the convolutional networks perform significantly worse when predicting the M_{HZ} weights indicates that the reason for this big discrepancy most likely comes from the HZ data that is depicted in Fig. 3.10. Since both network types struggled with the same issue in the same way, it can be assumed that the issue likely doesn't lie in the network types or their structures. While there may be network types that can achieve better results, they are likely to have the same issue with predictions for M_{HZ} weights performing significantly worse than for M_{HH} weights.
That being said, the neural networks are able to achieve a satisfying accuracy when it comes to predicting the M_{HH} weights. Therefore, the set-up of these networks can be used to correctly assign HH events a high weight. While this means that applying a cut on the M_{HH} weight could be used to reduce background data without removing (much) signal data, it can't currently be used to separate background events from signal entirely. For this, a strong separation in the discriminant (Fig. 3.21a, 3.27b) is required, which is not what was achieved.

The neural networks have all been run using just the four-momenta as input data. As was discussed in Sec 3.5, this doesn't need to be the case: the networks can take any variables as input data, as long as they are coupled to the events. Both the feed-forward and convolutional network could not be improved by using rotationally symmetric input variables, that are based off of the four-momenta. Instead, they performed significantly worse.

The feed-forward network was also tested with two additional parameters, $(p_x^{b_1} - p_x^{b_1})^2 + (p_y^{b_1} - p_y^{b_1})^2$ and $(p_x^{b_1} - p_x^{b_1})^2 + (p_z^{b_1} - p_z^{b_1})^2$, that indicate differences between events with a good separation between M_{HH} and M_{HZ} weights and events with a bad separation. However, adding these parameters to the input parameters of the feed-forward network doesn't change its performance, meaning that the network is already able to extract the information they provide from the four-momenta.

The best performing CNN achieves $R^2 \approx 0.90$ for M_{HH} weights and $R^2 \approx 0.71$ for M_{HZ} weights. It uses four convolutional layers followed by a pool layer and two linear layers, the second of which is the output layer. The best performing feed-forward network achieves $R^2 \approx 0.90$ for M_{HH} weights and $R^2 \approx 0.70$ for M_{HZ}. After the input layer, it has three hidden layers, with a decreasing number of nodes on each layer.

Overall, it isn't clear which of the two network types, feed-forward or convolutional, is a better choice for implementing the MEM. The convolutional network achieved a slightly better performance, with the highest R^2 value when training on just M_{HH}, and very similar values to the feed-forward network when training on both M_{HH} and M_{HZ} weights. Since the convolutional network slightly overfits when training on both weights, it can be assumed that there is still a little bit more potential in its capabilities. However, while the discriminant of the convolutional network has a similar shape to that of the feed-forward network, where the peak at $D_{\pm} < -0.95$ has almost vanished, the peak at $D_{\pm} > 0.95$ is larger, at almost 42% of the events. This means that the convolutional network is even more likely to predict the M_{HH} weights to be larger than they should be compared to the feed-forward network. This difference can also be seen when comparing the distribution of the weights for the convolutional network (Fig. 3.27c) to the feed-forward network (Fig. 3.21b): the densest region of the distribution is shifted further to the left for the convolutional network.

From this, it can be seen that both neural networks types need more work before they can be used in an analysis. That being said, both have shown results that head in the right direction. More testing, possibly with other network types, needs to be done to be able to reach the full potential of combining the MEM for Higgs pair production with neural networks. This testing will need to include developing a better understanding of the relation between the final-state four-momenta and the resulting weights.

Both network types had significantly quicker run times for training and validation compared to what MoMEMta requires for calculating the M_{HH} and M_{HZ} weights. This shows that if it's possible to increase the accuracy of the networks' predictions for M_{HZ}, then implementing the MEM with neural networks to reduce computing resources will be a viable practice.

Conclusion

In this thesis, the matrix element method has been implemented with neural networks to search for Higgs pair production using simulated data.
The two signal decay modes consist of Higgs self-coupling and general Higgs pair production via a box diagram. Significant for this analysis is that one Higgs boson decays into b quarks while the other decay into W bosons. The background decay mode contains a Higgs boson and a Z boson, where the Z boson decays into b quarks and the Higgs to W bosons. The generation of the simulated data and the jet reconstruction are done by hand. The weights for the Matrix Element Method (MEM) are calculated with the dedicated program MoMEMta using two different matrix elements: one for the Higgs pair production, M_{HH}, and one for the background, M_{HZ}. The resulting weights for the HH data mostly have the expected values, where weights calculated with M_{HH} are larger than those with M_{HZ}. For the HZ data, there is not as clear of a distinction of M_{HZ} weights being larger than M_{HH} weights. The data have been studied at depth to understand why true HZ events occasionally yield larger matrix element weights for M_{HH} than for M_{HZ}. It has been found that the the separation power strongly relates to the four-momenta of the two b quarks. A deeper investigation of the theoretical calculations yielding the matrix element formulae in order to resolve their relation to the b quark four-momenta was beyond the scope of this research.

Two types of neural networks have been studied for this analysis: feed-forward and convolutional networks. For each, various structures with different hyperparameters have been tested. The best-performing feed-forward network achieved $R^2 \approx 0.90$ on M_{HH} weights and $R^2 \approx 0.70$ on M_{HZ} weights; the best-performing convolutional network achieved $R^2 \approx 0.90$ on M_{HH} weights and $R^2 \approx 0.71$ on M_{HZ} weights. While this is a success with regards to the M_{HH} weights, the networks are still open for improvements for HZ event weights. The reason for the somewhat lower performance of the prediction of the M_{HZ} weights lies in the less distinguished training data calculated with MoMEMta.

In an attempt to improve the networks, in particular with regards to M_{HZ}, two cases, one with rotationally symmetric input parameters and one where two additional input parameters derived from the four-momenta, are tested. The feed-forward network's performance remains the same for the additional parameters, indicating that they already learned all the information that can be gained from the additional four-momenta, and both networks perform significantly worse with the rotationally symmetric input parameters.

Appendix A

Detailed Mathematical Description of the Standard Model of Particle Physics

Sec. A.1 and A.2 are oriented towards [21].

A.1 Quantum Field Theory

To derive the SM, one must first move from describing points in space, as is done in classical mechanics, to regions of space. This means that instead of calculating positions as a function of time $\mathbf{x}(t)$, systems are calculated using functions of position and time $\psi(t, \mathbf{x})$. These are known as fields — an example would be describing the movement of a body of water, instead of an individual water particle. In quantum field theory, a Lagrangian density $\mathcal{L}(\phi_i, \partial_\mu)$ (in the following simply referred to as a Lagrangian) is a function of fields ψ_i, and its derivatives in time and space are $\partial_\mu \phi_i \equiv \frac{\partial \phi_i}{\partial x^\mu}$ [24].

Hamilton's Principle (also known as the Principle of Least Action) states that the evolution of a system from time t_1 to t_2 is such that the action

$$S = \int_{t_1}^{t_2} \mathcal{L}(\phi_i, \partial_\mu \phi_i)\, d^4x \tag{A.1}$$

has a stationary value for the actual "path" [76, 77]. In less formal terms, this means that the path is chosen for which infinitesimal increments along the path cause minimal change to the value to S, i.e. $\delta S = 0$. From this, the Euler-Lagrange equations of motion

$$\partial_\mu \left(\frac{\partial \mathcal{L}}{\partial (\partial_\mu \phi_i)} \right) = \frac{\partial \mathcal{L}}{\partial \phi_i} \tag{A.2}$$

can be derived [76]. Once the Lagrangian for a particle type has been established, the Euler-Lagrange equation will result in the equations of motion for that particle type [24].

Gauge invariance is the concept that a Lagrangian should maintain its form when a phase is applied to the Lagrangian's field(s). A phase transformation represents a change to the field, and there are two types: global and local. A global phase transformation is applied to all points in a system equally. A local phase transformation is also applied to all points; however, each point is influenced differently by the new phase, depending on its coordinates. Mathematically, a global gauge transformation is described as the following:

$$\psi \rightarrow e^{i\alpha}\psi, \tag{A.3}$$

where α is independent of spacetime, which means that it is unaffected by the derivative of the field:

$$\partial_\mu \psi \rightarrow e^{i\alpha}\partial_\mu \psi. \tag{A.4}$$

Since the derivative of a field isn't affected by global gauge transformation, accordingly, the Lagrangian remains unaffected. The result of this is that the equations of motion, that are generated by the Euler-Lagrange equation (A.2), are also unchanged. In other words, a global gauge transformation is gauge invariant: it doesn't affect the laws of physics.

This is not so simple when it comes to local gauge transformations, as the phase $\alpha\left(\boldsymbol{x}\right)$ is now dependent on spacetime:

$$\psi \rightarrow e^{i\alpha(\boldsymbol{x})}\psi \equiv \tilde{\psi}. \tag{A.5}$$

When the derivative is applied to the transformed field $\tilde{\psi}$, an additional term appears:

$$\partial_\mu \tilde{\psi} = \partial_\mu \left(e^{i\alpha(\boldsymbol{x})}\right)\psi + e^{i\alpha(\boldsymbol{x})}\partial_\mu \psi = i\left(\partial_\mu \alpha\left(\boldsymbol{x}\right)\right)e^{i\alpha(\boldsymbol{x})}\psi + e^{i\alpha(\boldsymbol{x})}\partial_\mu \psi. \tag{A.6}$$

This leads to a constraint: when constructing a Lagrangian to describe a physical theory, it must be expanded by an additional field that transforms such that it causes the additional term in Eq. A.6 to be cancelled. This can, for example, be done by adding an addition term that contains the local phase [24].

A.2 Gauge Fields and the Introduction of Particles

Now that the groundwork has been laid, it can be applied to the Dirac Lagrangian[1], which describes spin-$\frac{1}{2}$ particles[2]:

$$\mathcal{L} = i\bar{\psi}\gamma^\mu \partial_\mu \psi - m\bar{\psi}\psi, \tag{A.7}$$

[1] While the Dirac, Proca and Klein-Gordon equations [24, 25] can be derived mathematically, their Lagrangians are derived axiomatically [24].

[2] Using natural units: $\hbar = 1$, $c = 1$.

where $\bar{\psi}$ is the adjoint spinor of ψ[3]. If the field in Eq. A.7 were to transform like Eq. A.5, then the derivative of the field in the first term of the Lagrangian would transform like Eq. A.6, resulting in

$$\mathcal{L} \to \mathcal{L} - \left(\partial_\mu \alpha\left(\boldsymbol{x}\right)\right) \bar{\psi}\gamma^\mu \psi \tag{A.8}$$

which clearly is not invariant under local phase transformation. To ease the choice of how the new field (with which the Lagrangian will be expanded) should transform locally, the phase $\alpha\left(\boldsymbol{x}\right)$ can be defined as

$$\alpha\left(\boldsymbol{x}\right) = -q\lambda\left(\boldsymbol{x}\right) \tag{A.9}$$

which redefines the local phase transformation to

$$\psi \to e^{-iq\lambda(\boldsymbol{x})}\psi \tag{A.10}$$

and results in Eq. A.8 being rewritten as

$$\mathcal{L} \to \mathcal{L} + q\bar{\psi}\gamma^\mu\psi\partial_\mu\lambda\left(\boldsymbol{x}\right). \tag{A.11}$$

This Lagrangian is also not invariant under a local gauge transformation. To reconcile this problem, a vector field A_μ (referred to as a *gauge field*) is introduced that transforms locally with

$$A_\mu \to A_\mu + \partial_\mu\lambda. \tag{A.12}$$

Implementing Eq. A.9 into Eq. A.6 results in

$$\partial_\mu\psi = e^{-iq\lambda(\boldsymbol{x})}\left(\partial_\mu - iq\left(\partial_\mu\lambda\right)\right)\psi. \tag{A.13}$$

Since the transformation of the gauge field was chosen so that it contains $\partial_\mu\lambda$, a substitution for the partial derivative can be introduced, known as the *covariant derivative*

$$\mathcal{D}_\mu = \partial_\mu + iqA_\mu, \tag{A.14}$$

which transforms with

$$\mathcal{D}_\mu \to \partial_\mu + iq\left(A_\mu + \partial_\mu\lambda\right) \tag{A.15}$$

and has the property

$$\mathcal{D}_\mu\psi \to e^{-iq\lambda(\boldsymbol{x})}\mathcal{D}_\mu\psi. \tag{A.16}$$

In other words, with a specific choice of gauge field (Eq. A.12) and with the help of the redefined phase (Eq. A.10), a new derivative can be constructed (Eq. A.14) that makes

[3] $\bar{\psi} \equiv \psi^\dagger\gamma^0$, where γ^0 is the time-like gamma-matrix.

the Dirac Lagrangian invariant under local phase transformation (note the similarity between Eq. A.16 and Eq. A.4). Using these tools, a local gauge invariant version of the Dirac Lagrangian can be constructed:

$$\mathcal{L} = i\bar{\psi}\gamma^{\mu}\partial_{\mu}\psi - m\bar{\psi}\psi + \left(q\bar{\psi}\gamma^{\mu}\psi\right)A_{\mu}. \tag{A.17}$$

It's important to realise that the redefinition of the phase and the introduction of the gauge field do not just serve mathematical purposes. As with all terms in a Lagrangian, the new term seen in Eq. A.17 has a physical interpretation: it introduces both the electromagnetic field A_{μ} and the electromagnetic charge q. The free term for the gauge field, however, isn't included in the Dirac Lagrangian. Since the gauge field is a vector field, the free term can be described by the Proca Lagrangian:

$$\mathcal{L} = -\frac{1}{16\pi}F^{\mu\nu}F_{\mu\nu} + \frac{1}{8\pi}m_A^2 A^{\nu}A_{\nu}. \tag{A.18}$$

The Proca Lagrangian describes massive (m_A) spin-1 particles. The first term includes the electromagnetic tensor, which is defined as $F^{\mu\nu} = \partial^{\mu}A^{\nu} - \partial^{\nu}A^{\mu}$ and is local phase invariant under the aforementioned transformation of A_{μ} in Eq. A.12. The second term, specifically $A^{\nu}A_{\nu}$, is not local phase invariant and the only way to solve this problem is by setting the mass parameter m_A to 0, eliminating the second term entirely. This means that the gauge field A_{μ} must have a massless mediating particle — this massless gauge boson is interpreted as the photon. This results in a U(1) gauge symmetry, and is the basis of quantum electrodynamics (QED). The reason this is referred to as a U(1) gauge symmetry is that the phase in the transformation is described by a scalar[4] $e^{i\alpha(\boldsymbol{x})}$ [24].

It should be noted that in the example with the Dirac Lagrangian it looks like local gauge invariance was demanded first and then the origin of the electromagnetic force conveniently followed; however, it is the other way around: to be able to construct a Lagrangian describing the electromagnetic force, it's required that local gauge invariance is introduced.

A.3 The Higgs Boson

A.3.1 Spontaneous Symmetry-breaking

Spontaneous symmetry breaking describes the notion that a system can be in a state of equilibrium that is unstable, so a small change or perturbation in said system can lead it to fall into a lower-energy state of equilibrium. In some cases, there can be multiple (or even infinite) lower-energy states of equilibrium, all of which are equally likely — this means that the system can enter any of these states. A simple example for this behaviour can be found in ferromagnetism: when the temperature of a magnet is above the Curie temperature, the spins of the ferromagnets don't have a particular

[4] Technically, a 1 × 1 unitary matrix, hence U(1)

orientation — the magnet is paramagnetic. This means that the magnet doesn't have a polarisation; it is in a state of rotational symmetry. Once the magnet falls below the Curie temperature, the ferromagnets will align their spins, create a polarisation, and the magnet becomes magnetic. Which direction they align in is arbitrary (and equally likely); however, there is no longer a rotational symmetry and the symmetry is therefore broken [25]. The reason this is referred to as *spontaneous* is because no external agency is responsible for the breaking [24].

The following mathematical description of spontaneous symmetry-breaking and the Higgs mechanism follows [25].

The symmetries of a Lagrangian can be spontaneously broken by introducing perturbations around a ground state. Take the following potential of a scalar field:

$$V(\phi) = \frac{1}{2}\mu^2\phi^2 + \frac{1}{4}\lambda\phi^4. \tag{A.19}$$

The corresponding Lagrangian is

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}(\partial_\mu\phi)(\partial^\mu\phi) - V(\phi) && \text{(A.20)}\\ &= \frac{1}{2}(\partial_\mu\phi)(\partial^\mu\phi) - \frac{1}{2}\mu^2\phi^2 - \frac{1}{4}\lambda\phi^4, && \text{(A.21)}\end{aligned}$$

where the $(\partial_\mu\phi)(\partial^\mu\phi)$ term can be associated with the kinetic energy of the particle, the $\frac{1}{2}\mu^2\phi^2$ term with the mass and the $\lambda\phi^4$ with the self-interaction of the field. For the potential to have a finite minimum, λ must be positive, as the ϕ^4 begins to dominate for larger values. μ^2 is not restricted to being positive (where it results in an upward-facing parabolic potential) — choosing it to be negative leads to potential known as a double-well potential (Fig. A.1a), with minima at

$$\phi_{min} = \pm v = \pm\left|\sqrt{-\frac{\mu^2}{\lambda}}\right|. \tag{A.22}$$

In this case, the μ^2 term can no longer be associated with a mass. The vacuum expectation value v describes the minima of the potential; which minima the field enters can be chosen freely. This leads to spontaneous symmetry-breaking, as the choice breaks the symmetry of the Lagrangian.

To find the mass term of the Lagrangian, perturbations around the minimum must be considered. These are introduced as $\phi(x) = v + \eta(x)$. Since v is a constant, it vanishes under the derivative of the field, and the Lagrangian becomes

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}(\partial_\mu\eta)(\partial^\mu\eta) - V(\eta) && \text{(A.23)}\\ &= \frac{1}{2}(\partial_\mu\eta)(\partial^\mu\eta) - \frac{1}{2}\mu^2(v+\eta)^2 - \frac{1}{4}\lambda(v+\eta)^4. && \text{(A.24)}\end{aligned}$$

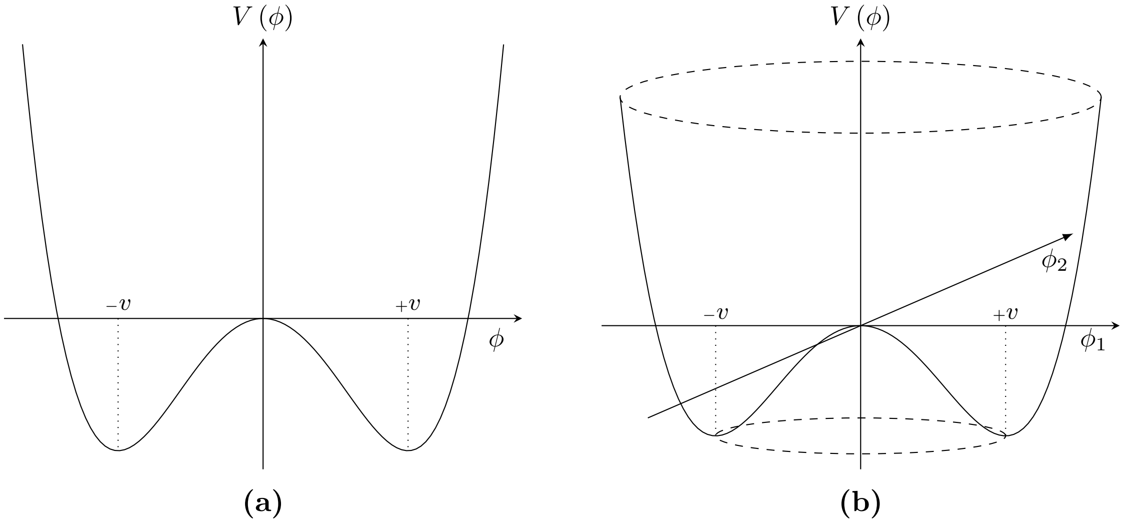


Figure A.1: **(a)** The double-well potential with minima described as the vacuum expectancy value ±v. **(b)** The "Mexican hat" potential of a complex scalar field.

By using $\mu^2 = -\lambda v^2$, the Lagrangian can be rewritten as

$$\mathcal{L} = \frac{1}{2}\left(\partial_\mu \eta\right)\left(\partial^\mu \eta\right) - \lambda v^2 \eta^2 - \lambda v \eta^3 - \frac{1}{4}\lambda \eta^4 + \frac{1}{4}\lambda v^4, \tag{A.25}$$

which reveals a previously hidden term proportional to η^2 that can be interpreted as a mass:

$$m_\eta = \sqrt{2\lambda v^2}. \tag{A.26}$$

The remaining η^3 and η^4 terms describe self-interactions. The final Lagrangian is as follows:

$$\mathcal{L}\left(\eta\right) = \frac{1}{2}\left(\partial_\mu \eta\right)\left(\partial^\mu \eta\right) - \frac{1}{2}m_\eta^2 \eta^2 - V(\eta), \text{ with } V(\eta) = \lambda v \eta^3 + \frac{1}{4}\lambda \eta^4. \tag{A.27}$$

This Lagrangian is the same as the Lagrangian in Eq. A.23, only now it's being described by perturbations around the minimum. Up until here, the field ϕ has been a real scalar field. For a complex scalar field,

$$\phi = \frac{1}{\sqrt{2}}\left(\phi_1 + i\phi_2\right) \tag{A.28}$$

with potential

$$V(\phi) = \mu^2\left(\phi^*\phi\right) + \lambda\left(\phi^*\phi\right)^2, \tag{A.29}$$

a similar procedure can be followed, only now the Lagrangian is described by two real fields,

$$\mathcal{L} = \frac{1}{2}\left(\partial_\mu \phi_1\right)\left(\partial^\mu \phi_1\right) + \frac{1}{2}\left(\partial_\mu \phi_2\right)\left(\partial^\mu \phi_2\right) - \frac{1}{2}\mu^2\left(\phi_1^2 + \phi_2^2\right) - \frac{1}{4}\lambda\left(\phi_1^2 + \phi_2^2\right)^2, \tag{A.30}$$

and the double-well potential has received an additional dimension (Fig. A.1b). The minima are zero ($\phi_1^2 + \phi_2^2 = 0$) for $\mu^2 > 0$ and for $\mu^2 < 0$ the minima are defined by

$$\phi_1^2 + \phi_2^2 = -\frac{\mu^2}{\lambda} = v^2. \tag{A.31}$$

As before, a vacuum state can be chosen — here, there are infinite possibilities, so the state $(\phi_1, \phi_2) = (v, 0)$ is selected. In this case, perturbations around the minimum of both scalar fields are considered: $\phi_1(x) = v + \eta(x)$ and $\phi_2(x) = \epsilon(x)$, so the complete field becomes $\phi = v + \eta + i\epsilon$. The potential becomes

$$V(\eta, \epsilon) = \mu^2\phi^2 + \lambda\phi^4, \text{ with } \phi^2 = \phi\phi^* = \frac{1}{2}\left[(v+\eta)^2 + \epsilon^2\right], \tag{A.32}$$

which, using $\mu^2 = \lambda v^2$, can be written-out as

$$V(\eta, \epsilon) = -\frac{1}{4}\lambda v^4 + \lambda v^2\eta^2 + \lambda v\eta^3 + \frac{1}{4}\lambda\eta^4 + \frac{1}{4}\lambda\epsilon^4 + \lambda v\eta\epsilon^2 + \frac{1}{2}\lambda\eta^2\epsilon^2. \tag{A.33}$$

Noticeably, there is only one term that is quadratic in a field: $\lambda v\eta^2$. This represents a mass of $m_\eta = \sqrt{2\lambda v^2}$, which is that same as the case of the real scalar field (see Eq. A.26), and the terms that are of higher orders in the fields η and ϵ represent interactions. Therefore, the scalar field ϵ doesn't have a mass term. The final Lagrangian is

$$\mathcal{L} = \frac{1}{2}(\partial_\mu\eta)(\partial^\mu\eta) - \frac{1}{2}m_\eta^2\eta^2 + \frac{1}{2}(\partial_\mu\epsilon)(\partial^\mu\epsilon) - V_{int}(\eta, \epsilon), \tag{A.34}$$

with

$$V_{int}(\eta, \epsilon) = \lambda v\eta^3 + \frac{1}{4}\lambda\eta^4 + \frac{1}{4}\lambda\epsilon^4 + \lambda v\eta\epsilon^2 + \frac{1}{2}\lambda\eta^2\epsilon^2, \tag{A.35}$$

where the perturbations of the field η are in the direction of a change in the potential and the perturbations of the field ϵ are in the direction of equal potential (they go "around" the valley of Fig. A.1b). As a result, while the field η describes a massive gauge boson, the field ϵ describes a massless gauge boson, known as a Goldstone boson.

A.3.2 The Higgs Mechanism

The Higgs Mechanism is built on combing spontaneous symmetry breaking with local gauge invariance. A complex scalar field ϕ is not invariant under a local U(1) gauge transformation:

$$\phi(x) \to \phi'(x) = e^{ig\chi(x)}\phi(x). \tag{A.36}$$

As with the case in Eq. A.14, a covariant derivative must be introduced

$$\partial_\mu \to \mathcal{D}_\mu \equiv \partial_\mu + igB_\mu \tag{A.37}$$

with a new gauge field B_μ, which transforms with

$$B_\mu \to {B_\mu}' \equiv B_\mu - \partial_\mu \chi(x). \tag{A.38}$$

This results in the Lagrangian

$$\mathcal{L} = -\frac{1}{4} F^{mu\nu} F_{\mu\nu} + \left(D_\mu \phi\right)^* \left(D^\mu \phi\right) - \mu^2 \phi^2 - \lambda \phi^4, \tag{A.39}$$

where the new gauge field appears in the kinematic term $F_{\mu\nu} = \partial^\mu B^\nu - \partial^\nu B^\mu$. As was the case previously, the gauge field B results in an additional term, breaking the gauge invariance, and therefore requires the gauge field to be massless. Using the covariant derivative, the Lagrangian becomes

$$\begin{aligned}\mathcal{L} = & -\frac{1}{4} F^{\mu\nu} F_{\mu\nu} + \left(\partial_\mu \phi\right)^* \left(\partial \mu \phi\right) - \mu^2 \phi^2 - \lambda \phi^4 & \text{(A.40)}\\ & - ig B_\mu \phi^* \left(\partial^\mu \phi\right) + ig \left(\partial_\mu \phi^*\right) B^\mu \phi + g^2 B_\mu B^\mu \phi^* \phi. & \text{(A.41)}\end{aligned}$$

As before, for $\mu^2 < 0$ the minimum of $\phi_1 + i\phi_2 = v$ is chosen and the potential is expanded around it: $\phi(x) = \frac{1}{\sqrt{2}} \left(v + \eta(x) + i\epsilon(x)\right)$. By substituting this into the Lagrangian that was received via the covariant derivative (Eq. A.40), the resulting Lagrangian is

$$\begin{aligned}\mathcal{L} = & \frac{1}{2} \left(\partial_\mu \eta\right) \left(\partial \mu \eta\right) - \lambda v^2 \eta^2 + \frac{1}{2} \left(\partial_\mu \epsilon\right) \left(\partial \mu \epsilon\right) & \text{(A.42)}\\ & - \frac{1}{4} F^{mu\nu} F_{\mu\nu} + \frac{1}{2} g^2 v^2 B_\mu B^\mu - V_{int} + gv B_\mu \left(\partial^\mu \epsilon\right), & \text{(A.43)}\end{aligned}$$

with $V_{int}\left(\nu, \eta, B\right)$ containing three and four-point interaction terms. Similarly to Eq. A.35, the scalar field ν has a mass term and the field ϵ describes a massless Goldstone boson. There is, however, a new term: $\frac{1}{2} g^2 v^2 B_\mu B^\mu$, which is a mass term for the gauge field B. This means that the introduction of local gauge invariance lead to a gauge boson with a mass.

An issue arises with the last term: $gvB_\mu\left(\partial^\mu \epsilon\right)$. This describes a coupling between the spin-0 scalar field ϵ and the spin-1 gauge field B. Additionally, it seems as if a new degree of freedom has appeared: before the breaking, there were four degrees of freedom — one for each scalar field ϕ_1 and ϕ_2, and two transverse polarisation states for B. After the breaking, the new gauge field has a mass and, with that, a longitudinal polarisation state which didn't exist before the breaking. This discrepancy can be resolved by applying another gauge transformation that eliminates the Goldstone boson: by recognising

$$\frac{1}{2} \left(\partial_\mu \epsilon\right) \left(\partial^\mu \epsilon\right) + gv B_\mu \left(\partial_\mu \epsilon\right) \frac{1}{2} g^2 v^2 B_\mu B^\mu = \frac{1}{2} g^2 v^2 \left(B_\mu + \frac{1}{gv} \left(\partial_\mu \epsilon\right)\right)^2 \tag{A.44}$$

and by choosing $\chi(x) = \frac{\epsilon}{gv}$ (see Eq. A.38), the field B transforms with

$$B_\mu(x) \rightarrow B_\mu{}'(x) = B_\mu(x) + \frac{1}{gv}(\partial_\mu \epsilon), \tag{A.45}$$

and the Lagrangian of Eq. A.42 becomes

$$\mathcal{L} = \frac{1}{2}(\partial_\mu \eta)(\partial^\mu \eta) - \lambda v^2 \eta^2 - \frac{1}{4}F^{\mu\nu}F_{\mu\nu} + \frac{1}{2}g^2 v^2 B_\mu{}' B^{\mu\prime} - V_{int}. \tag{A.46}$$

The choice of χ that appears in Eq. A.45 also affects the original transformation of ϕ (see Eq. A.36). This means that the complete gauge transformation of ϕ is

$$\phi(x) \rightarrow \phi'(x) = e^{ig\frac{\epsilon(x)}{gv}(x)}\phi(x) = e^{i\frac{\epsilon(x)}{v}}\phi(x). \tag{A.47}$$

The expansion of the gauge field around the minima, which was chosen to be $\phi = \frac{1}{\sqrt{2}}(v + \eta(x) + i\epsilon(x))$, can be approximated in first order to

$$\phi(x) \approx \frac{1}{\sqrt{2}}(v + \eta(x))\, e^{i\frac{\epsilon(x)}{v}}. \tag{A.48}$$

As a result, the transformed field $\phi\prime$ is no longer dependent on the field ϵ:

$$\phi(x) \rightarrow \phi'(x) = \frac{1}{\sqrt{2}}e^{-i\frac{\epsilon(x)}{v}}(v + \eta(x))\, e^{i\frac{\epsilon(x)}{v}} = \frac{1}{\sqrt{2}}(v + \eta(x)). \tag{A.49}$$

This means that the issue of the presence of the Goldstone boson can be resolved by choosing the complex scalar field ϕ to be real,

$$\phi(x) = \frac{1}{\sqrt{2}}(v + \eta(x)) = \frac{1}{\sqrt{2}}(v + h(x)), \tag{A.50}$$

and the additional degree of freedom becomes associated with longitudinal polarisation of the gauge field B.

The field $h(x)$ is known as the Higgs field — through it, the gauge boson B, that was introduced for the local gauge transformation, has gained a mass term. The final Lagrangian, described by $\mu^2 = -\lambda v^2$ and $\phi = \frac{1}{\sqrt{2}}(v + h(x))$ is as follows:

$$\mathcal{L} = \frac{1}{2}(\partial_\mu h)(\partial^\mu h) - \lambda v^2 h^2 - \frac{1}{4}F^{\mu\nu}F_{\mu\nu} + \frac{1}{2}g^2 v^2 B_\mu B^\mu \tag{A.51}$$

$$+ g^2 v B_\mu B^\mu h + \frac{1}{2}g^2 B_\mu B^\mu v^2 - \lambda v h^3 - \frac{1}{4}\lambda h^4. \tag{A.52}$$

The first two terms describe the Higgs field and its mass, the third and fourth terms the massive gauge boson, the fifth and sixth terms the interactions between the Higgs boson and the gauge boson, and the last two terms describe the self-interaction terms of the Higgs field. It should be noted that the Higgs field is a scalar field. From this, the masses of the Higgs field and the gauge boson are visible:

$$m_B = gv, \quad m_H = \sqrt{2\lambda v^2}. \tag{A.53}$$

The masses of both fields are dependent on the vacuum expectation value, where m_B is directly proportional to the strength of the gauge coupling [25].

In summary, a Lagrangian of a complex scalar field ϕ was constructed using a local gauge transformation, whereby a gauge field B was introduced. Then, the potential underwent spontaneous symmetry-breaking and perturbations around a chosen minimum of the potential were regarded. This lead to the problematic existence of a Goldstone boson; however, this could be resolved by applying a gauge transformation to the field B. By regarding the field ϕ in first order, which resulted in ϕ becoming a real scalar field, a Lagrangian could be constructed. Here, the field B and the gauge field η, which originated from the perturbations around the minimum, both gained mass terms. The gauge field η was then interpreted as the Higgs field h.

A.3.3 The Standard Model Higgs Boson

In the previous section, a Higgs boson was generated in the case of a U(1) gauge transformation. In the SM, the Higgs boson is generated via an SU(2)$_L$ × U(1)$_Y$ gauge transformation under the Glashow-Salam-Weinberg (GSW) model. To do this, a complex vector field ϕ is chosen as follows:

$$\phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ \phi_3 + i\phi_4 \end{pmatrix}, \tag{A.54}$$

where ϕ^+ and ϕ^0 are complex scalar fields. ϕ represents a weak isospin doublet and the two components differ by a charge of +1.

The minima of the Higgs potential for $\mu^2 < 0$ fulfil

$$\phi^\dagger \phi = \frac{1}{2}\left(\phi_1^2 + \phi_2^2 + \phi_3^2 + \phi_4^2\right) = \frac{v^2}{2} = -\frac{\mu^2}{2\lambda}. \tag{A.55}$$

By expanding the fields around the minimum

$$\langle 0|\, \phi \,|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v \end{pmatrix}, \tag{A.56}$$

ϕ becomes

$$\phi(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1(x) + i\phi_2(x) \\ v + \eta(x) + i\phi_4(x) \end{pmatrix}. \tag{A.57}$$

When the symmetry is broken, the result is a massive scalar boson and three massless Goldstone bosons, which will give the longitudinal degrees of freedom to the $W^\pm$ and the Z bosons [25]. The Higgs doublet then becomes

$$\phi(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + h(x) \end{pmatrix}. \tag{A.58}$$

The covariant derivatives of the $\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ gauge transformation are chosen as follows:

$$D_\mu = \partial_\mu + ig_W \boldsymbol{T} \cdot \boldsymbol{W}_\mu + ig' \frac{Y}{2} B_\mu. \tag{A.59}$$

Here, $\boldsymbol{T} = \frac{1}{2}\boldsymbol{\sigma}$ are the three generators of the SU(2) symmetry and $Y = 1^5$, therefore the covariant derivative applied to the field is

$$D_\mu \phi = \frac{1}{\sqrt{2}} \left(2\partial_\mu + \left(ig_W \boldsymbol{\sigma} \cdot \boldsymbol{W}_\mu + ig' B_\mu\right)\right) \phi. \tag{A.60}$$

Using the gauge transformation and deriving the Lagrangian with the covariant derivative will result in the fields

$$W^+ = \frac{1}{\sqrt{2}} \left(W_\mu^{(1)} - iW_\mu^{(2)}\right) \tag{A.61}$$

$$W^- = \frac{1}{\sqrt{2}} \left(W_\mu^{(1)} + iW_\mu^{(2)}\right) \tag{A.62}$$

$$Z_\mu = \frac{g' W_\mu^{(3)} + g_W B_\mu}{\sqrt{g_W^2 g'^2}} \tag{A.63}$$

$$A_\mu = \frac{g_W W_\mu^{(3)} - g' B_\mu}{\sqrt{g_W^2 g'^2}} \tag{A.64}$$

and the mass terms

$$m_W = \frac{1}{2} g_W v \tag{A.65}$$

$$m_Z = \frac{1}{2} v \sqrt{g_W^2 + g'^2} \tag{A.66}$$

$$m_A = 0. \tag{A.67}$$

By defining

$$\frac{g'}{g_W} \equiv \tan\left(\theta_W\right) \tag{A.68}$$

the following relations can be written [25]:

$$Z_\mu = \cos\left(\theta_W\right) \cdot W_\mu^{(3)} - \sin\left(\theta_W\right) \cdot B_\mu \tag{A.69}$$

$$A_\mu = \sin\left(\theta_W\right) \cdot W_\mu^{(3)} + \cos\left(\theta_W\right) \cdot B_\mu. \tag{A.70}$$

[5] $Y = 2(Q - I_3)$, where $Q = 0$ and $I_W^{(3)} = -\frac{1}{2}$

The angle θ_W is known as the weak-mixing angle, which can be determined experimentally (see [78, 3]). Using Eq. 1.11, the mass term for the Z boson can be rewritten as

$$m_Z = \frac{1}{2}\frac{g_W}{\cos\left(\theta_W\right)}v \tag{A.71}$$

from which a simple relation between the masses of the W boson and the Z boson is revealed:

$$\frac{m_W}{m_Z} = \cos\left(\theta_W\right). \tag{A.72}$$

Since $m_W = \frac{1}{2}g_W v$, measuring the mass of the W boson and its coupling parameter lead to a value for the vacuum expectation value:

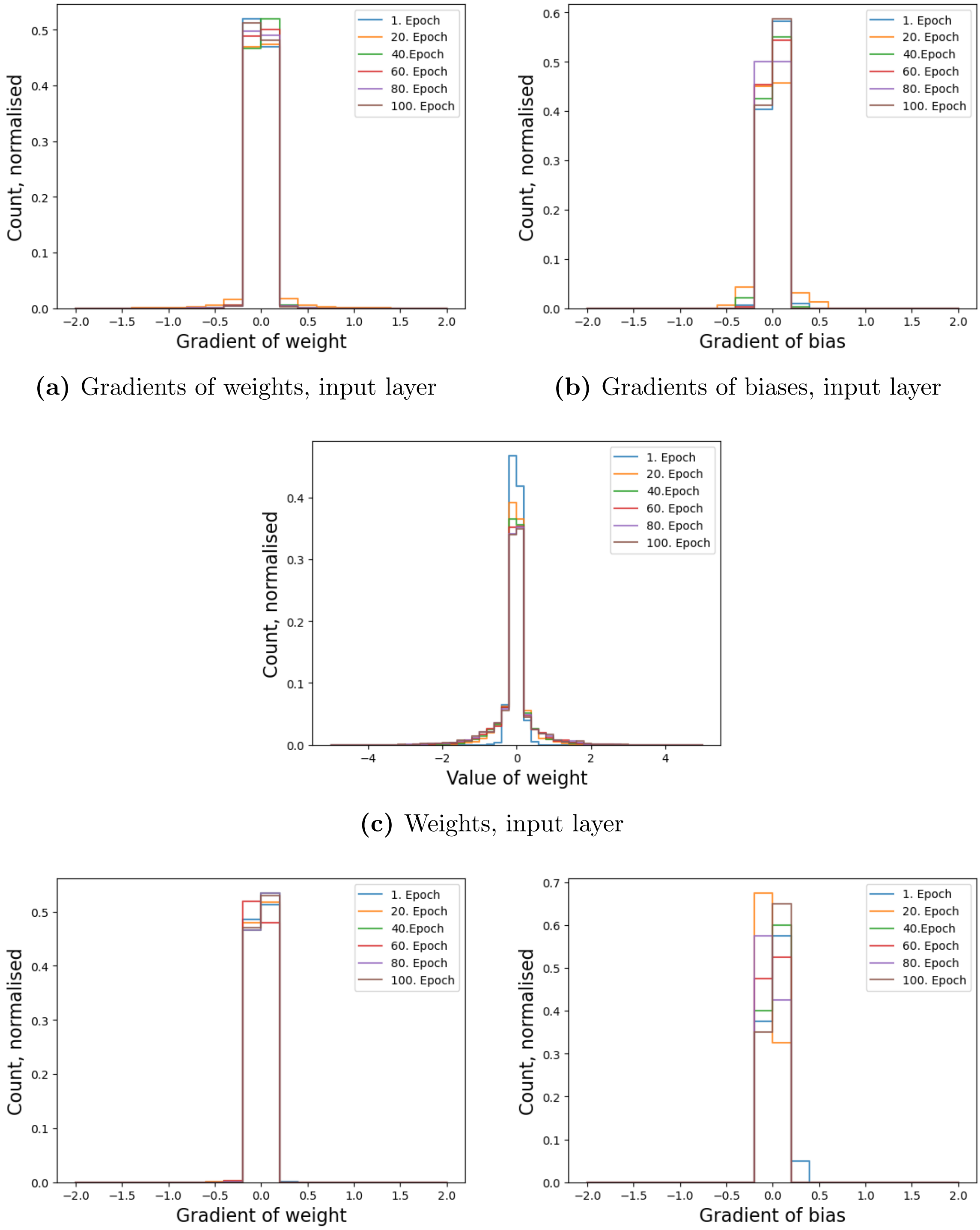
$$v = 246 \text{ GeV}. \tag{A.73}$$

Through the measurement of the Higgs boson's mass (see [1, 2]), using the vacuum expectation value and $m_H = \sqrt{2\lambda v^2}$, a value for the self-coupling parameter λ can be calculated. From this, using $v^2 = -\frac{\mu^2}{\lambda}$, a value for μ can be determined. In the end, the GSW model can be described by four parameters: g', g_W, μ and λ [25].

With this, the SM Higgs boson and its couplings to the W and Z bosons have been determined. Its couplings to fermions and their masses can be derived in a similar way (see [25]).

Appendix B

Analysis: Machine Learning



(a) Gradients of weights, input layer

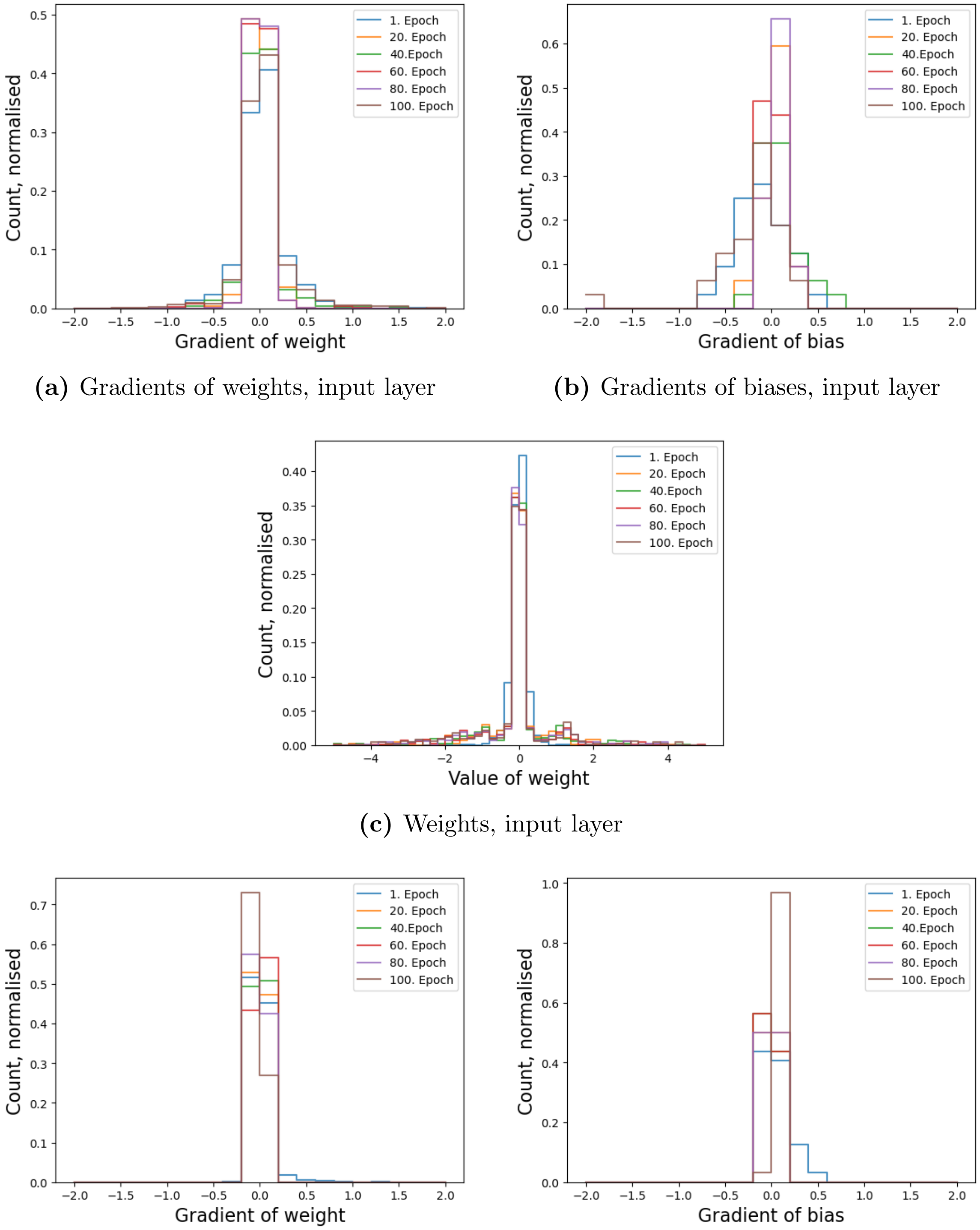
(b) Gradients of biases, input layer

(c) Weights, input layer

(d) Gradients of weights, final hidden layer

(e) Gradients of biases, final hidden layer

Figure B.1: Mixed network: Gradients of the weights and biases of the input and the last hidden layer, for a selected epochs.



(a) Gradients of weights, input layer

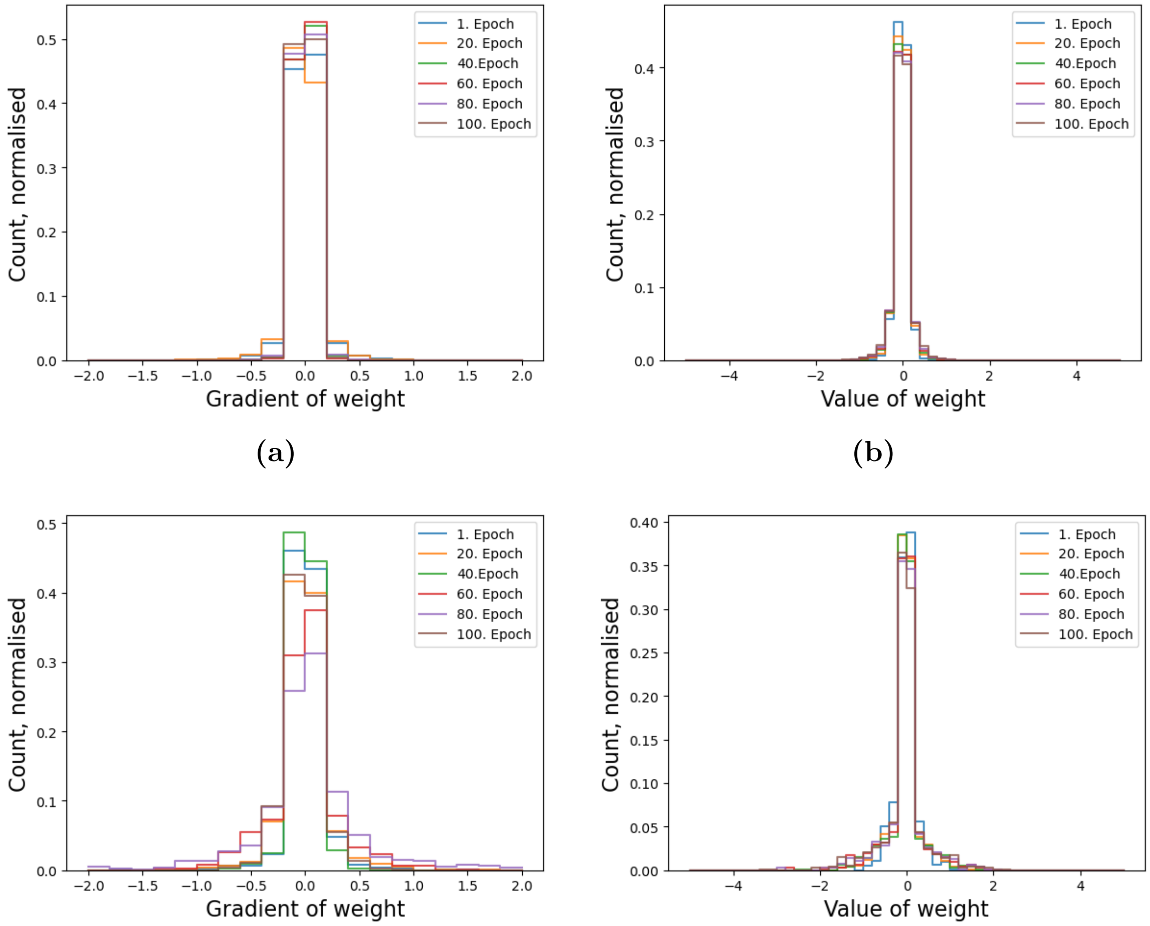
(b) Gradients of biases, input layer

(c) Weights, input layer

(d) Gradients of weights, final hidden layer

(e) Gradients of biases, final hidden layer

Figure B.2: Deep+narrow network: Gradients of the weights and biases of the input and the last hidden layer, for a selected epochs.



(a)

(b)

(c)

(d)

Figure B.3: Mixed residual network: **(a)** Gradient of the weight and **(b)** weights; Deep+narrow residual network: **(c)** Gradient of the weight and **(d)** weights;

Abbreviations

BR Branching Ratio

CL Confidence Level

CNN Convolutional Neural Network

GSW Glashow-Salam-Weinberg

ggF Gluon-Gluon Fusion

HH Higgs-Higgs-boson

HZ Higgs-Z-boson

LHC Large hadron Collider

MEM Matrix Element Method

MC MCMonte Carlo

ML Machine Learning

NLO Next-to-Leading Order

NNLO Next-to-Next-to-Leading Order

Particle ID Particle Identification Number

pp Proton-Proton

QCD Quantum Chromodynamics

SGD Stochastic Gradient Descent

SM Standard Model of Particle Physics

VBF Vector-Boson Fusion

Bibliography

[1] ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. 2012. doi: 10.1016/j.physletb.2012.08.020.

[2] CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 716(1):30–61, sep 2012. doi: 10.1016/j.physletb.2012.08.021. URL https://doi.org/10.1016%2Fj.physletb.2012.08.021.

[3] R. L. Workman and Others. Review of Particle Physics. *PTEP*, 2022:083C01, 2022. doi: 10.1093/ptep/ptac097.

[4] Giuseppe Degrassi, Stefano Di Vita, Joan Elias-Miró, José R. Espinosa, Gian F. Giudice, Gino Isidori, and Alessandro Strumia. Higgs mass and vacuum stability in the Standard Model at NNLO. *Journal of High Energy Physics*, 2012 (8), aug 2012. doi: 10.1007/jhep08(2012)098. URL https://doi.org/10.1007%2Fjhep08%282012%29098.

[5] Vincenzo Branchina and Emanuele Messina. Stability, Higgs Boson Mass, and New Physics. *Phys. Rev. Lett.*, 111:241801, Dec 2013. doi: 10.1103/PhysRevLett.111.241801. URL https://link.aps.org/doi/10.1103/PhysRevLett.111.241801.

[6] Jérôme Martin. Everything you always wanted to know about the cosmological constant problem (but were afraid to ask). *Comptes Rendus Physique*, 13(6-7):566–665, jul 2012. doi: 10.1016/j.crhy.2012.04.008. URL https://doi.org/10.1016%2Fj.crhy.2012.04.008.

[7] David E Morrissey and Michael J Ramsey-Musolf. Electroweak baryogenesis. *New Journal of Physics*, 14(12):125003, dec 2012. doi: 10.1088/1367-2630/14/12/125003. URL https://doi.org/10.1088%2F1367-2630%2F14%2F12%2F125003.

[8] James D. Wells. Higgs naturalness and the scalar boson proliferation instability problem. *Synthese*, 194(2):477–490, dec 2014. doi: 10.1007/s11229-014-0618-8. URL https://doi.org/10.1007%2Fs11229-014-0618-8.

[9] D0 Collaboration. A precision measurement of the mass of the top quark. *Nature*, 429(6992):638–642, jun 2004. doi: 10.1038/nature02589. URL https://doi.org/10.1038%2Fnature02589.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, may 2015. doi: 10.1038/nature14539.

[11] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012. doi: 10.1109/CVPR.2012.6248110.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*, 60(6):84–90, may 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://doi.org/10.1145/3065386.

[13] Juergen Schmidhuber. Deep Learning in Neural Networks: An Overview. 2014. doi: 10.48550/ARXIV.1404.7828.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[15] Eric Mjolsness and Dennis DeCoste. Machine Learning for Science: State of the Art and Future Prospects. *Science*, 293(5537):2051–2055, 2001. doi: doi:10.1126/science.293.5537.2051. URL https://www.science.org/doi/abs/10.1126/science.293.5537.2051.

[16] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91:045002, Dec 2019. doi: 10.1103/RevModPhys.91.045002. URL https://link.aps.org/doi/10.1103/RevModPhys.91.045002.

[17] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, Alexander Himmel, Adam Aurisano, Kazuhiro Terao, and Taritree Wongjirad. Machine learning at the energy and intensity frontiers of particle physics. *Nature*, 560(7716):41–48, 2018. doi: 10.1038/s41586-018-0361-2. URL https://doi.org/10.1038/s41586-018-0361-2.

[18] Matthew D. Schwartz. Modern Machine Learning and Particle Physics. *Harvard Data Science Review*, mar 2021. doi: 10.1162/99608f92.beeb1183. URL https://doi.org/10.1162%2F99608f92.beeb1183.

[19] ATLAS Collaboration. Operation of the ATLAS trigger system in Run 2. *Journal of Instrumentation*, 15(10):P10004, oct 2020. doi: 10.1088/1748-0221/15/10/P10004. URL https://dx.doi.org/10.1088/1748-0221/15/10/P10004.

[20] Matthew Feickert and Benjamin Nachman. A Living Review of Machine Learning for Particle Physics, 2021.

[21] Christoph Ames. Modelling Pair Production of Top Squarks with Decays via Tau Sleptons in the pMSSM. Master's thesis, Ludwig-Maximilians-Universität München, 2020. URL https://www.etp.physik.uni-muenchen.de/publications/theses/index.html.

[22] Claudio Campagnari and Melissa Franklin. The discovery of the top quark. *Rev. Mod. Phys.*, 69:137–212, Jan 1997. doi: 10.1103/RevModPhys.69.137. URL https://link.aps.org/doi/10.1103/RevModPhys.69.137.

[23] Peter W. Higgs. Broken Symmetries and the Masses of Gauge Bosons. *Phys. Rev. Lett.*, 13:508–509, Oct 1964. doi: 10.1103/PhysRevLett.13.508. URL https://link.aps.org/doi/10.1103/PhysRevLett.13.508.

[24] David J Griffiths. *Introduction to elementary particles; 2nd rev. version*. Physics textbook. Wiley, New York, NY, 2008. URL https://cds.cern.ch/record/111880.

[25] Mark Thomson. *Modern particle physics*. Cambridge University Press, New York, 2013. ISBN 978-1-107-03426-6. doi: 10.1017/CBO9781139525367.

[26] Nicola Cabibbo. Unitary Symmetry and Leptonic Decays. *Phys. Rev. Lett.*, 10:531–533, Jun 1963. doi: 10.1103/PhysRevLett.10.531. URL https://link.aps.org/doi/10.1103/PhysRevLett.10.531.

[27] Makoto Kobayashi and Toshihide Maskawa. CP Violation in the Renormalizable Theory of Weak Interaction. *Prog. Theor. Phys.*, 49:652–657, 1973. doi: 10.1143/PTP.49.652.

[28] Anthony Zee. *Quantum Field Theory in a Nutshell: Second Edition*. Princeton University Press, 2 2010. ISBN 978-0-691-14034-6.

[29] Anthony Zee. *Einstein Gravity in a Nutshell*. Princeton University Press, New Jersey, 5 2013. ISBN 978-0-691-14558-7.

[30] LIGO Scientific Collaboration, Virgo Collaboration. Observation of Gravitational Waves from a Binary Black Hole Merger. *Phys. Rev. Lett.*, 116:061102, Feb 2016. doi: 10.1103/PhysRevLett.116.061102. URL https://link.aps.org/doi/10.1103/PhysRevLett.116.061102.

[31] Pankaj Agrawal, Debashis Saha, Ling-Xiao Xu, Jiang-Hao Yu, and C.-P. Yuan. Determining the shape of the Higgs potential at future colliders. *Physical Review D*, 101(7), April 2020. ISSN 2470-0029. doi: 10.1103/physrevd.101.075023. URL http://dx.doi.org/10.1103/PhysRevD.101.075023.

[32] Dario Buttazzo, Giuseppe Degrassi, Pier Paolo Giardino, Gian F. Giudice, Filippo Sala, Alberto Salvio, and Alessandro Strumia. Investigating the near-criticality of the Higgs boson. *Journal of High Energy Physics*, 2013(12), December 2013. ISSN 1029-8479. doi: 10.1007/jhep12(2013)089. URL http://dx.doi.org/10.1007/JHEP12(2013)089.

[33] Andrew Noble and Maxim Perelstein. Higgs self-coupling as a probe of the electroweak phase transition. *Physical Review D*, 78(6), September 2008. ISSN 1550-2368. doi: 10.1103/physrevd.78.063518. URL http://dx.doi.org/10.1103/PhysRevD.78.063518.

[34] Oliver Gould, Sinan Güyer, and Kari Rummukainen. First-order electroweak phase transitions: A nonperturbative update. *Physical Review D*, 106(11), December 2022. ISSN 2470-0029. doi: 10.1103/physrevd.106.114507. URL http://dx.doi.org/10.1103/PhysRevD.106.114507.

[35] Svend Erik Rugh and Henrik Zinkernagel. The Quantum Vacuum and the Cosmological Constant Problem, 2000.

[36] Fred Jegerlehner. The Hierarchy Problem and the Cosmological Constant Problem Revisited: Higgs Inflation and a New View on the SM of Particle Physics. *Foundations of Physics*, 49(9):915–971, May 2019. ISSN 1572-9516. doi: 10.1007/s10701-019-00262-2. URL http://dx.doi.org/10.1007/s10701-019-00262-2.

[37] ATLAS Collaboration. Constraints on the Higgs boson self-coupling from single- and double-Higgs production with the ATLAS detector using pp collisions at s = 13 TeV. *Physics Letters B*, 843:137745, 2023. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2023.137745. URL https://www.sciencedirect.com/science/article/pii/S0370269323000795.

[38] CMS Collaboration. Search for nonresonant Higgs boson pair production in final states with two bottom quarks and two photons in proton-proton collisions at $\sqrt{s}$ = 13 TeV. *Journal of High Energy Physics*, 2021(3), March 2021. ISSN 1029-8479. doi: 10.1007/jhep03(2021)257. URL http://dx.doi.org/10.1007/JHEP03(2021)257.

[39] Pierre Artoisenet, Vincent Lemaître, Fabio Maltoni, and Olivier Mattelaer. Automation of the matrix element reweighting method. *Journal of High Energy Physics*, 2010(12), December 2010. ISSN 1029-8479. doi: 10.1007/jhep12(2010)068. URL http://dx.doi.org/10.1007/JHEP12(2010)068.

[40] Frank Fiedler, Alexander Grohsjean, Petra Haefner, and Philipp Schieferdecker. The matrix element method and its application to measurements of the top quark mass. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 624(1):203–218, dec 2010. doi: 10.1016/j.nima.2010.09.024. URL https://doi.org/10.1016%2Fj.nima.2010.09.024.

[41] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *Physics Reports*, 810:1–124, May 2019. ISSN 0370-1573. doi: 10.1016/j.physrep.2019.03.001. URL http://dx.doi.org/10.1016/j.physrep.2019.03.001.

[42] Daniel J. Amit. *Modeling Brain Function: The World of Attractor Neural Networks*. Cambridge University Press, 1989.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[46] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.10.013. URL https://www.sciencedirect.com/science/article/pii/S0031320317304120.

[47] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, 7:e623, July 2021. ISSN 2376-5992. doi: 10.7717/peerj-cs.623. URL https://doi.org/10.7717/peerj-cs.623.

[48] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *Journal of High Energy Physics*, 2014(7), July 2014. ISSN 1029-8479. doi: 10.1007/jhep07(2014)079. URL http://dx.doi.org/10.1007/JHEP07(2014)079.

[49] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H.-S. Shao, and M. Zaro. The automation of next-to-leading order electroweak calculations. *Journal of High Energy Physics*, 2018(7), July 2018. ISSN 1029-8479. doi: 10.1007/jhep07(2018)185. URL http://dx.doi.org/10.1007/JHEP07(2018)185.

[50] Andy Buckley, James Ferrando, Stephen Lloyd, Karl Nordström, Ben Page, Martin Rüfenacht, Marek Schönherr, and Graeme Watt. LHAPDF6: parton density access in the LHC precision era. *The European Physical Journal C*, 75 (3), March 2015. ISSN 1434-6052. doi: 10.1140/epjc/s10052-015-3318-8. URL http://dx.doi.org/10.1140/epjc/s10052-015-3318-8.

[51] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet user manual: (for version 3.0.2). *The European Physical Journal C*, 72(3), March 2012. ISSN 1434-6052. doi: 10.1140/epjc/s10052-012-1896-2. URL http://dx.doi.org/10.1140/epjc/s10052-012-1896-2.

[52] Matteo Cacciari and Gavin P. Salam. Dispelling the N^3 myth for the k$_t$ jet-finder. *Physics Letters B*, 641(1):57–61, September 2006. ISSN 0370-2693. doi: 10.1016/j.physletb.2006.08.037. URL http://dx.doi.org/10.1016/j.physletb.2006.08.037.

[53] R. Brun and F. Rademakers. ROOT: An object oriented data analysis framework. *Nucl. Instrum. Meth. A*, 389:81–86, 1997. doi: 10.1016/S0168-9002(97)00048-X.

[54] Jim Pivarski, Henry Schreiner, Angus Hollands, Pratyush Das, Kush Kothari, Aryan Roy, Jerry Ling, Nicholas Smith, Chris Burr, and Giordon Stark. Uproot, February 2024. URL https://doi.org/10.5281/zenodo.10699405.

[55] Michael L. Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

[56] Sébastien Brochet, Christophe Delaere, Brieuc François, Vincent Lemaître, Alexandre Mertens, Alessia Saggio, Miguel Vidal Marono, and Sébastien Wertz. MoMEMta, a modular toolkit for the Matrix Element Method at the LHC. *The European Physical Journal C*, 79(2):126, 2019. doi: 10.1140/epjc/s10052-019-6635-5. URL https://doi.org/10.1140/epjc/s10052-019-6635-5.

[57] Olivier Mattelaer and Pierre Artoisenet. MadWeight: automatic event reweighting with matrix elements. *PoS*, CHARGED2008:025, 2009. doi: 10.22323/1.073.0025.

[58] T. Hahn. Cuba — a library for multidimensional numerical integration. *Computer Physics Communications*, 168(2):78–95, June 2005. ISSN 0010-4655. doi: 10.1016/j.cpc.2005.01.010. URL http://dx.doi.org/10.1016/j.cpc.2005.01.010.

[59] G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192–203, 1978. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(78)90004-9. URL https://www.sciencedirect.com/science/article/pii/0021999178900049.

[60] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. CUDA, release: 10.2.89, 2020. URL https://developer.nvidia.com/cuda-toolkit.

[61] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[62] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[63] ATLAS Collaboration. ATLAS b-jet identification performance and efficiency measurement with $t\bar{t}$ events in pp collisions at $\sqrt{s} = 13$ TeV. *The European Physical Journal C*, 79(11), November 2019. ISSN 1434-6052. doi: 10.1140/epjc/s10052-019-7450-8. URL http://dx.doi.org/10.1140/epjc/s10052-019-7450-8.

[64] ATLAS Collaboration. Performance of electron and photon triggers in ATLAS during LHC Run 2. *The European Physical Journal C*, 80(1), January 2020. ISSN 1434-6052. doi: 10.1140/epjc/s10052-019-7500-2. URL http://dx.doi.org/10.1140/epjc/s10052-019-7500-2.

[65] ATLAS Collaboration. Muon reconstruction and identification efficiency in ATLAS using the full Run 2 pp collision data set at $\sqrt{s} = 13$ TeV. *The European Physical Journal C*, 81(7), July 2021. ISSN 1434-6052. doi: 10.1140/epjc/s10052-021-09233-2. URL http://dx.doi.org/10.1140/epjc/s10052-021-09233-2.

[66] Lukas von Stumpfeldt. Separating Higgs-Selfinteraction Processes from Higgs-Pairproduction using Neural Networks. Master's thesis, Ludwig-Maximilian-University Munich, 2024.

[67] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. PYTHIA 6.4 Physics and Manual. *JHEP*, 05:026, 2006. doi: 10.1088/1126-6708/2006/05/026.

[68] Stephen D. Ellis and Davison E. Soper. Successive combination jet algorithm for hadron collisions. *Physical Review D*, 48(7):3160–3166, October 1993. ISSN 0556-2821. doi: 10.1103/physrevd.48.3160. URL http://dx.doi.org/10.1103/PhysRevD.48.3160.

[69] S. Catani, Yu.L. Dokshitzer, M.H. Seymour, and B.R. Webber. Longitudinally-invariant k”$_{a5}$-clustering algorithms for hadron-hadron collisions. *Nuclear Physics B*, 406(1):187–224, 1993. ISSN 0550-3213. doi: https://doi.org/10.1016/0550-3213(93)90166-M. URL https://www.sciencedirect.com/science/article/pii/055032139390166M.

[70] Stephen D. Ellis, Christopher K. Vermilion, and Jonathan R. Walsh. Recombination algorithms and jet substructure: Pruning as a tool for heavy particle searches. *Physical Review D*, 81(9), May 2010. ISSN 1550-2368. doi: 10.1103/physrevd.81.094023. URL http://dx.doi.org/10.1103/PhysRevD.81.094023.

[71] Hung-Liang Lai, Marco Guzzi, Joey Huston, Zhao Li, Pavel M. Nadolsky, Jon Pumplin, and C. P. Yuan. New parton distributions for collider physics. *Phys. Rev. D*, 82:074024, 2010. doi: 10.1103/PhysRevD.82.074024.

[72] Celine Stauch. Separation of HH and HZ Final States Using Spin Correlations. Master's thesis, Ludwig-Maximilian-University Munich, 2023.

[73] Nitish Shirish Keskar and Richard Socher. Improving Generalization Performance by Switching from Adam to SGD, 2017. URL https://arxiv.org/abs/1712.07628.

[74] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015.

[76] Herbert Goldstein, Charles Poole, and John Safko. *Classical Mechanics (3rd Edition)*. 06 2001. ISBN 0201657023.

[77] Michael E. Peskin and Daniel V. Schroeder. *An Introduction to quantum field theory*. Addison-Wesley, Reading, USA, 1995. ISBN 978-0-201-50397-5.

[78] CMS Collaboration. Measurement of the weak mixing angle with the Drell-Yan process in proton-proton collisions at the LHC. *Physical Review D*, 84 (11), December 2011. ISSN 1550-2368. doi: 10.1103/physrevd.84.112002. URL http://dx.doi.org/10.1103/PhysRevD.84.112002.

Acknowledgements

Working on an analysis and writing a thesis for more than $3^{1}/_{2}$ years requires the effort of more than just one person — I would like to thank everyone that supported me along the way.

First and foremost, a special thanks goes to Professor Otmar Biebel for giving me this opportunity. When I started, the analysis team had to be rebuilt form scratch due to a necessary topic change and I was the first to join. It was very exciting to be taking the first steps for our group in the topics of Higgs physics, the matrix element method and machine learning, all at once. I've really enjoyed working with Otmar due to his friendliness, and thanks to his help and guidance, I've learnt a lot and I'm proud of what I have been able to achieve.

Thank you to Celine, Lars and Nikolai for taking the time to proof-read my thesis and giving very helpful feedback. With them, I'd also like to thank everyone else in our group and the group from Thomas Kuhr, especially Alex, David, Eshita, Eylül, Fabian, Katrin, Nick, Stefanie and Yaroslav, for making the office a fun and enjoyable place to be.

Thank you to Anna S., Benedikt and Michi, with whom I might not have even managed to get to the point where I could start working on a Ph.D. thesis. Studying and hanging-out together has been an absolute joy and I look forward to continuing to do so. In that same vein, thank you to Anna B. for struggling through the bachelors and masters theses with me.

I'd really like to thank my gymnastics group, who I hold very dearly, and my co-trainers, especially Nicole and Sabine, who is also one of my best friends. Without gymnastics balancing everything out, I might have gone mad.

And lastly, I want to thank my family: Mama, Papa, Elliot and Stephanie. I would not have made it though my studies and working on this thesis without them. They have always been there for me, having my back with every decision I have made and (gently) pushing me to continue onwards.