Testing a New Method of Constraining Standard Model Effective Field Theory with Physics-Based Machine Learning

by

Kye Emond

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Mathematical Physics Honours

in the
Department of Physics
Faculty of Science

SIMON FRASER UNIVERSITY
Spring 2024

Abstract

Standard Model Effective Field Theory (SMEFT) is a formalism that introduces parameters known as Wilson Coefficients to model new physics. We evaluate a new method to constrain the possible values of two such Wilson Coefficients that affect the production and decay of Higgs bosons in the ATLAS experiment at the Large Hadron Collider. Previous work determined constraints for these coefficients using likelihoods constructed by binning events and comparing the binned counts to theory. Our method promises to produce stronger constraints by constructing a likelihood which processes each event individually, rather than the binned totals. An important part of this construction is analytically intractable, so we train deep neural networks to approximate it. Although we find evidence that this method could indeed produce stronger constraints on any proposed theories, we also find that standard feedforward neural networks fail to train to the required accuracy with a variety of architectures.

Keywords: effective field theory; machine learning; particle physics; vector boson fusion; HWW; cHj3; Higgs

Acknowledgements

There are a lot of people that helped out with this thesis. Of course, I would like to thank Bernd Stelzer for suggesting this project, supervising my thesis, and continually providing suggestions and advice. I would also like to thank Matthew Basso for generating all the MadGraph data used to train my networks, for providing tons of advice, for explaining the project to me at the start, and for being there to discuss ideas with. Nikita Dolganov was also very helpful, jumping in near the end to quickly try a new procedure for reweighting data that we hoped might help with training.

Everyone in the SFU ATLAS group who helped with debugging, accessing Cedar, and many other things deserve acknowledgement as well. So thank you to Jiayi, Metea, Tiago, Jackson, Callum, Damian, and Oliver.

Thank you to Ben Adcock, who took the time to meet with me and give me advice on training neural networks to approximate functions.

Thank you very much to Jonathan Jedwab and Grayson Davis, who both gave a lot of feedback on the writing and math in my thesis. Without you both, this document would probably be unreadable, incomprehensible to anyone not in particle physics, and maybe not finished either.

Thank you to Katie and Isaac who both eagerly volunteered to proofread a 30-page math-filled thesis out of the kindness of their hearts.

Finally, thank you to the people in my life – Layla, Michelle, Mom, and Dad – who all commiserated with me after listening to my endless stream of complaints and confusion despite having no clue what on Earth I was talking about.

Table of Contents

Abstract ii

Acknowledgements iii

Table of Contents iv

List of Figures v

List of Tables vi

1 Introduction 1

2 Theory 4

2.1 Particle Physics Primer . . . 4
2.2 Statistical Inference . . . 8
2.3 Particle Physics Likelihood Ratio . . . 9
2.4 Standard Model Effective Field Theory . . . 10
2.5 The Quadratic Classifier . . . 12
2.6 Neural Networks . . . 13

3 Implementation 18

3.1 Data Generation . . . 18
3.2 Neural Network Design . . . 21
3.3 Constraining Coefficients with Likelihood Ratios . . . 22

4 Results 24

4.1 Proof of Concept . . . 24
4.2 Full Test . . . 27
4.3 Hyperparameter Optimization . . . 29

5 Conclusion 33

Bibliography 35

List of Figures

Figure 2.1 The Standard Model 5
Figure 2.2 The ATLAS detector 7
Figure 2.3 Vector Boson Fusion 7
Figure 2.4 Higgs to WW Decay 8
Figure 2.5 Single layer of a neural network 14
Figure 2.6 Full neural network 15
Figure 2.7 Neural network function approximation 16
Figure 2.8 Gradient descent 17

Figure 3.1 Construction of a confidence interval 23

Figure 4.1 Proof of concept c_{HW} confidence interval 25
Figure 4.2 Proof of concept c_{Hj3} confidence interval 26
Figure 4.3 Best full c_{HW} confidence intervals 27
Figure 4.4 Best full c_{Hj3} confidence intervals 28
Figure 4.5 Neural network improvements with size 30

List of Tables

Table 3.1 Kinematic variables in our data 20

Chapter 1

Introduction

The Standard Model of particle physics is the incredible culmination of over 50 years of experimental and theoretical science. It describes every particle we have observed along with their interactions through the strong, weak, and electromagnetic forces. Despite its amazing success however, we know that the Standard Model cannot be the full story. For one, the observed oscillation of neutrino flavour means that neutrinos must have mass, whereas the classic Standard Model predicts massless neutrinos [29]. The currently accepted model of cosmology requires the existence of a new substance known as dark matter, which is also unaccounted for in the Standard Model [6]. The theory does not explain gravity and attempts to incorporate gravity into the Model have so far been unsuccessful [15]. These examples clearly indicate that we must search for physics beyond the Standard Model in order to properly describe the universe.

There are infinitely many possible extensions to the Standard Model, any of which could be the true description of existence. In turn, this means testing each theory one by one is an infeasibly large task. This is why approaches to discovering new physics that apply to many models at once are important. This thesis focuses on using machine learning to constrain the parameters of one such approach, known as the Standard Model Effective Field Theory (SMEFT).

An Effective Field Theory (EFT) is a formalism that rigorously describes physical processes at energies much less than a chosen energy scale Λ, without assuming anything about the structure of the theory at higher energies [37]. For example, when considering the behaviour of hydrogen atoms, we do not need to consider the quantum chromodynamics that keep the proton together. We instead treat the proton as an indivisible particle and perform our calculations using only nonrelativistic quantum mechanics.

Applying EFT to the Standard Model results in the Standard Model Effective Field Theory [14], in which new physical effects are parametrized by real-valued, dimensionless parameters known as Wilson Coefficients. Any extension to the Standard Model can be reduced to SMEFT at low enough energies, with the Wilson Coefficients determined by the structure of the theory itself. Though in principle it would take infinitely many Wilson

Coefficients to fully specify SMEFT, in practice its effects are suppressed by powers of our energy scale Λ. To make accurate predictions, it is sufficient to account for just the least suppressed Coefficients, of which there only are finitely many [26]. Experimentally constraining these Wilson Coefficients is thus an effective and feasible way to determine which models could theoretically describe our universe.

This thesis tests a new method to constrain two specific Wilson Coefficients c_{HW} and c_{Hj3}, which modify the observables of Vector Boson Fusion (VBF) Higgs boson creation and its decay to two W bosons. The typical method for discriminating between hypotheses θ_0 and θ_1 given a set of data x is to compare the likelihood ratio

$$L(\theta_0, \theta_1, x) = \frac{\mathcal{L}(\theta_1|x)}{\mathcal{L}(\theta_0|x)} = \frac{p(x|\theta_1)}{p(x|\theta_0)} \tag{1.1}$$

to some cutoff value, which by the Neyman-Pearson lemma [34], is guaranteed to give the smallest chance of incorrectly accepting the null hypothesis θ_0 for a specified probability of incorrectly rejecting the null. In the above equation, $p(x|\theta)$ is the probability density of x given θ, which is equal to the likelihood $\mathcal{L}(\theta|x)$ of θ given x. We can use this likelihood ratio to compare hypotheses where the Wilson Coefficients take different values, allowing us to determine minimal confidence intervals for those Wilson Coefficients. Unfortunately, determining this likelihood ratio requires integrating over every possible history that could have led to the data x, which is completely infeasible [10].

Past work [5], [7] has dealt with this issue by focusing on a few important kinematic observables such as transverse momenta or invariant masses, then binning these observables and treating the count of each bin as independent Poisson processes. While these analyses can lead to good constraints in specific directions of parameter space, there are often directions in the parameter space with limited sensitivity [9], [11], and binning variables results in a loss of information about their mutual correlations.

Another method for bypassing the calculation issue is to use machine learning to train an estimator $\hat{L}(\theta_0, \theta_1, x)$ to approximate the likelihood ratio $L(\theta_0, \theta_1, x)$. Given enough training data, such an estimator can learn to identify details that are missed by binning a subset of observables, leading to strong constraints in all directions. Another benefit of this method is that after the initial training period, evaluation of the likelihood estimator takes seconds, even on existing laptop computers.

This machine learning-based method can be further improved by incorporating knowledge of the underlying physics into the estimation. In most cases, the EFT prediction for the differential cross-section of any given event is a quadratic function of the Wilson Coefficients [21]. We can take this into account when designing the likelihood ratio estimator to create a "Quadratic Classifier" that displays improved accuracy of estimation and interpolation between training points in some scenarios [17]. We employ the ideas of this

Quadratic Classifier to test whether it might produce better constraints on c_{HW} and c_{Hj3} than previous research.

The upcoming chapters explain the procedure used to test these constraints and show the results of the tests. More precisely, Theory explains more about SMEFT and gives the background required to understand the estimator we constructed along with the method we used to produce constraints. Implementation discusses the specific design of the machine learning algorithm used to build the estimator, the procedure used to construct constraints from the estimator, as well as the method used for generating training data. Results demonstrates that the method is effective at producing constraints with simpler datasets, but that it struggles to learn to the required accuracy on larger sets of data. It also elaborates on how various choices affect the accuracy of our model. Finally, the Conclusion discusses the impact of this thesis and paths for future work.

Chapter 2

Theory

2.1 Particle Physics Primer

The Standard Model of Particle Physics is a type of theory known as a quantum field theory. These theories generalize quantum mechanics to relativistic energies and speeds. In any such quantum field theory, the fundamental constituents of the universe are quantum fields, and their evolution over time is governed by a function known as a Lagrangian.

Excitations in the fields governed by Lagrangians are called particles. Each particle has a corresponding anti-particle which inverts all its quantum numbers, such as electric charge and colour charge. These particles and anti-particles are split into two groups — the fermions with intrinsic angular momentum of odd multiples of $\hbar/2$, and the bosons with even multiples. Fermions make up what we typically consider matter, while bosons like photons are associated with forces like electromagnetism.

Fermions are further split into baryons (or quarks) and leptons. The number of baryons minus the number of anti-baryons is known as baryon number. Similarly, the number of leptons minus the number of anti-leptons is known as lepton number. In all experiments to date, we have never observed a change in either baryon number or lepton number. All of these particles and their categories can be seen in Figure 2.1.

In order to experimentally test the Standard Model, we need to observe interactions between particles and check whether the results of these interactions agree with our theory. However, many interactions occur only when the particles involved have high enough energies, and many others occur only very rarely. To mitigate these issues, we have constructed experimental devices known as particle accelerators. In many such accelerators, beams of particles are sped up, almost to the speed of light, and then intersected with the goal of causing high-energy collisions between the particles in the beams at a high rate.

Two very important quantities for characterizing the interactions in these particle accelerators are luminosity L and cross-section σ.

The total (or integrated) luminosity measures the number of opportunities particles have had to interact for each unit of area of the particle beam. We can think of it as the total

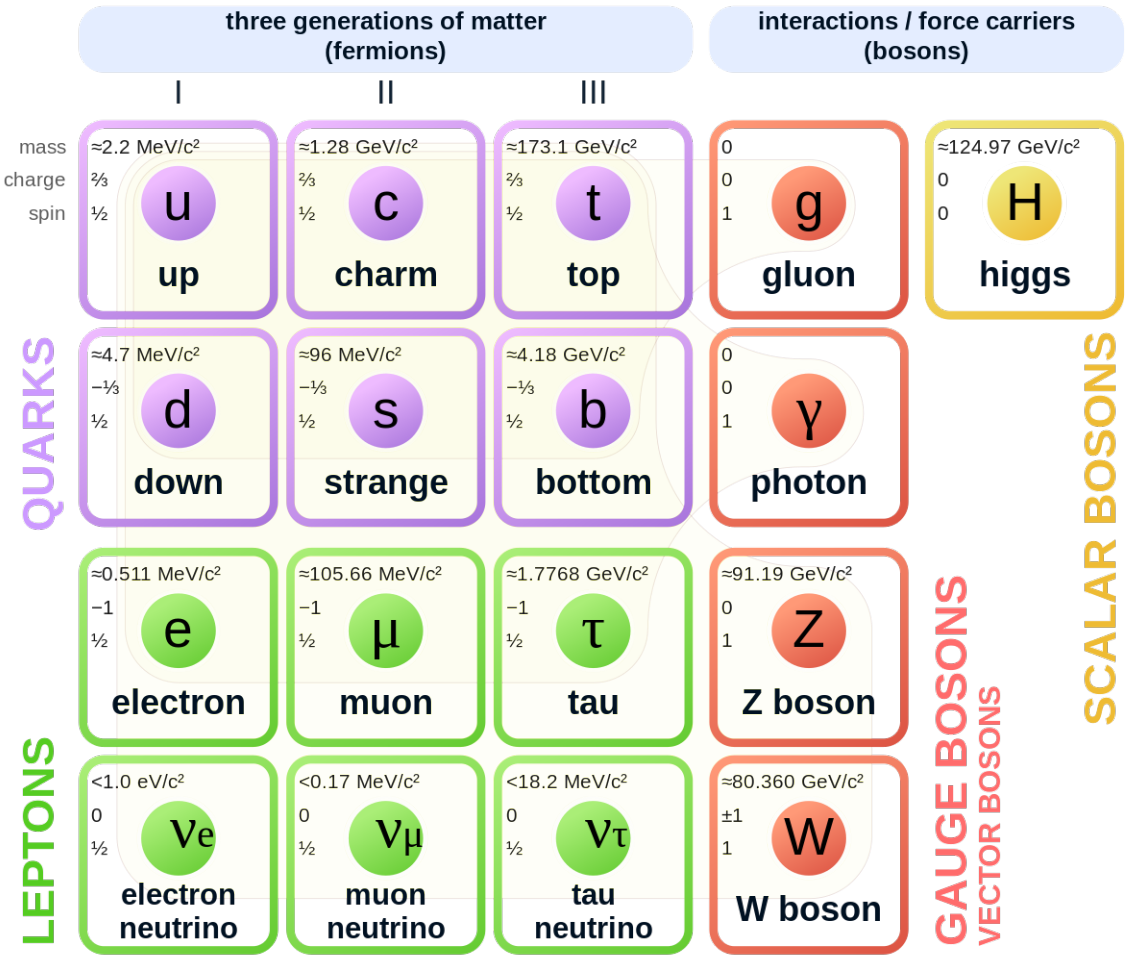

Figure 2.1: The particles of the Standard Model [22].

number of particles that have been shot through the beam divided by the cross-sectional area of that beam. A related quantity is the instantaneous luminosity, the time derivative of the integrated luminosity. This quantity measures how effective an accelerator is at providing opportunities for particle interactions, as it equals the number of opportunities for particles to interact per unit area and per unit time.

The other quantity of interest is the cross-section of events. This quantity is a measure of the probability of any individual event occurring. Namely, the total expected number of events N is given by

$$N = \sigma L. \tag{2.1}$$

The differential cross-section $\frac{\mathrm{d}\sigma}{\mathrm{d}x}(x)$ is equally important, if not more so. Similarly to how the total cross-section σ is a measure of probability, the differential cross-section $\frac{\mathrm{d}\sigma}{\mathrm{d}x}(x)$ is a measure of probability density with respect to some observed quantity x. For example, $\frac{\mathrm{d}\sigma}{\mathrm{d}\theta}(\theta)$ measures the probability density of seeing a particle scatter off of another at an angle θ. Specifically, given that we know an interaction has occurred, the probability density p for the observable x is given by

$$p(x) = \frac{1}{\sigma}\frac{\mathrm{d}\sigma}{\mathrm{d}x}(x). \tag{2.2}$$

When particles are created and fly away from the collision site, we often want to measure their trajectories. Since the setup in particle accelerators is usually rotationally symmetric about the beam axis, using some form of cylindrical or spherical coordinates is useful. Indeed, the angle around the beam line is described with a regular angle. However, due to the massive speeds of the particles along the beam, relativistic effects can make it difficult to work with the angle from the beam axis. Instead, particle physicists often use a value known as rapidity [38], which encodes the angle from the beam line while having desirable properties such as rapidity differences being the same in all reference frames. Sometimes a high-energy approximation to rapidity called pseudorapidity is used instead, since it can be easier to calculate. It can be defined as

$$\eta = -\tan(\theta/2), \tag{2.3}$$

where θ is the angle from the beam axis.

The largest and most powerful particle accelerator as of writing this thesis is the Large Hadron Collider (LHC) in Switzerland. It is a ring 27 km in circumference that accelerates beams to create collisions at energies of 14 TeV (this corresponds to protons travelling at 99.9999991% the speed of light in the lab reference frame). These collisions occur at an instantaneous luminosity of $10^{34}\mathrm{cm}^{-1}\mathrm{s}^{-1}$ [23].

This particle accelerator is accompanied by the ATLAS detector, which collects data from 40 million collisions per second. This data contains information on the trajectories of many types of object, such as charged leptons or photons. Another important object type ATLAS tracks is known as a jet. These occur when quarks are launched away from collisions at high energies. Strong interactions among those energetic quarks cause the production of many more quarks and antiquarks, which all shoot off in the same direction through the detector as large streams (or jets).

One of the most important impacts of this astounding quantity of data is the 2012 discovery of the Higgs boson [16]. This relatively new particle can be produced in many ways, but one of the most common production methods at ATLAS is through Vector Boson Fusion (VBF), shown in Figure 2.3.

Once a Higgs boson is produced, through VBF or otherwise, it will quickly decay into other particles. When the Higgs decays specifically to two W bosons, which then decay to

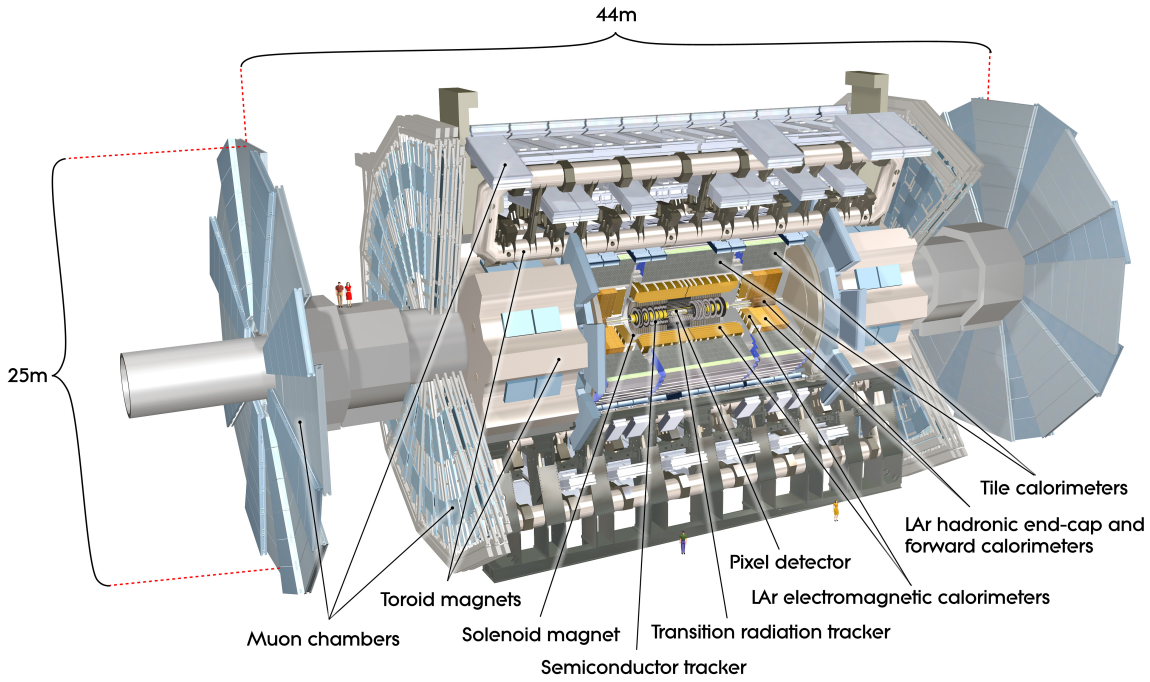

Figure 2.2: A computer-generated image of the ATLAS detector [35].

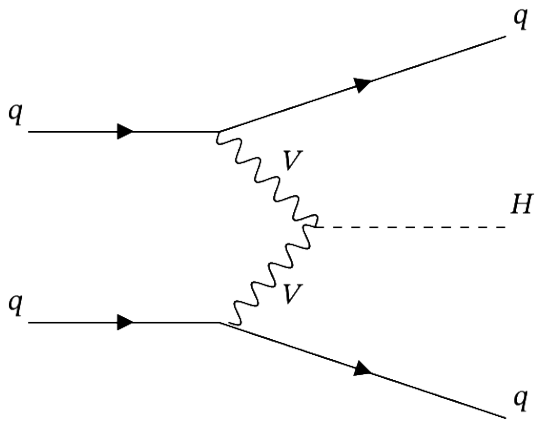

Figure 2.3: Feynman diagram of Vector Boson Fusion creating a Higgs boson.

two different-flavour leptons such as an electron and a muon, the paths these leptons take through the ATLAS detector is uniquely identifiable. This process is shown in Figure 2.4

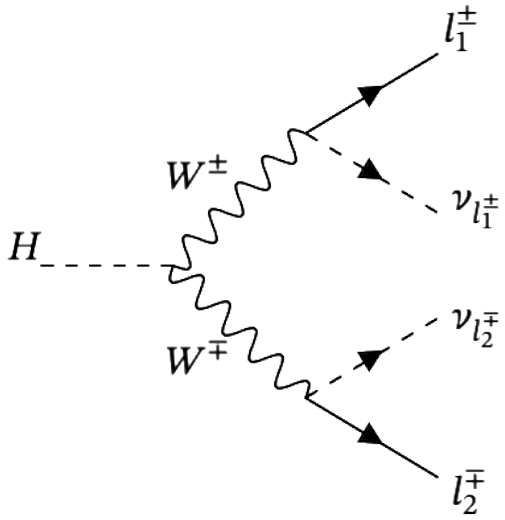

Figure 2.4: Feynman diagram of a Higgs to WW to two lepton decay.

In this thesis, we use simulated versions of these indentifiable VBF Higgs to WW events at the ATLAS collider to train and test a machine learning algorithm with the goal of constraining theories beyond the Standard Model.

2.2 Statistical Inference

Assume that we have a null hypothesis H_0, an alternative hypothesis H_1, a set of data $\mathbf{x}$ generated according to either H_0 or H_1, and a specified acceptable probability p of rejecting the null hypothesis when it is in fact correct. The Neyman-Pearson lemma guarantees that the likelihood ratio test incorrectly rejects the null hypothesis with at most a probability p while minimizing the chances of accepting the null hypothesis when it is false [34]. In order to perform this likelihood ratio test, we reject the null hypothesis in favour of the alternative hypothesis if and only if the inequality

$$L(H_0, H_1, \mathbf{x}) = \frac{\mathcal{L}(H_1|\mathbf{x})}{\mathcal{L}(H_0|\mathbf{x})} > \alpha, \tag{2.4}$$

holds, where $\mathcal{L}(H|\mathbf{x})$ is the likelihood of a hypothesis H given the data $\mathbf{x}$, and α is a number chosen based on our p value. More precisely, α is the value for which, assuming H_0 is true, we would expect to observe a dataset $\mathbf{x}'$ which makes the inequality hold with probability p.

Unfortunately, for most practical purposes, particle physics included, the distribution of $L(H_0, H_1, \mathbf{x}')$ assuming H_0 is not known, which makes calculating α impossible. We found two solutions to this issue. The first is to generate an empirical distribution by simulating many datasets and calculating the likelihood ratio for each of them. Though this method works without any extra requirements, it is computationally expensive. The second is to apply Wilks' Theorem [39], which is what we will use in this thesis.

Let Ω be a set of hypotheses where for each hypothesis, parameters $(\theta_1, \theta_2, \ldots, \theta_n)$ determining the probability density of some variable x fall into some region $S \subseteq \mathbb{R}^n$. Let $\omega \subseteq \Omega$ be another set of hypotheses where those $(\theta_1, \theta_2, \ldots, \theta_n)$ fall into some $T \subseteq \text{interior}(S)$ for each hypothesis. Let $\mathbf{x}$ be a set of realizations of x, and finally, define the test statistic

$$\Lambda = -2 \ln \left(\frac{\sup\{\mathcal{L}(H_0|\mathbf{x}) | H_0 \in \omega\}}{\sup\{\mathcal{L}(H_1|\mathbf{x}) | H_1 \in \Omega\}} \right). \tag{2.5}$$

Wilks' Theorem states that under certain assumptions, and as the number of data samples goes to infinity, Λ asymptotically follows a χ^2 distribution with degrees of freedom equal to the number of free parameters of Ω minus the number of free parameters of ω.

Given enough data, this asymptotic distribution allows us to determine α when the null hypothesis assumes that the parameters of interest take on specific values, while the alternative hypothesis allows the parameters to range over all possible values. Scanning over all possible values of our parameters and using these results to accept or reject each value then allows us to construct minimal confidence intervals for each parameter in our hypotheses.

2.3 Particle Physics Likelihood Ratio

As shown above, if we can obtain the likelihood ratio between two hypotheses for a set of collected data, it is then possible to obtain minimal confidence intervals for any parameters that determine these hypotheses. In the context of particle physics, this likelihood ratio takes a very specific form.

Let $x_i \in X$ represent the data collected for the ith event in our search region X. This could be as simple as a single measurement of the final transverse momentum of the event, or it could be as complicated as an array of hundreds of measurements performed by hundreds of instruments. In all cases, any two hypotheses H_0 and H_1 for the form of the Lagrangian uniquely determine the total cross-section $\sigma(X|H_j)$ for an observation in the search region, and the fully differential cross-section $\frac{\mathrm{d}\sigma}{\mathrm{dx}}(x_i|H_j)$ for observing x_i. As mentioned in Section 2.1, these quantities give us the probability density for x_i, or in other words, the likelihood of the hypothesis H_j as

$$p(x_i|H_j) = \mathcal{L}(H_j|x_i) = \frac{1}{\sigma(X|H_j)} \frac{\mathrm{d}\sigma}{\mathrm{d}x}(x_i|H_j). \tag{2.6}$$

Multiplying the total cross-section by the luminosity of the experiment that we are working with also gives the total expected number of events $N(H_j)$. This information is enough to calculate the likelihood ratio between H_0 and H_1.

Given a dataset $\mathbf{x} = (x_1, x_2, \ldots, x_{\mathcal{N}})$ with $\mathcal{N}$ measurements, we can calculate the likelihood ratio

$$L(H_0, H_1, \mathbf{x}) = \frac{p(\mathbf{x}|H_1)}{p(\mathbf{x}|H_0)} = \frac{P(\mathcal{N} \text{ events}|H_1) \prod_{i=1}^{\mathcal{N}} p(x_i|H_1)}{P(\mathcal{N} \text{ events}|H_0) \prod_{i=1}^{\mathcal{N}} p(x_i|H_0)}, \tag{2.7}$$

where $P(\mathcal{N} \text{ events}|H_j)$ is the probability of observing $\mathcal{N}$ events given the hypothesis H_j, and $p(x_i|H_j)$ is the probability density of observing the data x_i given the hypothesis H_j and that an event has occurred.

In particle physics experiments, the rate of interactions is large enough that the number of events observed in a small region of parameter space such as X is approximately sampled from a Poisson distribution. Replacing the probabilities with their respective Poisson probability mass functions and the probability densities with the expression given in Equation 2.6, then simplifying results in

$$L(H_0, H_1, \mathbf{x}) = \exp\left(N(H_0) - N(H_1) - \sum_{i=1}^{\mathcal{N}} \ln\left(\frac{\frac{\mathrm{d}\sigma}{\mathrm{d}x}(x_i|H_0)}{\frac{\mathrm{d}\sigma}{\mathrm{d}x}(x_i|H_1)}\right)\right). \tag{2.8}$$

From the form of the likelihood ratio, we can also see that

$$L(H_0, H_1, \mathbf{x}) = \frac{L(H_0, H^*, \mathbf{x})}{L(H_1, H^*, \mathbf{x})} \tag{2.9}$$

for any choices of H^*, H_0, and H_1. This means that in order to calculate the likelihood ratio between any two hypotheses H_0, H_1, we really only need to be able to directly calculate the likelihood ratio of any hypothesis with respect to one other hypothesis H^* of our choice. In particle physics, this one specific hypothesis is typically taken to be the Standard Model, and that is the convention that this thesis will adopt.

2.4 Standard Model Effective Field Theory

Several features stand out as being important to any model-independent approach to physics beyond the Standard Model. These features force the theory to agree with our experimental observations up to this point without being too constrained to describe new physics. The following are the features that Degrande et al. [24] cite as being most important:

- The theory should respect the symmetries of the Standard Model such as invariance under the action of the Poincaré group and $SU(3)_C \times SU(2)_L \times U(1)_Y$ gauge symmetry.
- It should respect unitarity and analyticity.

- It should reduce to the Standard Model for low energies.
- It should be general enough to represent any physics beyond the Standard Model while still providing guidance on where to find new effects.

Standard Model Effective Field Theory incorporates all of these requirements. The Standard Model by itself is "the most general theory of quark and lepton fields, along with a single Higgs doublet field, interacting via $SU(3)_C \times SU(2)_L \times U(1)_Y$ gauge symmetry with all operators restricted to be of energy dimension four or less" [24]. Thus, to incorporate new physics into this model as an Effective Field Theory, we simply add new operators $\mathcal{O}_i^{(d)}$ of energy dimension d higher than 4. In order to have the dimensions of these new terms match with the Standard Model terms, the operators must be suppressed by factors of some chosen energy scale Λ raised to the power of $d-4$. This means the Lagrangian for SMEFT is

$$\mathcal{L}_{\text{SMEFT}} = \mathcal{L}_{\text{SM}} + \sum_{d=5}^{\infty} \sum_i \frac{c_i^{(d)}}{\Lambda^{d-4}} \mathcal{O}_i^{(d)}, \tag{2.10}$$

where $\mathcal{L}_{\text{SM}}$ is the regular Standard Model Lagrangian, the $\mathcal{O}_i^{(d)}$ and the Λ^{d-4} are the operators and their suppression as mentioned above, and the $c_i^{(d)}$ are dimensionless parameters that are chosen to be real-valued by convention, known as Wilson Coefficients [14].

This linear dependence on the operators in the Lagrangian carries through to the quantum mechanical amplitude for a transition between initial and final states, known as the matrix element. The matrix element, which encodes all the dynamics of the transition process, can thus be written as

$$\mathcal{M}_{\text{SMEFT}} = \mathcal{M}_{\text{SM}} + \sum_{d=5}^{\infty} \sum_i \frac{c_i^{(d)}}{\Lambda^{d-4}} \mathcal{M}_i^{(d)}. \tag{2.11}$$

As we saw in the section on particle physics likelihood ratios, we can perform inference knowing only the differential cross-sections of events. These differential cross-sections are proportional to the absolute value of the matrix element squared,

$$|\mathcal{M}_{\text{SMEFT}}|^2 = |\mathcal{M}_{\text{SM}}|^2 + 2\sum_{d=5}^{\infty} \sum_i \frac{c_i^{(d)}}{\Lambda^{d-4}} \text{Re}\left(\mathcal{M}_{\text{SM}}^* \mathcal{M}_i^{(d)}\right) + \sum_{d,f=5}^{\infty} \sum_{i,j} \frac{c_i^{(d)} c_j^{(f)}}{\Lambda^{d+f-8}} \text{Re}\left(\mathcal{M}_i^{(d)*} \mathcal{M}_j^{(f)}\right), \tag{2.12}$$

where $\text{Re} : \mathbb{C} \to \mathbb{R}$ is the function mapping complex numbers to their real component, and z^* is used to denote the complex conjugate of any $z \in \mathbb{C}$. The suppression by powers of Λ seen above means that only the lowest-dimensional operators need to be taken into account to obtain accurate calculations for the differential cross-sections. Because operators of odd energy dimension all violate baryon or lepton number conservation [31], we will not consider their effects in this thesis. This leaves dimension-6 operators as those with the

largest impact. The standard choices of these dimension-6 operators are the 59 operators of the Warsaw Basis [26].

The interference between dimension-8 operators and the Standard Model term leads to effects that are only suppressed by Λ^4, which is on the same order of magnitude as the effects caused by interference between multiple dimension-6 operators. This means that ideally, we would include the effects of dimension-8 operators that are only suppressed by Λ^4, but these effects are still under theoretical development, so we will ignore them and consider only dimension-6 operators for this thesis.

Simplifying to include only dimension-6 operators and rearranging, we obtain the final form of the differential cross-section for any event,

$$\frac{\mathrm{d}\sigma_{\mathrm{SMEFT}}}{\mathrm{d}x}(x|c_1, c_2, \ldots, c_n) = \frac{\mathrm{d}\sigma_{\mathrm{SM}}}{\mathrm{d}x}(x) \sum_{i=1}^{n} \left(\delta_{i,1} + \sum_{j=2}^{n} g_{i,j}(x) c_{j-1} \right)^2 \tag{2.13}$$

where $\delta_{k,l}$ is the Kronecker delta, c_{j-1} are the dimension-6 operators, and $g_{i,j} : X \to \mathbb{R}$ are real functions which can be determined from the matrix elements and Λ. This specific form is chosen to ensure that the factor multiplying the Standard Model differential cross-section is nonnegative. Importantly, this quadratic dependence on the Wilson Coefficients and the overall form of the differential cross-section holds even if information about the event is lost during measurement, as the differential cross-section for the observed measurement is simply the integral over all possible events of these differential cross-sections, each of which is quadratic in the Wilson Coefficients.

Unfortunately, this quadratic dependence on the Wilson Coefficients does not quite carry over fully to experimental datasets. When collecting these datasets, certain thresholds are chosen to ensure that only the data from relevant events are included. This leads to the observed cross-sections in this experimental dataset being modified by some factor $A(c_1, \ldots, c_n)$ known as an acceptance fraction. Changing the values of the Wilson Coefficients can change how many events pass the chosen thresholds, leading to the acceptance fraction changing with the Wilson Coefficients in a way that is not necessarily quadratic. However, a Taylor expansion of A around the Standard Model, multiplication into the differential cross-sections, then discarding of terms of degree larger than c_j^2, leads to a quadratic approximation for small c_j. Because all data in this thesis is simulated directly without any cutoffs, the acceptance fraction does not affect our results. However, for this research to be applicable to experimental data, we would need to verify that the approximation is valid for the values of Wilson Coefficients that we consider.

2.5 The Quadratic Classifier

Equation 2.8 gives an expression for the likelihood ratio of any event in terms of the ratio between the differential cross-sections predicted by two hypotheses and the expected number

of events for both hypotheses. The expected total number of events is easy to simulate with Monte-Carlo generators such as MadGraph [4], and Equation 2.13 demonstrates that the differential cross-section ratios can be easily calculated for any value of the n Wilson Coefficients of interest assuming that we know the $(n^2+3n)/2$ functions $g_{i,j}$. Unfortunately, analytically calculating the $g_{i,j}$ requires integration over all possible histories that could have led to our observations, which is an intractable calculation.

Chen et al. [17] suggest that instead of trying to calculate these functions, we train neural networks $n_{i,j}$ to approximate them. As a concrete example, considering a single Wilson Coefficient, the equation for the differential cross-section ratio becomes

$$\frac{\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SMEFT}}}{\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SM}}}(x|c) = (1 + c\alpha(x))^2 + (c\beta(x))^2, \tag{2.14}$$

and training the neural networks $n_\alpha \approx \alpha$ and $n_\beta \approx \beta$ allows us to calculate the differential cross-section ratio, and thus the likelihood ratio for any event.

2.6 Neural Networks

A neural network is a specific kind of machine learning algorithm originally based around trying to model how human brains work. They attempt to model how in humans certain inputs will cause specific neurons to "activate" and send out signals. These signals act as inputs to even more neurons, which can in turn activate, and send even more signals. This chain of activations continues until it eventually leads to a desired outcome, such as identifying an image as containing a cat.

A neural network is an attempt to use this idea to get a computer to give outputs which may depend in a very convoluted and unintuitive way on the specific inputs. For example, neural networks can be used to determine the numeric value displayed in a hand-drawn image with very high accuracy simply by inputting the RGB pixel values for the image as a vector.

The most basic implementation of this is known as a feedforward neural network. In this implementation, the network is a chain of non-affine functions $f_n \circ f_{n-1} \circ \cdots \circ f_2 \circ f_1$, with each f_j mapping from $\mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_j}$. In detail, each f_j is defined as

$$f_j(x_1, x_2, \ldots, x_{d_{j-1}}) = \left(A\left(\sum_{k=1}^{d_{j-1}} w_{1,k}x_k + b_1 \right), \ldots, A\left(\sum_{k=1}^{d_{j-1}} w_{d_j,k}x_k + b_{d_j} \right) \right), \tag{2.15}$$

where $A : \mathbb{R} \to \mathbb{R}$ is a non-affine function called the activation function, the $w_{i,k} \in \mathbb{R}$ are known as the weights of the neural network, and $b_i \in \mathbb{R}$ are called the biases. The activation function, weights, and biases can be different for each of the functions in the chain. This entire process is illustrated in Figures 2.5 and 2.6.

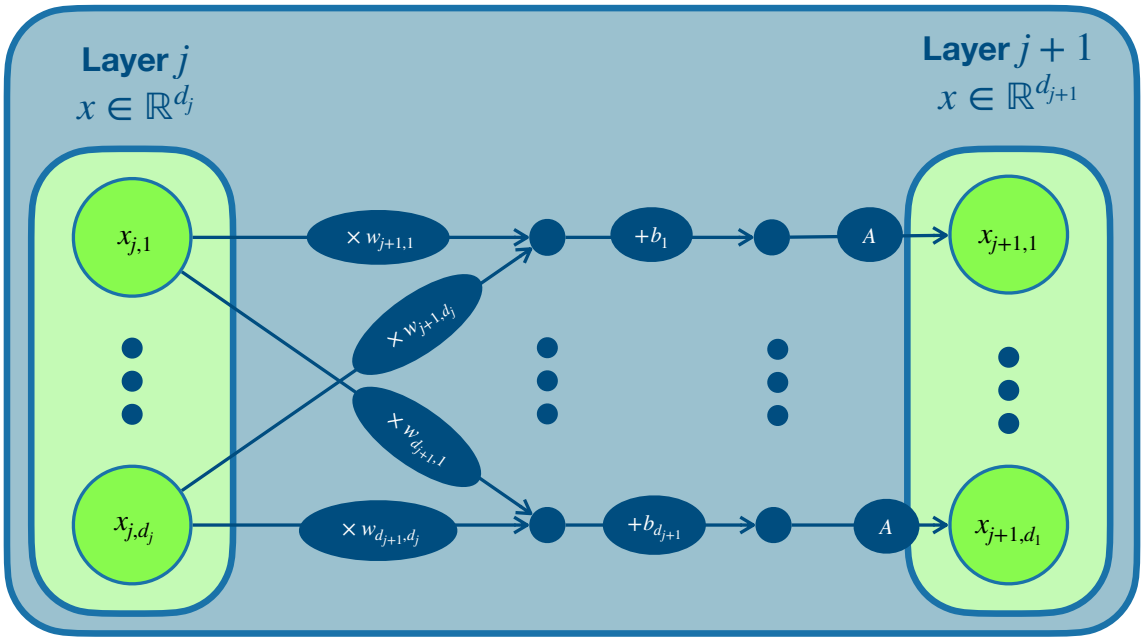

Figure 2.5: An illustration of one of the functions in a neural network.

When discussing neural networks, it is common to refer to the outputs of the functions f_j as layers. For example, the final output of the full composition is known quite straightforwardly as the output layer, while the outputs of the functions $f_1, \ldots, f_{n-1}$ are called hidden layers as they are never seen or used outside the evaluation of the network. The input to the function is sometimes known as the input layer, despite not actually being an output of the functions. In reference to its biological origins, each vector component of a layer is called a neuron. Thus, a neural network that has 3 hidden layers of 5 neurons each would be a chain of functions $\mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^5 \to \mathbb{R}^5 \to \mathbb{R}^5 \to \mathbb{R}^{d_{\text{out}}}$.

This general structure turns out to be very effective in a variety of situations from medical diagnoses to speech recognition, at least in part because feedforward neural networks are universal function approximators. More accurately, any Borel measurable function from one finite-dimensional space to another can be approximated to an arbitrary degree of accuracy using a feedforward neural network with the correct weights and biases [28].

Unfortunately, the "correct weights and biases" to approximate any given function are unknown in general. Determining these values for a specific problem is the "learning" in "machine learning". The simplest method to determine these values is using gradient descent. To perform gradient descent, one must decide on a function $\mathfrak{L} : \{g : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}\} \to \mathbb{R}$, mapping each function from $\mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ to some real number. When the function $\mathfrak{L}$, called loss, is evaluated on the neural network, it should measure how "incorrect" the output of the network is. Ideally, $\mathfrak{L}$ should be minimized if and only if the neural network is exactly equal to the function we are trying to approximate. With this

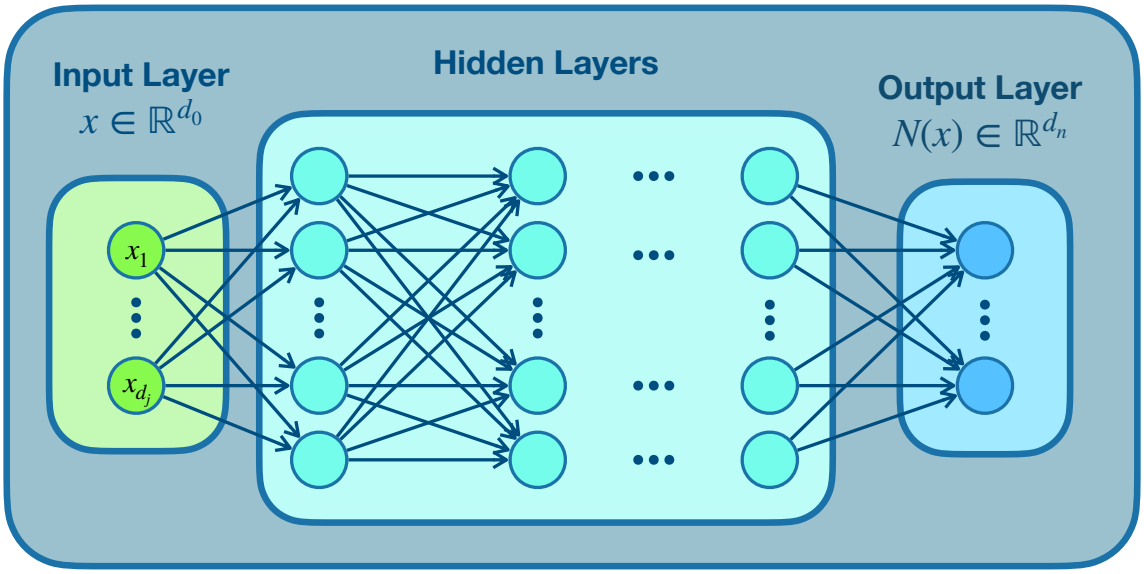

Figure 2.6: An illustration of a full neural network. Each circle represents a neuron, while the arrows show which neurons influence the following neurons.

loss function defined, determining the correct weights and biases becomes a minimization problem rather than an approximation problem.

Given any starting weights and biases p_0, we can find the gradient of the loss function with respect to these parameters. Each of the weights and biases is then adjusted by some small step proportional to the gradient to obtain new weights and biases p_1. This process is repeated, generating a sequence of parameter values $\{p_j\}$ which converges to a local minimum. In practice, we do not often use gradient descent directly. Instead, we mostly use optimization techniques inspired by gradient descent, but designed to converge more quickly and accurately.

We do not usually have a direct method for calculating the function we are trying to approximate. If we did, then we could use the function directly; we would not need to approximate it. This means that using a loss function that depends directly on the target function, such as the L^p distance between the neural network and the target function, is infeasible. However, it is not uncommon to have large quantities of inputs, along with their corresponding outputs. For example, even though we may not know what function to use to determine whether an image contains a cat or a bat just from its RGB values, we have thousands of images of cats and bats that humans have labelled. In any scenario where we do have such a large amount of data, we can use the data itself to define the loss function. Specifically, the loss function is often defined in such a way that $\mathfrak{L}$ is minimized if and only if the neural network outputs the correct value for each input in our dataset. Data used in this way is known as training data.

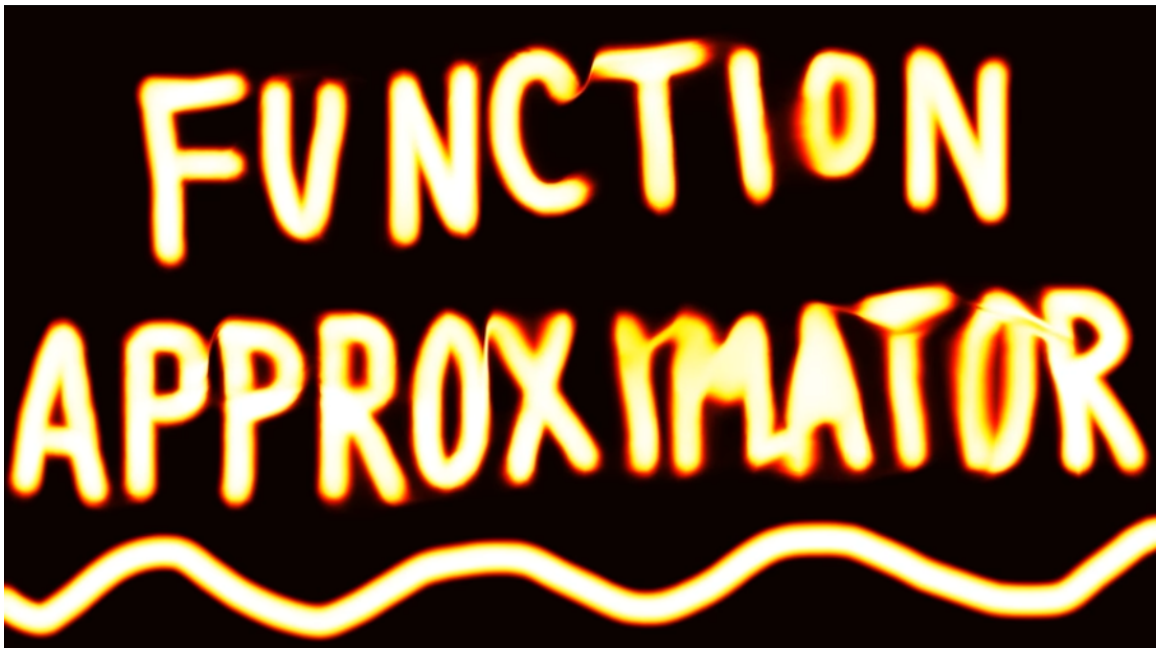

Figure 2.7: A demonstration of a neural network's ability to approximate arbitrary functions. The output of a network trained to approximate the 2-dimensional drawing of text and a squiggle is shown here. Image taken from [25].

Though the existence of optimal weights and biases is guaranteed, gradient descent and similar methods are not guaranteed to find them in a finite amount of time. The effectiveness of these optimization algorithms in finding good weights and biases depends on the structure of the neural network as well as the chosen loss function. Different choices of activation function, different numbers of layers, different numbers of neurons on each layer, and different loss functions can all affect convergence. Many optimization algorithms also have parameters built in that can be tuned to improve convergence. All of these extra parameters that are not trained, but instead must be chosen by humans are known as hyperparameters. To train the best-performing neural network, designers train networks with a variety of hyperparameters, evaluate their performances on a set of data known as validation data, and then choose the network that performs the best.

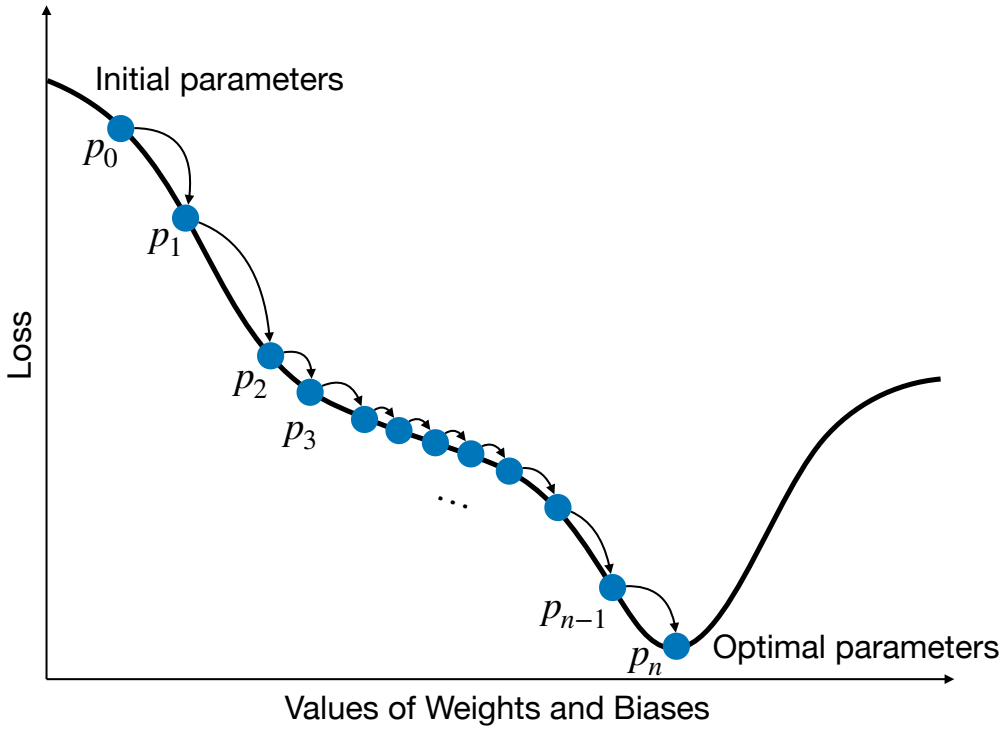

Figure 2.8: A simplified one-dimensional example of how gradient descent and similar methods minimize the chosen loss function by generating the sequence $\{p_j\}$.

Chapter 3

Implementation

3.1 Data Generation

All training and testing data used in this thesis was simulated. To create this data, we used MadGraph@NLO [4] to generate artificial VBF events. We then used MadGraph's reweighting feature along with the SMEFTsim package [12], [13] to reweight the generated events and include leading-order corrections to the Standard Model caused by nonzero Wilson Coefficients. We performed this procedure once for each value of $c_{\text{Hj3}} \in \{0, \pm 0.01, \pm 0.02, \pm 0.05, \pm 0.1, \pm 0.2, \pm 0.5, \pm 1.0, \pm 2.0, \pm 5.0\}$ with $c_{\text{HW}} = 0$, and again for c_{HW} ranging over those same values while $c_{\text{Hj3}} = 0$.

Each event was then passed through Athena [20] to remove events that did not include two jets of quarks and two leptons. This was done because we wanted to investigate whether our method effectively constrains Wilson Coefficients that take part in Higgs to WW to two lepton events (as mentioned in the section on particle physics). Two jets and two leptons are characteristic products of these types of events, so requiring them guaranteed that we were getting the events we expected.

Once this procedure was completed, we wrote the kinematic variables and weights for each event and for each Wilson Coefficient value to a file. The weights were the probability densities for each event, normalized to the total expected number of events rather than to 1. The set of kinematic variables is shown in Table 3.1. Since the events were simulated directly, the kinematics were exact, with no effects from the detector. We call this type of dataset a base dataset.

Variable Name	Description
lep0_id	Whether the most energetic charged lepton is an electron or a muon
lep1_id	Whether the second-most energetic charged lepton is an electron or a muon
lep0_pt	The momentum of the most energetic charged lepton transverse to the beam axis

Variable Name	Description
lep1_pt	The momentum of the second-most energetic charged lepton transverse to the beam axis
lep0_eta	The pseudorapidity of the most energetic charged lepton
lep1_eta	The pseudorapidity of the second-most energetic charged lepton
lep0_phi	The angle of the most energetic charged lepton around the beam axis
lep1_phi	The angle of the second-most energetic charged lepton around the beam axis
lep0_m	The mass of the most energetic charged lepton
lep1_m	The mass of the second-most energetic charged lepton
jet0_pt	The momentum of the most energetic jet transverse to the beam axis
jet1_pt	The momentum of the second-most energetic jet transverse to the beam axis
jet0_eta	The pseudorapidity of the most energetic jet
jet1_eta	The pseudorapidity of the second-most energetic jet
jet0_phi	The angle of the most energetic jet around the beam axis
jet1_phi	The angle of the second-most energetic jet around the beam axis
jet0_m	The centre-of-mass energy of the most energetic jet
jet1_m	The centre-of-mass energy of the second-most energetic jet
met_et	The magnitude of the missing momentum transverse to the beam axis
met_phi	The angle of the missing momentum around the beam axis
Mll	The centre-of-mass energy of the system consisting of the two most energetic leptons
Ptll	The momentum transverse to the beam axis of the system of the two most energetic leptons
DPhill	The angle between the two most energetic leptons around the beam axis
DEtall	The difference in pseudorapidity between the two most energetic leptons
DYll	The difference in rapidity between the two most energetic leptons
Mjj	The centre-of-mass energy of the system consisting of the two most energetic jets
Ptjj	The momentum transverse to the beam axis of the system of the two most energetic jets
DPhijj	The angle between the two most energetic jets around the beam axis
DEtajj	The difference in pseudorapidity between the two most energetic jets
DYjj	The difference in rapidity between the two most energetic jets

Variable Name	Description
nJets	The total number of jets
sqrtHT	The square root of the sum of the magnitudes of the momenta transverse to the beam axis of the leptons, the jets, and the missing momentum
METSig	The significance of the missing momentum
Ml0j0	The centre-of-mass energy of the most energetic lepton and jet together
Ml0j1	The centre-of-mass energy of the most energetic lepton and the second-most energetic jet together
Ml1j0	The centre-of-mass energy of the second-most energetic lepton and the most energetic jet together
Ml1j1	The centre-of-mass energy of the second-most energetic lepton and jet together

Table 3.1: The variables included for each event in the base datasets we generated

The data format given by this generation procedure is useful for training. When evaluating confidence intervals however, we need a dataset consisting of random events sampled from Poisson distributions according to their weights. This is the type of output that would be expected from actual collider data since the weights are the expected numbers of events of each type. In order to obtain datasets like this for testing purposes, we take a base dataset and choose some value of the Wilson Coefficients. Then for each event in the base dataset, we sample a Poisson distribution with mean value equal to the weight of the event given the chosen value of the Wilson Coefficients. This results in some natural number, typically 0 or 1, though it can rarely give larger numbers. We include each event from the base dataset a number of times equal to the random number obtained for that event.

For example, assume the base dataset contains some event with kinematic variables x_j that has a weight of 0.5 for $c_{\text{Hj3}} = 0, c_{\text{HW}} = 0.1$. Then, if we wanted to generate a dataset at those Wilson Coefficient values, we would sample a natural number n_j from a Poisson distribution with mean 0.5. Our kinematics x_j would be included in the final dataset n_j times. So, if $n_j = 0$, the value x_j would not appear in the dataset. If $n_j = 1$, we would include x_j in the dataset once, and if n_j were larger than 1, x_j would be included more than one time.

One issue with this method is that base datasets can sometimes include nonphysical negative weights for certain events. This occurs when MadGraph adds corrections to the leading-order weight calculations in order to de-emphasize the region of phase space around the events with negative weights. This does not affect the training of our neural network

as the likelihood ratio calculation only incorporates the weights through the ratio of the differential cross-sections. These ratios are always positive, regardless of the sign of the weights. However, the inclusion of the negative weights does affect this collision dataset generation method.

To account for the negative weights de-emphasizing specific regions of phase space in the collision dataset, we actually use the absolute value of the weights when defining the Poisson distribution in the above procedure. The events with negative weights are flagged in the final dataset so that during the confidence interval calculation they are subtracted from the sum rather than added. This ensures that the de-emphasis on the regions of phase space including these events is incorporated into the final calculation, with de-emphasis proportional to the weight assigned by MadGraph.

3.2 Neural Network Design

We designed our machine learning algorithms in Python with the modules Tensorflow [2] and Keras [18]. For this thesis, we only trained our likelihood ratio approximators to consider the variation of a single Wilson Coefficient at a time in order to test the simplest possible version. Thus, for each approximator, we only needed to train the two neural networks n_α and n_β from Section 2.5.

For each approximator, both networks were implemented as separate fully connected feed-forward neural networks with a Keras Normalization layer at the start to give the data 0 mean and a standard deviation of 1. We tested several combinations of layer number and neuron number for these networks. The accuracy of networks at different sizes is covered in the next chapter. In each case however, we kept the widths of the layers at 10 times the depths, as recommended by Adcock and Dexter [3].

We tested both ReLU [33] and ELU [19] activation functions for the hidden layers in our networks, while simple linear activation functions were used on the output layers of both n_α and n_β.

Similarly, we tested a variety of loss functions to train our neural networks. These loss functions and their results will be covered in more depth in the next chapter. However, there are several commonalities we can cover here. Given a set of events $\mathbf{x}$ along with the differential cross-section ratios for each event x_j and Wilson Coefficient value of interest $c_k \in \mathcal{C}$, each loss function took the form

$$\mathfrak{L}_\mathcal{E}(n_\alpha, n_\beta|\mathbf{x}, \mathcal{C}) = \frac{1}{(\#\mathbf{x})(\#\mathcal{C})} \sum_j \sum_k |aw(x_j|c_k)|\mathcal{E}(x_j|c_k, n_\alpha, n_\beta), \tag{3.1}$$

where $\#$ denotes cardinality, a is some positive constant, $w(x_j|c_k) = \frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SMEFT}}(x_j|c_k) + \frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SM}}(x_j)$, and $\mathcal{E}(x_j|c_k, n_\alpha, n_\beta)$ is some error function with a minimum when the output of n_α and n_β are perfectly accurate. Both n_α and n_β are used in all these loss functions, so even

though they were disconnected neural networks, they were trained at the same time and their parameters were updated as though they were a single network. The absolute value of $w(x_j|c_k)$ is used in the loss function instead of the value directly because, as mentioned in the previous section, sometimes the weights used for training can be negative. The absolute value ensures that the training minimizes all the errors instead of maximizing those with negative weights.

Before training, weights and biases were all randomly initialized according to normal distributions with mean 0 and variance 0.01, as recommended by Adcock and Dexter [3].

All training was performed using the Adam optimizer [30], with the Keras default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. We added norm clipping with a value of 1.0 to limit the optimization step size at each iteration. We also set the learning rate to exponentially decay over the course of the training. Its value was given by $0.01 \cdot 0.9^{t/100}$, where t is the number of training iterations, or "epochs", that have passed. This allowed the network to learn quickly initially, then to take smaller steps later in the training to better localize a minimum, improving the convergence rate and accuracy by several orders of magnitude in some of our proof of concept tests.

Finally, we also added batch normalization [8] between each hidden layer, as we found that this led to more consistently accurate training in our proof of concept tests.

3.3 Constraining Coefficients with Likelihood Ratios

With the neural networks n_α and n_β trained, we can construct a log-likelihood estimator as

$$\ln(\hat{L}(c|\mathbf{x})) = N_{\text{SMEFT}}(c) - N_{\text{SM}} - \sum_j \ln((1 + cn_\alpha(x_j))^2 + (cn_\beta(x_j))^2), \tag{3.2}$$

where N_{SMEFT} and N_{SM} are the expected number of observed events. This estimator approximates the log-likelihood ratio with the Standard Model in the numerator and SMEFT with the Wilson Coefficient of interest c in the denominator.

We can then numerically find its minimum $\ln(\hat{L}_{\min}(\mathbf{x}))$ with respect to the Wilson Coefficient, which corresponds approximately to the Wilson Coefficient with the maximum likelihood given our observed data. Using $\ln(\hat{L}_{\min}(\mathbf{x}))$, we can then construct an estimator for the test statistic Λ from Equation 2.5,

$$\hat{\Lambda}(c) = -2\left(\ln(\hat{L}_{\min}(\mathbf{x})) - \ln(\hat{L}(c|\mathbf{x}))\right) \approx \Lambda. \tag{3.3}$$

Assuming this estimator accurately approximates the test statistic, it should asymptotically follow a χ^2 distribution with 1 degree of freedom if the Wilson Coefficient is truly equal to c and the likelihood uses enough data.

Using this asymptotic distribution, we can determine that we should reject the hypothesis that the Wilson Coefficient is equal to c if and only if

$$\hat{\Lambda}(c) > \alpha = F^{-1}_{\chi^2,1}(1-p), \tag{3.4}$$

where $F^{-1}_{\chi^2,1}$ is the inverse cumulative distribution function for a χ^2 distribution with 1 degree of freedom, and p is the confidence level for the desired confidence interval.

With this value determined, we can numerically obtain the values of the Wilson Coefficient for which the inequality holds, as shown in Figure 3.1.

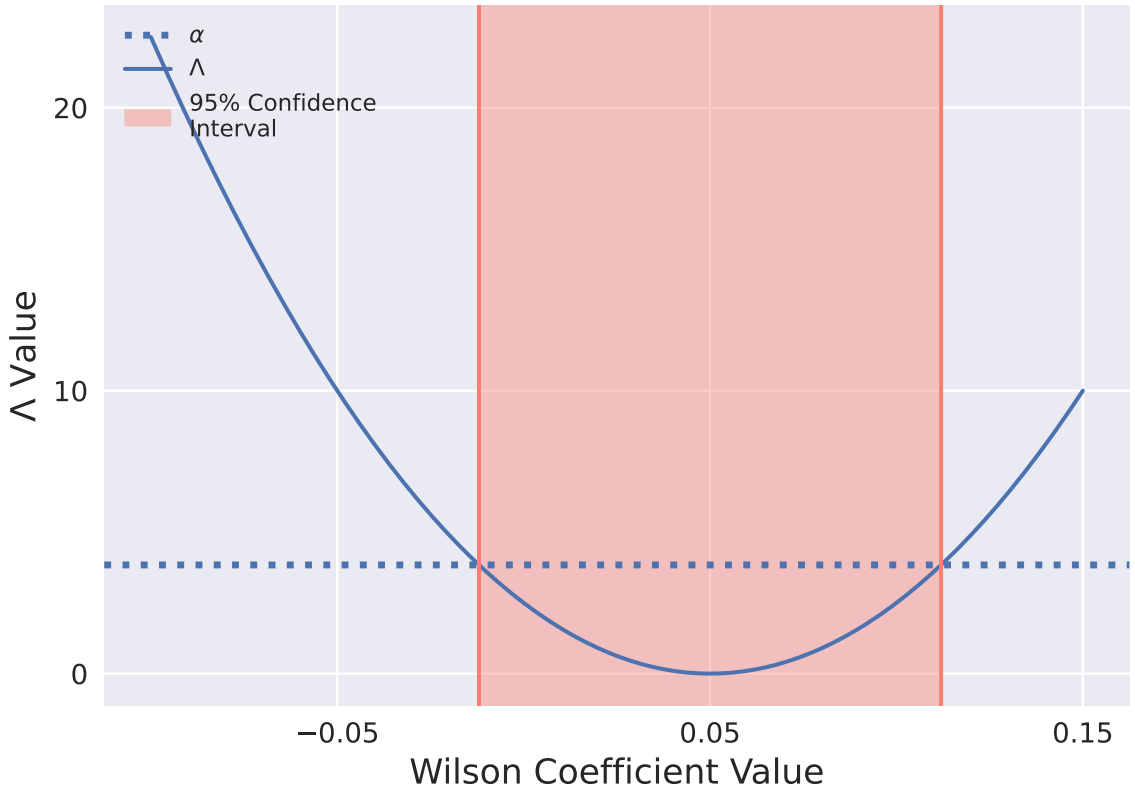

Figure 3.1: An example construction of a confidence interval. The dotted line is the value of $\alpha = F^{-1}_{\chi^2,1}(1-p)$ with $p = 0.95$, the solid blue line is an example test statistic one might expect to find, and the red span shows the calculated confidence interval.

Chapter 4

Results

4.1 Proof of Concept

To confirm that the above-described procedure does indeed generate minimal confidence intervals for the Wilson Coefficients given accurate enough neural networks, we decided to first run a proof of concept experiment.

To test the method with perfectly accurate neural networks, we generated base datasets of only 1000 events each for c_{HW} and c_{Hj3}. Using these smaller datasets allowed the networks to memorize them perfectly in a relatively short time. Specifically, we trained networks of 10 hidden layers with 100 neurons each using a loss function where $\mathcal{E}$ from Equation 3.1 was given by

$$\mathcal{E}(x|c, n_\alpha, n_\beta) = \left(\ln\left((1 + cn_\alpha(x))^2 + (cn_\beta(x))^2\right) - \ln\left(r(x|c)\right)\right)^2, \tag{4.1}$$

with r the differential cross-section ratio $\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SMEFT}}/\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SM}}$. This $\mathcal{E}$ is the residual of our log differential cross-section estimator squared (sometimes called the squared error). We trained our networks on this loss function for 50 000 epochs, leading to mean values of $\mathcal{E}$ on the order of 10^{-6} across all c and x for both Wilson Coefficients.

We then generated random collision datasets sampled from the 1000-event base dataset for each Wilson Coefficient value in $\{0, \pm 0.05, \pm 0.2, \pm 1.0\}$, and constructed 95% confidence intervals according to the procedure from the previous section. Figures 4.1 and 4.2 show representative confidence intervals for c_{HW} and c_{Hj3}, respectively. Each generated confidence interval not shown in a figure also contained the true Wilson Coefficient value and had a width within 0.002 of those shown in the figures.

Qualitatively, this demonstrates that the procedure generates the expected confidence intervals. In order to quantitatively test that the method was giving intervals that contained the true Wilson Coefficient values with the correct confidence levels, we generated 100 different collision datasets for each Wilson Coefficient, sampled from the 1000-event base dataset with the assumption that the corresponding Wilson Coefficient was equal to 0.05. For each of these datasets, we produced a 75% confidence interval and tested whether it

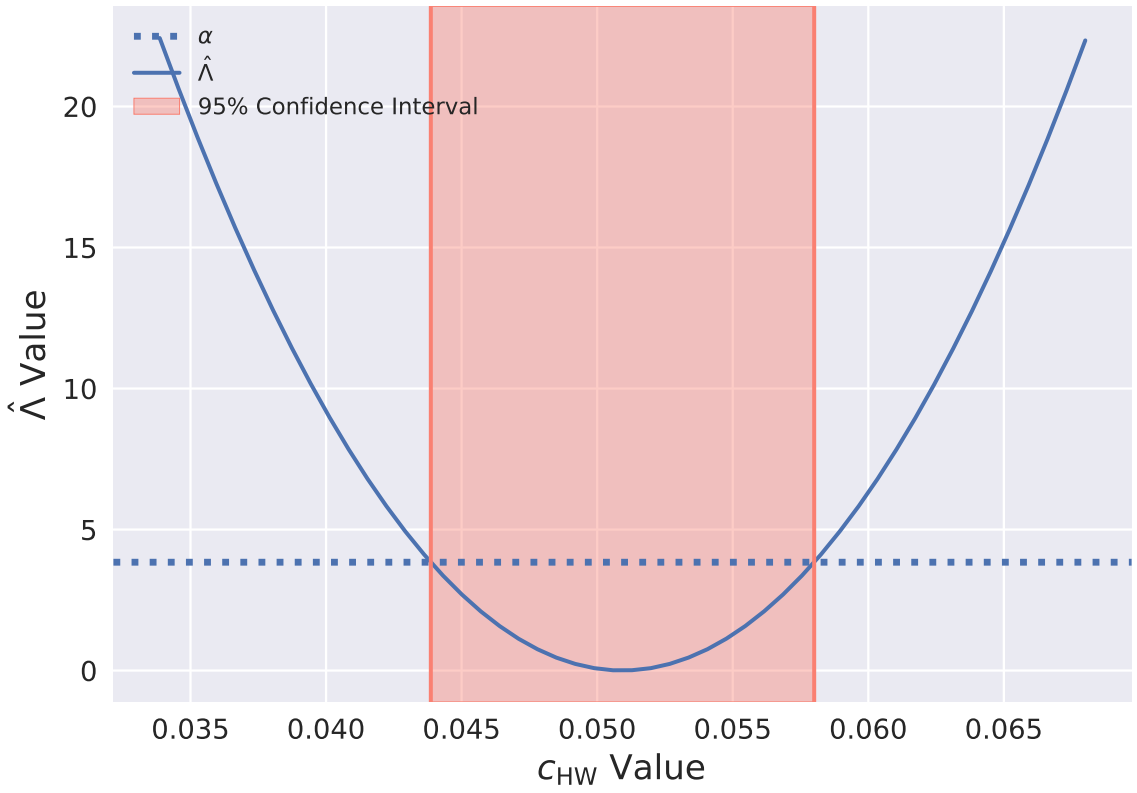

Figure 4.1: A representative 95% confidence interval for a collision dataset sampled from the 1000-event training data with $c_{\mathrm{HW}} = 0.05$. The dotted line is the value of $\alpha = F^{-1}_{\chi^2,1}(1-p)$ with $p = 0.95$, the solid blue line is the test statistic estimator generated with the perfectly trained neural networks, and the red span is the generated confidence interval.

contained the true Wilson Coefficient value of 0.05. With the c_{HW} datasets, we found that a total of 73 of the 100 confidence intervals contained 0.05. For c_{Hj3}, we found that a total of 77 of the 100 confidence intervals contained 0.05. Both of these results fall well within the 4.3 standard deviation of a binomial distribution with a 75% probability of success and 100 events. This provides evidence that the generated confidence intervals do indeed have the requested confidence level.

Our tests indicate that we have enough data for Wilks' Theorem to apply, they show that the procedure works for a range of Wilson Coefficient values, and they demonstrate the level of constraint that would be expected if this method were to work on experimental data. The 95% confidence intervals we generated here show that this method has the potential to constrain both Wilson Coefficients to the order of 10^{-2}. The current best constraints on these Wilson Coefficients from VBF $H \to WW$ data are on the order of unity [1]. Thus, if our procedure were to work on experimental data, we could expect to make constraints up to 100 times tighter than state-of-the-art constraints.

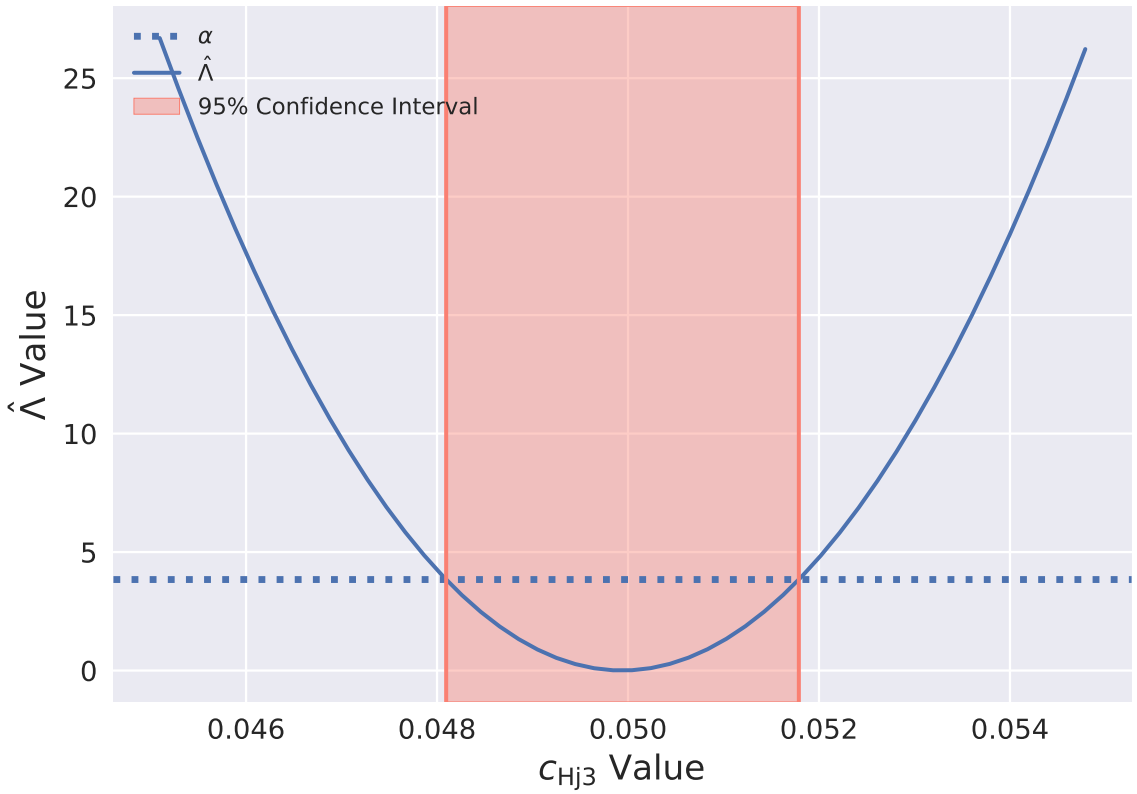

Figure 4.2: A representative 95% confidence interval for a collision dataset sampled from the 1000-event training data with $c_{Hj3} = 0.05$. The dotted line is the value of $\alpha = F^{-1}_{\chi^2,1}(1-p)$ with $p = 0.95$, the solid blue line is the test statistic estimator generated with the perfectly trained neural networks, and the red span is the generated confidence interval.

We must note, however, that these constraints are not directly comparable to actual experimental results and serve more as a guideline for what could, theoretically, be achievable. Our data does not take into account systematic uncertainties from the experimental setup or statistical uncertainties in measured observables. Furthermore, these tests were performed without any background polluting our data — though with perfectly trained networks, this should in principle have no impact on the intervals. Including all of these would almost certainly weaken constraints obtained using our method. Still, these results are promising. An improvement by 2 orders of magnitude is unlikely to be completely undone by incorporating experimental uncertainty. Additionally, the Neyman Pearson Lemma tells us that these intervals give the tightest constraints possible without more data or more accurate measurements, which further indicates that this improvement upon previous results should be expected.

4.2 Full Test

We will give our full results first and then cover our observations from obtaining these results in the following section. Figures 4.3 and 4.4 show confidence intervals generated for a range of Wilson Coefficient values using the most accurate neural networks that we were able to train.

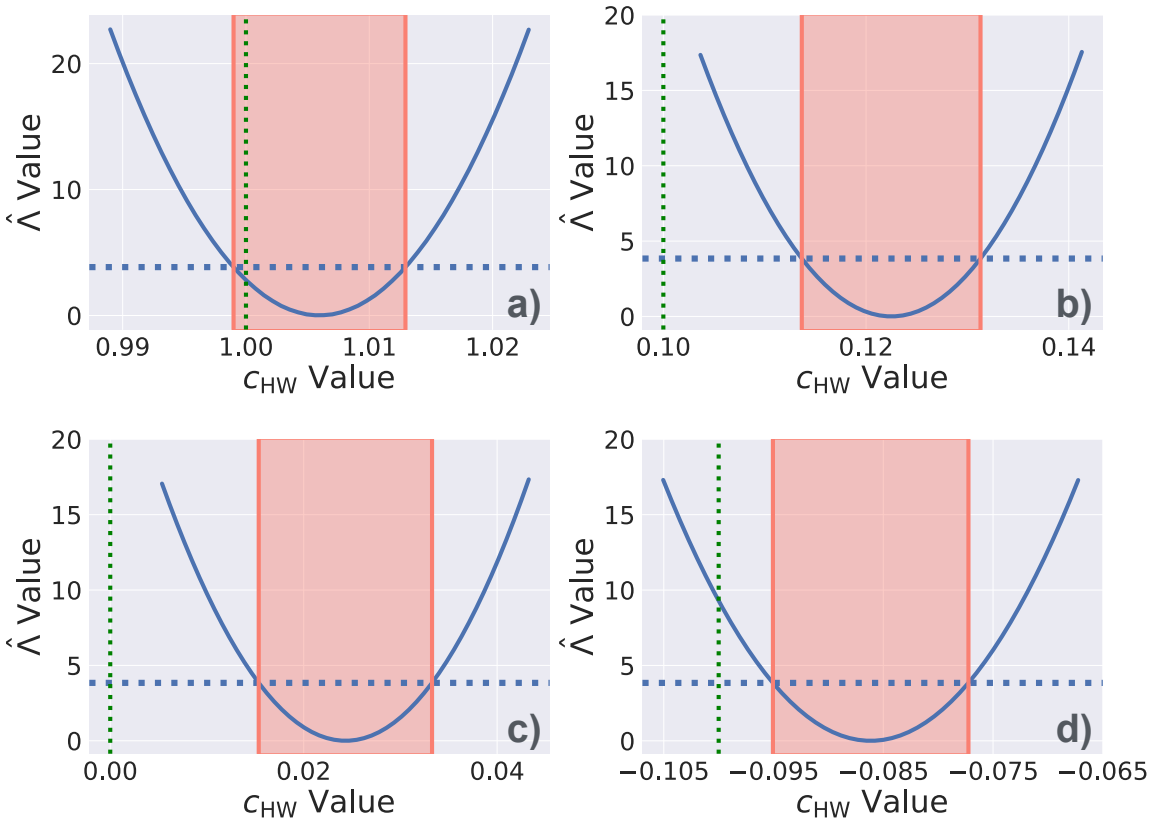

Figure 4.3: The best 95% confidence intervals for c_{HW} that we were able to obtain. All datasets used in the production of these figures were sampled from the 12M-event network training data. Plot a) was generated with $c_{\mathrm{HW}} = 1$, b) with $c_{\mathrm{HW}} = 0.1$, c) with $c_{\mathrm{HW}} = 0$, and d) with $c_{\mathrm{HW}} = -0.1$. The solid blue lines in the plots are the test statistic estimator outputs. The horizontal dotted lines are the values of $\alpha = F^{-1}_{\chi^2,1}(1-p)$ with $p = 0.95$. The red spans are the confidence intervals. The dotted vertical green lines show the true value of c_{HW} used to generate data for each plot.

These intervals were created by evaluating the neural networks on their training data. In general, neural networks perform better on this data than on previously unseen inputs. Based on the two figures, our networks were not even performing accurately enough for our method on their training data, so we decided to focus on improving training performance to the point where we could produce accurate intervals before worrying about generalization.

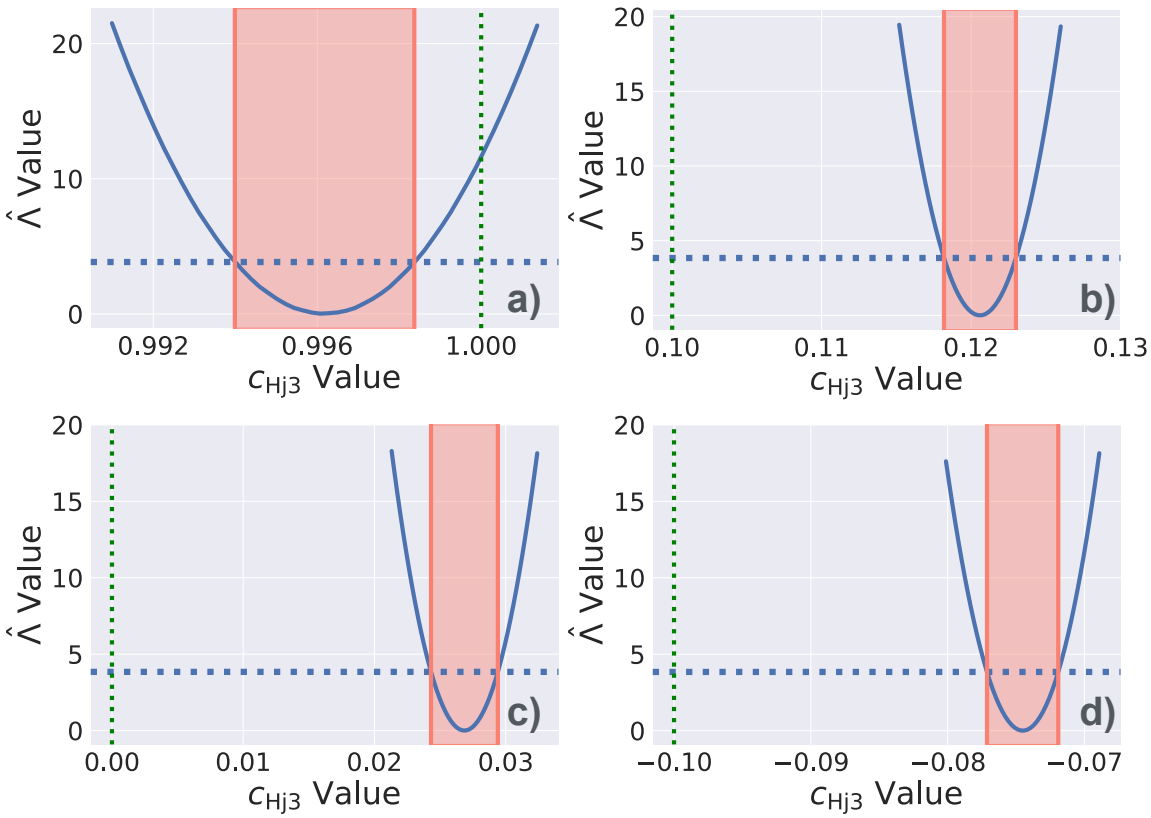

Figure 4.4: The best 95% confidence intervals for c_{Hj3} that we were able to obtain. All datasets used in the production of these figures were sampled from the 12M-event network training data. Plot a) was generated with $c_{\mathrm{Hj3}} = 1$, b) with $c_{\mathrm{Hj3}} = 0.1$, c) with $c_{\mathrm{Hj3}} = 0$, and d) with $c_{\mathrm{Hj3}} = -0.1$. The solid blue lines in the plots are the test statistic estimator outputs. The horizontal dotted lines are the values of $\alpha = F^{-1}_{\chi^2,1}(1-p)$ with $p = 0.95$. The red spans are the confidence intervals. The dotted vertical green lines show the true value of c_{Hj3} used to generate data for each plot.

Because of this, all results presented in this section and the next are only applicable to training data unless otherwise specified.

The accuracy of the networks was measured with the loss function from Equation 3.1, using $\mathcal{E}$ given by Equation 4.1. This is a useful measure of error because if we consider the log output of our network on any given event to be a realization of a random variable centred on the true log of the differential cross-section ratio, Eq. 4.1 represents a sample of the variance of that random variable. Since the entire log likelihood ratio estimation is the sum of individual log differential cross-section estimations, its variance is the sum of the individual log differential cross-section ratio variances, and hence proportional to our loss. Minimizing this metric is therefore equivalent to minimizing the error in our likelihood ratio estimation. This metric is called mean squared error.

The collision datasets used to produce the figures were generated from sampling the 12 million-event base dataset used to train the neural networks. These networks all had 10 hidden layers of 100 neurons, and were trained for 10 000 epochs using the loss function described in the previous section. They also used ELU activation functions for their hidden layers.

As can be seen in the figures, the confidence intervals did tend to be near the true Wilson Coefficient values, but rarely actually contained them. Among the eight subfigures displaying 95% confidence intervals, only one contained the true value, which is far fewer than should be expected assuming the intervals were correctly generated. Since we confirmed in the previous section that our procedure was functioning correctly, this allows us to conclude that the issue was that our networks were not producing accurate enough outputs.

Despite this, these figures do show some promising results. We can see that the confidence interval widths are still on the order of 10^{-2}. This helps to confirm that the constraint sizes determined by our proof of concept are correct regardless of how the networks behave. Furthermore, as mentioned above, the confidence intervals are always within 0.03 of the true Wilson Coefficient values. This indicates that even if our method is not completely accurate, it is still at least partially achieving its goal of identifying Wilson Coefficients. Notably, the offset on the order of 0.03 is still much smaller than the current best constraints on c_{HW} and c_{Hj3} from VBF $H \to WW$ data. Thus, it is possible that our method could be modified to produce constraints that are not minimal, but still on the order of 10^{-2} or 10^{-1}; much better than current methods.

4.3 Hyperparameter Optimization

In order to obtain the best-performing neural networks from the previous section, we tested hundreds of network architectures with a variety of optimizers, loss functions, activation functions, and methods of regularization. Since the entire procedure for both Wilson Coefficients is identical, and each training of a new neural network can take days to weeks, we decided to perform these tests only on c_{HW}, then carry the results over to c_{Hj3} to make the most efficient use of our limited time. Below, we summarize how various changes affected the results of our trainings.

The clearest and most straightforward improvements came from increasing the size of our neural networks. In general, we found that trainings run with larger networks had lower mean squared error. This was supported by a series of trainings we performed with identical hyperparameters apart from network size. Figure 4.5 shows the improvements in mean squared error as we increased network size, and displays sample confidence intervals for each network size. We can see that as size increases, the mean squared error decreases. The figure also shows that the confidence intervals trend closer to the true values for larger networks, as we would expect based on network accuracy. The random nature of the

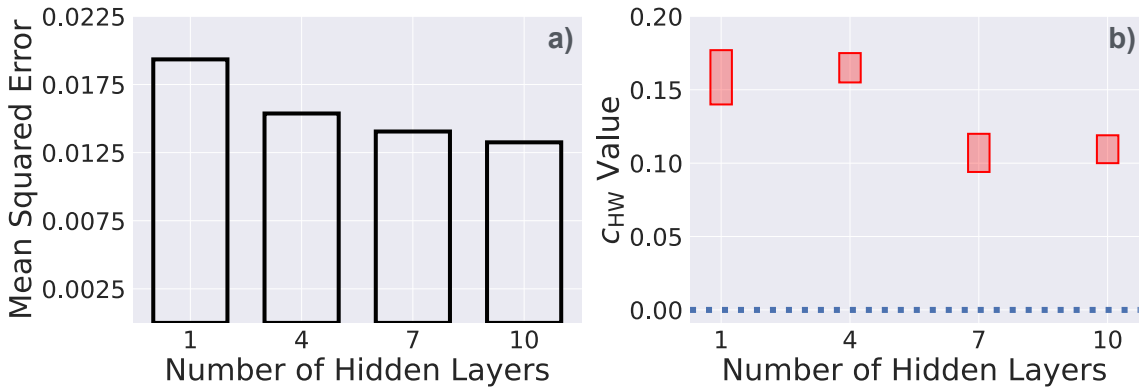

Figure 4.5: A comparison of neural network performance against size. Each hidden layer had 10 times as many neurons as the number of hidden layers. Plot a) shows the mean squared error of the network for four different network sizes. The red boxes in plot b) are sample confidence intervals for each network size. The blue dotted line in plot b) shows the true Wilson Coefficient value.

confidence intervals and the small sample size does, however, make the trend less consistent than that of the mean squared error.

This improvement as a function of network size is to be expected. Adding more parameters to most models should decrease the total error on the data being fit. For example, a polynomial with n parameters can perfectly fit up to n data points. This usually comes at the cost of decreased performance on data that was not part of the fit. In the case of neural networks, however, it has been found that drastically overparametrized models can actually perform better on both the fitting data and other previously unseen data [32]. Thus, not only is our observed increase in accuracy as a function of network size expected, we could also, in principle, continue to increase network size for further improvements. To reach this overparametrized region on our data, however, we would need a network depth in the hundreds.

Training such a large network would be difficult using our setup, as larger networks come with issues as well as their benefits. Each time network depth was increased by 3 and layer width was increased by 30, the duration of each epoch of training roughly tripled. In the tests we ran to produce Fig. 4.5, we also saw that the number of epochs we had to train before the loss reached its minimum increased by increments of approximately 100 for each larger network, starting from about 300. This meant that training time rapidly grew to unreasonable lengths for large networks. Some trainings with network depths of 10 or more also resulted in loss diverging to infinity. Though we were able to prevent this by implementing norm clipping in the optimizer, it could be indicative of issues such as exploding gradients [36], that often occur in larger networks. Still, increasing network size was the most obvious way to improve accuracy.

A different way to improve accuracy was changing the loss function. We started off our tests by training networks on the loss function introduced by Chen et al. [17]. In the large-sample limit they use, it can be shown through simple algebra that their loss function is equivalent to some constant plus the loss from Equation 3.1, with $\mathcal{E}$ given by

$$\mathcal{E}(x|c, n_\alpha, n_\beta) = \left(\frac{1}{1 + (1 + cn_\alpha(x))^2 + (cn_\beta(x))^2} - \frac{1}{1 + r(x|c)} \right)^2, \tag{4.2}$$

and r being the differential cross-section ratio $\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SMEFT}}/\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SM}}$. We call this loss the reciprocal loss function. Eventually, we decided to test other loss functions, and since we were trying to minimize the variance of our log likelihood ratio estimator, which was proportional to the mean squared error, it seemed appropriate to test the mean squared error loss from Equation 4.1. Through our testing, we found that the mean squared error loss led to lower values of mean squared error on the trained networks, and thus to improved performance in the generation of confidence intervals. To demonstrate this, we trained a set of otherwise identical networks with both of these loss functions. The mean squared error for the reciprocal loss was 2.27×10^{-2} and 1.43×10^{-2} for the mean squared error loss, supporting the conclusion that it was a better loss function for our purposes than the one from the paper introducing the Quadratic Classifier [17].

When we found that the mean squared error loss was still not producing accurate enough networks, we searched for other potential loss functions that might improve performance. Brehmer et al. [10] suggest that incorporating a term including the "score", or derivative of the log likelihood ratio near the Standard Model, into the loss function can improve network accuracy. We tried following this advice, and actually added another term with the second derivative as well just to test if it would help, leading to the derivative loss function, defined by

$$\begin{aligned} \mathcal{E}(x|c, n_\alpha, n_\beta) = A &\left[\ln\left((1 + cn_\alpha(x))^2 + (cn_\beta(x))^2 \right) - \ln\left(r(x|c) \right) \right]^2 \\ + B &\left[\frac{\partial}{\partial c} \left(\ln\left((1 + cn_\alpha(x))^2 + (cn_\beta(x))^2 \right) - \ln\left(r(x|c) \right) \right)_{c=0} \right]^2 \\ + C &\left[\frac{\partial^2}{\partial c^2} \left(\ln\left((1 + cn_\alpha(x))^2 + (cn_\beta(x))^2 \right) - \ln\left(r(x|c) \right) \right)_{c=0} \right]^2, \end{aligned} \tag{4.3}$$

where again r is the differential cross-section ratio $\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SMEFT}}/\frac{\mathrm{d}}{\mathrm{d}x}\sigma_{\mathrm{SM}}$, and $A, B, C \in \mathbb{R}$ are arbitrary hyperparameters that satisfy the constraint $A + B + C = 1$. We tested this loss function for 21 different combinations of A, B, C — defining two variables $X, Y \in \{0, 0.25, 0.5, 0.75, 1\}$, and then letting $A = X$, $B = (1-X)Y$, and $C = (1-X)(1-Y)$. Other than these hyperparameter changes, the networks were identical. Of all these trainings, the ones with the lowest mean squared error of 2.019×10^{-2} were those with $X = 1$, or in other words, just the mean squared error loss. The hyperparameter configuration with the second-

lowest mean squared error of 2.046×10^{-2} was given by $X = 0.75, Y = 0$. Based on these results, adding terms for the score or the derivative of the score to the loss function seems to just make accuracy worse. Furthermore, the calculation of the score and score derivative greatly increase the time it takes to evaluate the loss, and therefore the overall training time of the network. Thus, we concluded that the best loss function for our purposes was the mean squared error loss.

Originally, all our networks used ReLU activation functions for their hidden layers, but in line with Clevert et al. [19], we found in our proof of concept testing that ELU activation functions led to faster, more consistent, and more accurate convergence. ReLU activations gave mean squared errors on the order of 1 to 10, while ELU activations led to mean squared errors on the order of 10^{-6}. When we tried swapping the ReLU activations for ELU in our full tests with the 12 million-event datasets, we found that the impact was much smaller. ELU activations did, however, still seem to perform slightly better in general. This conclusion was supported by a test we ran on one set of networks with ReLU activations, and another with ELU activations, that were otherwise identical. The ReLU networks gave a mean squared error of 1.490×10^{-2}, while the ELU networks gave a mean squared error of 1.435×10^{-2}.

The final change that had a consistent impact on our training was to add batch normalization [8] between each hidden layer. Similar to the situation with the activation functions, we originally found in our proof of concept testing that adding in batch normalization helped the training to converge more consistently and accurately. When we tested this change with our full sized dataset the effect was much smaller, but it did still improve accuracy. With two otherwise identical networks, we found that training with batch normalization gave a mean squared error of 1.435×10^{-2}, while no batch normalization gave a mean squared error of 1.467×10^{-2}, which supports this conclusion.

Thus, the optimal structure we found for training our neural networks was to use as large a network as possible with ELU activation functions on each hidden layer, batch normalization between each layer, and minimizing the mean squared error loss.

Chapter 5

Conclusion

In this thesis, we explored a new method for constraining the c_{HW} and c_{Hj3} Wilson Coefficients of the Standard Model Effective Field Theory Warsaw Basis. To do so, we trained several versions of the Quadratic Classifier [17] on simulated Vector Boson Fusion $H \to WW$ data, then tested their effectiveness in producing confidence intervals.

As a proof of concept, we trained neural networks for the classifier on datasets of only 1000 events, allowing the networks to perfectly fit that specific data. We then evaluated these perfectly trained networks on artificial collision datasets sampled from the perfectly learned training data. With these evaluations, we were able to use our procedure to construct minimal confidence intervals for c_{HW} and c_{Hj3} in these simulated scenarios. This allowed us to confirm that with accurate neural networks, our method does indeed produce minimal confidence intervals for the Wilson Coefficients of interest. The width of these intervals also provided evidence that with accurate enough networks, it might be possible to constrain c_{HW} and c_{Hj3} down to the order of 10^{-2}. This is roughly 2 orders of magnitude smaller than the current best constraints from VBF $H \to WW$ processes [1].

However, once we included larger quantities of data with the goal of getting the classifier to generalize and produce accurate results on previously unseen events, we observed that the constraints contained the true values of the Wilson Coefficients too infrequently to be accurate confidence intervals, even just on the training data. Though the widths of the intervals stayed on the same order of magnitude no matter what, the inaccuracy of the networks was enough that summing over all events during the calculation of the log likelihood ratio led to significant biases in the positions of the confidence intervals. In fact, the majority of 95% confidence intervals did not include the true values of the Wilson Coefficients according to which the data had been generated.

In order to decrease these biases, we investigated varying many hyperparameters of our models. We found that replacing ReLU activation functions with ELU activations tended to improve accuracy. Similarly, adding in batch normalization between hidden layers improved performance. Furthermore, we found that the mean squared error loss function produced networks more accurate than a variety of other loss functions, including the one used by

Chen et al. [17]. Finally, we found that increasing network size consistently decreased network error. However, it also drastically increased training time for the networks and led to other issues associated with deep networks, such as exploding gradients [36].

There are many paths for future work related to this thesis. As increasing network size proved to be an effective method for improving accuracy in our tests, machine learning architectures such as residual neural networks [27], designed to allow for deeper networks while avoiding their pitfalls, could provide improved accuracy.

One could also attempt to incorporate the bias of the confidence intervals generated by inaccurate networks into a systematic uncertainty, sacrificing the minimal widths of the confidence intervals in order to actually constrain the coefficients. One way to do this would be to empirically modify our results and increase confidence interval size to produce accurate constraints. Since confidence intervals constructed using our method were often quite near to the true Wilson Coefficient values, generating many such intervals and modelling their distribution around the true Wilson Coefficients should be possible. On the theoretical side, since our log likelihood ratio calculation exclusively uses a large sum of network evaluations rather than the evaluation of the networks on any single event, it should be possible to describe the log likelihood ratio estimator value as the true log likelihood ratio plus some Gaussian random variable due to the central limit theorem. This could provide a starting point for a mathematical analysis of the systematic uncertainty in the confidence interval generation, allowing for this uncertainty to be incorporated into the intervals analytically.

If we were able to obtain accurate confidence intervals on simulated data, next steps would aim to apply this method to actual ATLAS data. The first step would be to train the networks on background events that cannot be filtered out based on kinematics alone, since the networks would need to deal with these events when applied to actual experimental data. This improved network could then be applied to current ATLAS data, or even the data from the Large Hadron Collider's upcoming operational run (Run-3) to obtain new constraints on c_{HW} and c_{Hj3}.

In this scenario, it would also be interesting to combine the likelihood ratio estimators for c_{HW} and c_{Hj3} into a single joint likelihood ratio estimator. As discussed by Chen et al. [17], this would require training a total of five neural networks — only one more than required for the two separate estimators trained in this thesis — and would give the full two-dimensional distribution of the Coefficients' likelihood ratios.

Finally, the procedure presented in this thesis would be relatively simple to apply to data beyond simply Vector Boson Fusion $H \to WW$ events. It is possible that other types of events could be learned more easily by neural networks, making this a more effective method when applied to that data. It could also allow for constraints on other Wilson Coefficients, or stronger constraints on the two studied in this thesis.

Bibliography

[1] G Aad, B Abbott, K Abeling, NJ Abicht, SH Abidi, A Aboulhorma, H Abramowicz, H Abreu, Y Abulaiti, AC Abusleme Hoffman, et al. Integrated and differential fiducial cross-section measurements for the vector boson fusion production of the Higgs boson in the H→ WW*→ e$\nu\mu\nu$ decay channel at 13 TeV with the ATLAS detector. *Physical Review D*, 108(2304.03053), 2023.

[2] 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. Software available from tensorflow.org.

[3] Ben Adcock and Nick Dexter. The gap between theory and practice in function approximation with deep neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2):624–655, 2021.

[4] 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), jul 2014.

[5] Jack Y Araz, Shankha Banerjee, Rick S Gupta, and Michael Spannowsky. Precision SMEFT bounds from the VBF Higgs at high transverse momentum. *Journal of High Energy Physics*, 2021(4):1–29, 2021.

[6] Gianfranco Bertone and Dan Hooper. History of dark matter. *Reviews of Modern Physics*, 90(4):045002, 2018.

[7] Tisa Biswas, Anindya Datta, and Biswarup Mukhopadhyaya. Following the trail of new physics via the vector boson fusion Higgs boson signal at the Large Hadron Collider. *Physical Review D*, 105(5):055028, 2022.

[8] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[9] Johann Brehmer, Kyle Cranmer, Felix Kling, and Tilman Plehn. Better Higgs boson measurements through information geometry. *Physical Review D*, 95(7), apr 2017.

[10] Johann Brehmer, Kyle Cranmer, Gilles Louppe, and Juan Pavez. A guide to constraining effective field theories with machine learning. *Physical Review D*, 98(5):052004, 2018.

[11] Johann Brehmer, Felix Kling, Tilman Plehn, and Tim M.P. Tait. Better Higgs-CP tests through information geometry. *Physical Review D*, 97(9), may 2018.

[12] Ilaria Brivio. Smeftsim 3.0 — a practical guide. *Journal of High Energy Physics*, 2021(4), April 2021.

[13] Ilaria Brivio, Yun Jiang, and Michael Trott. The smeftsim package, theory and tools. *Journal of High Energy Physics*, 2017(12), December 2017.

[14] Ilaria Brivio and Michael Trott. The standard model as an effective field theory. *Physics Reports*, 793:1–98, 2019.

[15] S Carlip. Quantum gravity: a progress report. *Reports on Progress in Physics*, 64(8):885, jul 2001.

[16] Dean Carmi, Adam Falkowski, Eric Kuflik, Tomer Volansky, and Jure Zupan. Higgs after the discovery: a status report. *Journal of High Energy Physics*, 2012(10):1–31, 2012.

[17] Siyu Chen, Alfredo Glioti, Giuliano Panico, and Andrea Wulzer. Parametrized classifiers for optimal EFT sensitivity. *Journal of High Energy Physics*, 2021(5):1–39, 2021.

[18] François Chollet et al. Keras. https://keras.io, 2015.

[19] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

[20] ATLAS Collaboration. Athena, June 2023.

[21] ATLAS collaboration et al. Methodology for EFT interpretation of Higgs boson Simplified Template Cross-section results in ATLAS. Technical report, ATL-PHYS-PUB-2019-042, 2019.

[22] Wikimedia Commons. File:Standard Model of Elementary Particles.svg. https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg, 2019. Accessed on November 14, 2023.

[23] CERN Communications and Outreach Group. FAQ - LHC the guide. Brochure, 2017.

[24] Celine Degrande, Nicolas Greiner, Wolfgang Kilian, Olivier Mattelaer, Harrison Mebane, Tim Stelzer, Scott Willenbrock, and Cen Zhang. Effective field theory: a modern approach to anomalous couplings. *Annals of Physics*, 335:21–32, 2013.

[25] Emergent Garden. Watching neural networks learn. https://www.youtube.com/watch?v=TkwXa7Cvfr8, 2023. Accessed on November 14, 2023.

[26] Bohdan Grzadkowski, M Iskrzyński, Mikolaj Misiak, and Janusz Rosiek. Dimension-six terms in the Standard Model Lagrangian. *Journal of High Energy Physics*, 2010(10):1–18, 2010.

[27] Fengxiang He, Tongliang Liu, and Dacheng Tao. Why resnet works? residuals generalize. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12):5349–5362, 2020.

[28] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[29] S F King. Neutrino mass models. *Reports on Progress in Physics*, 67(2):107, dec 2003.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[31] Andrew Kobach. Baryon number, lepton number, and operator dimension in the standard model. *Physics Letters B*, 758:455–457, 2016.

[32] Marc Lafon and Alexandre Thomas. Understanding the double descent phenomenon in deep learning. *arXiv preprint arXiv:2403.10459*, 2024.

[33] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[34] Jerzy Neyman and Egon Sharpe Pearson. IX. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

[35] Joao Pequenao. Computer generated image of the whole ATLAS detector. 2008.

[36] George Philipp, Dawn Song, and Jaime G. Carbonell. The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions, 2018.

[37] Antonio Pich. Effective field theory. *arXiv preprint hep-ph/9806303*, 1998.

[38] Mark Thomson. *Modern particle physics*. Cambridge University Press, New York, 2013.

[39] Samuel S Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The annals of mathematical statistics*, 9(1):60–62, 1938.