Bachelor's thesis

STUDY OF H→WW→e mu FOR QUANTUM ENTANGLEMENT WITH THE ATLAS DETECTOR AT CERN

Andrii Vak

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: doc. Dr. André Sopczak
May 16, 2025

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Vak Andrii. *STUDY OF H→WW→e mu FOR QUANTUM ENTANGLEMENT WITH THE ATLAS DETECTOR AT CERN.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2025.

First of all, I would like to express my gratitude to my supervisor, doc. Dr. André Sopczak, for his support and great insights during the past year of working on this thesis. Further, I thank my parents for their patience and kindness they gave me throughout this project. I am grateful to my friends, especially to Parkhomenko Illia for his encouragement and Khmelnytskyi Daniil for important remarks on early versions of this work. Finally, I would like to express my gratitude to all those who supported me throughout the years of studying at FIT CTU.

I acknowledge the use of AI-based tools, including OpenAI's ChatGPT and Perplexity AI, for support in refining the structure of the thesis, improving writing clarity, and assisting with technical formulations. All critical analysis, scientific interpretations, and final formulations are my own.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

I declare that I have used AI tools during the preparation and writing of my thesis. I have verified the generated content. I confirm that I am aware that I am fully responsible for the content of the thesis.

In Prague on May 16, 2025

Abstract

This Bachelor thesis adresses the problem of reconstruction of kinematic properties of W bosons in $H \to WW^* \to l\nu l\nu$ decay. The primary focus is to examine the capability of several machine learning algorithms to accurately reconstruct the 4-momenta of the W boson. Using simulated data with the kinematics of the final decay products, this work compares the performance of models such as Random Forest, AdaBoosted Decision Trees and Deep Neural Network using several validation metrics.

The training pipeline is developed using the Python programming language to ease model training and allow performing experiments on architecture and underlying methods. The research finds that a Deep Neural Network method allows for the most precise reconstruction of the W kinematics.

The contribution of this work lies in emphasizing the great potencial of using machine learning methods to reconstruct the W boson kinematics of particle systems, setting a foundation for future improvements in testing the Bell inequalities and quantum tomography.

Keywords Higgs boson, W boson, leptonic decay, machine learning, kinematics reconstruction, deep neural network, Quantum entanglement

Abstrakt

Tato bakalářská práce se zabývá problémem rekonstrukce kinematických vlastností W bosonu v $H \to WW^*. \to l\nu l\nu$ rozpadu. Primárně se zaměřuje na zkoumání schopnosti několika algoritmů strojového učení přesně rekonstruovat 4-moment bosonu W. Na základě simulovaných dat s kinematikou konečných produktů rozpadu tato práce porovnává výkonnost modelů, jako jsou Random Forest, AdaBoosted Decision Trees a Deep Neural Network, pomocí několika validačních metrik.

Trénovací pipeline je vyvinuta pomocí programovacího jazyka Python, aby se usnadnilo trénování modelů a umožnilo provádět experimenty s architekturou a základními metodami. Výzkumem bylo zjištěno, že metoda Deep Neural Network umožňuje přesnou rekonstrukci kinematiky.

Přínos této práce spočívá ve zdůraznění velkého potenciálu využití metod strojového učení k rekonstrukci kinematiky bosonů W částicových systémů, což vytváří základ pro budoucí zlepšení testování Bellových nerovností a kvantové tomografie.

Klíčová slova Higgsův boson, W boson, leptonový rozpad, strojové učení, rekonstrukce kinematiky, deep neural network, Kvantové provázání

Contents

List of Figures

List of Tables

List of abbreviations

ANN	Artificial Neural Network
ATLAS	A Toroidal LHC ApparatuS
AdaBoost	Adaptive Boosting
CERN	Conseil Européen pour la Recherche Nucléaire European Organization for Nuclear Research
CMS	Compact Muon Solenoid
DNN	Deep Neural Network
EDA	Exploratory Data Analysis
FNN	Feedforward Neural Network
LHC	Large Hadron Collider
LR	Learning Rate
ML	Machine Learning
QCD	Quantum Chromodynamics
QE	Quantum Entanglement
RELU	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
SGD	Stochastic Gradient Descent
SM	Standard Model
WBF	Weak Boson Fusion
ggF	gluon-gluon Fusion

Chapter 1

Introduction

Quantum entanglement (QE) is a fascinating phenomenon and is one of the key factors behind the disparity between classical and quantum mechanics. It refers to the situation where a pair of particles is generated so that the individual quantum states of each remain indefinite until measured, and the act of measuring one instantaneously determines the result of measuring the other, even when at a distance from each other [1].

The phenomenon was first described in a paper by Albert Einstein, Boris Podolsky, and Nathan Rosen in 1935 [2], challenging the quantum understanding of the physical world. In their paper, the scientists mentioned above raised a crucial question: Can the quantum-mechanical description of physical reality, as given by the wave function, be considered complete? They argued that it violates two fundamental principles of classical physics — locality and realism. Quantum mechanics allows for correlations, observed in measurements on entangled particles that are spatially separated, but simultaneously suggests that the measurement on one particle can, without a delay, affect the state of another, regardless of distance. Due to this apparent conflict, the quantum mechanics theory described through the wave function was deemed incomplete. Einstein, Podolsky, and Rosen proposed the existence of hidden variables — unknown parameters that determine the state of a system, which the wave function fails to describe fully.

A significant advancement in the discussion of QE came in 1964 when physicist John S. Bell formulated what is now known as Bell's theorem. He formalized the assumptions of the Einstein-Podolsky-Rosen argument into what is called local hidden variable theories — models in which the outcomes of measurements are determined by pre-existing properties (the hidden variables) and no influence can travel faster than the speed of light. These theories directly reflect the classical ideas of locality and realism. Bell then considered an experiment where measurements are performed independently on the two separated particles of an entangled pair. Suppose the measurement results depend upon hidden variables contained within each particle. In that case, the correlation

between two results must obey a specific set of inequalities (Bell inequalities) derived from this experiment's assumptions. John Bell then showed that quantum physics predicts correlations that violate those inequalities, meaning that no theory based on local hidden variables can fully reproduce the statistical correlations predicted by quantum mechanics [3].

1.1 Motivation

QE is used in several modern quantum technologies, including quantum teleportation, superdense coding, quantum cryptography, and quantum computing. These applications exploit the unique correlations between entangled particles to perform tasks that are either impossible or highly inefficient for classical systems. The research on this phenomenon is can be used for development of technologies that society will utilize in the future.

QE has been experimentally observed in different physical systems, ranging from subatomic particles to objects such as diamond crystals. Notably, entanglement was demonstrated between two diamond (each millimeter-sized) samples at room temperature, over a distance of approximately 15 centimeters, highlighting that entanglement is not limited to microscopic systems [4].

However, the study of QE in high-energy particle physics remains severely unexplored. A significant step forward was achieved when the A Toroidal LHC ApparatuS (ATLAS) Collaboration reported the highest-energy observation of entanglement in top-antitop quark events produced in proton-proton collisions at the Large Hadron Collider (LHC) [5].

Beyond the top-quark system, the leptonic decay of the Higgs boson via W bosons offers a particularly intriguing opportunity for testing QE. As described by Alan J. Barr [6], measurements of the emitted lepton directions allow to determine the expectation values of various quantum Bell operators. One can, therefore, go further and use these $H \to WW^*$ boson decays as a laboratory to perform Bell tests [6]. This decay allows for the examination of famous inequalities in a physical arrangement that has never been explored before: at energy scales characteristic of the Higgs boson mass, around 125 GeV; at extremely short time intervals on the order of 10^{-25} seconds, which correspond to the lifetime of the W boson; and at distances as small as 10^{-16} meters.

1.2 Goals

To accurately test Bell inequalities and determine the strength of correlations, one's goal would be to accurately reconstruct the kinematic properties of W bosons in the $H \to WW^* \to \ell\nu\ell\nu$ final state. Those properties will be used to determine the strength of QE in the Higgs and W system, which is related to the violation of Bell inequalities. This goal must be divided into several tasks:

1. Describe the basic principles of Higgs boson production and decay into a pair of W bosons and their subsequent decay into an electron and a muon;
2. Analyze the provided data;
3. Using the features of the electron and the muon, implement and train at least two classical machine learning algorithms to reconstruct the W-boson pair;
4. Develop and optimize deep-learning algorithms (at least one), taking into account that some of the W decay products escape undetected;
5. Evaluate the performance of both classical and deep-learning models on simulated data by comparing the results with the true 4-momenta of the W-bosons;
6. Present the results and discuss the performance of the different machine learning algorithms.

This thesis is organized into six chapters. Chapter 2 describes necessary knowledge of underlying physics, 4-momenta and presents several Higgs boson production modes as well as decays into a pair of W-bosons.Chapter 3 explores models chosen for reconstruction (regression problem) and discusses the methods to evaluate their precision.Chapter 4 give insights into implementation details.Chapter 5 provides analysis of given dataset. The obtained results will be summarized in Chapter 6. The conclusion is given in Chapter 7.

Chapter 2

Standard Model of particle physics

The Standard Model (SM) is a fundamental theoretical framework that describes the electromagnetic, weak, and strong interactions — three of nature's four known fundamental forces, with gravity being the only exception [7]. Alongside these interactions, the SM also classifies and characterizes all known elementary particles that have been experimentally observed to date.

The development of the SM is the result of decades of effort by physicists worldwide, combining theoretical insights and experimental discoveries. One of the earliest milestones was achieved in 1928 with Paul Dirac, who formulated his famous equation, implying the existence of an antimatter, which became a crucial foundation for the SM.

Here, the focus will be on the part about particles. They are described in the SM shown in Figure 2.1. Each particle is characterized by intrinsic properties such as mass, electric charge, spin, and color. Based on their spin, they are divided into two distinct groups: fermions, which possess half-integer spin (specifically, spin-$\frac{1}{2}$), and bosons, which have integer spin (often 1 or 0).

There are 12 fermions, which form the basic building blocks of matter. They are grouped into two families — quarks and leptons — each arranged in three generations of increasing mass [10]. Quarks, which participate in the strong interaction, carry electric and color charges [11]. They are further classified into two types: up-type quarks (up, charm, and top), which carry an electric charge of $+\frac{2}{3}e$, and down-type quarks (down, strange, and bottom), which have an electric charge of $-\frac{1}{3}e$.

The lepton family consists of the electron, muon, and tau, each accompanied by a corresponding neutrino. This association reflects that, in particle interactions, each charged lepton participates exclusively in reactions involving its own neutrino. The electron, muon, and tau have an electric charge of $-1e$, while their neutral counterparts are electrically uncharged. Within the

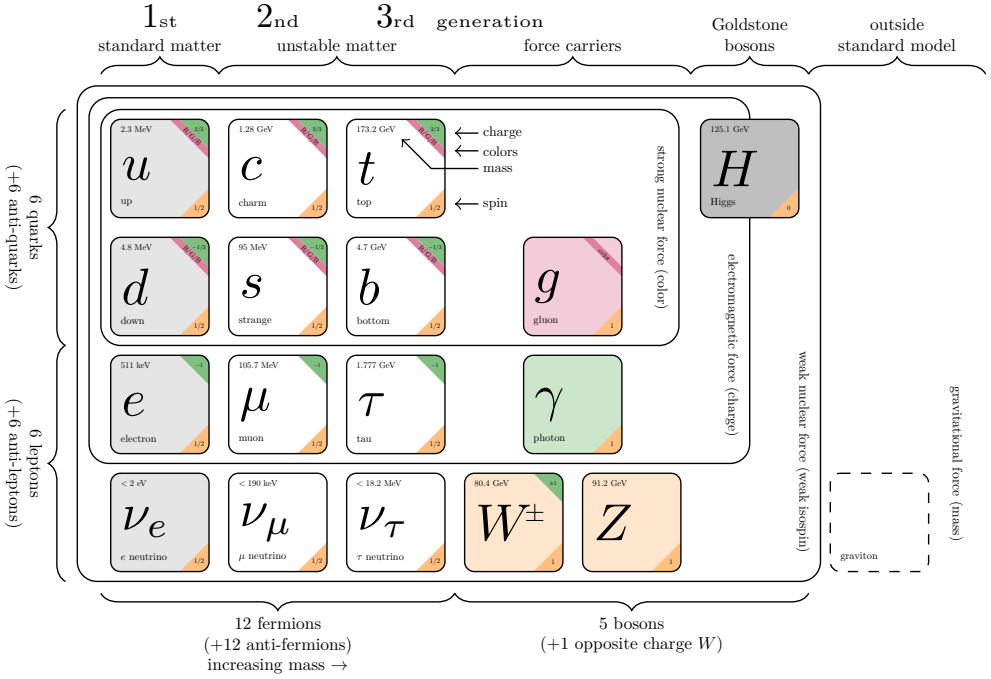

Figure 2.1 Standard Model of Particle Physics [8]. The masses and quantum numbers are taken from the Particle Data Group [9]

SM, neutrinos are assumed to be massless, although experimental evidence suggests they possess a small but nonzero mass.

For every fermion, a corresponding antiparticle with identical mass and opposite electric charge exists. In the case of quarks, the antiparticle also carries the opposite color charge.

The fundamental forces between these particles are mediated by gauge bosons, all of which are spin-1 particles. The massless photon carries the electromagnetic force, the weak interaction is mediated by the massive $W^{\pm}$ and Z^0 bosons, and gluons transmit the strong interaction. In addition, the Higgs boson — a unique scalar particle with spin-0 — plays a crucial role within the SM. This particle arises due to the Higgs mechanism, which introduces the Higgs field and is responsible for electroweak[1] symmetry breaking [12]. This mechanism enables the $W^{\pm}$ and Z^0 bosons to acquire mass via their interactions with the Higgs field [13, 14, 15]. The Higgs boson itself, with a measured mass of approximately 125 GeV [9], was discovered in 2012 by the ATLAS and Compact Muon Solenoid (CMS) experiments at LHC of European Organization for Nuclear Research (CERN) after a search lasting more than four decades [16].

[1] unification of two fundamental interactions: electromagnetic and weak interactions.

2.1 Higgs Boson production

The Higgs boson, like other fundamental particles, can be produced in high-energy particle collisions. In modern experimental physics, this is typically achieved using particle accelerators, where large numbers of particles are accelerated to velocities approaching the speed of light before colliding at extremely high energies. The LHC at CERN is currently the world's most powerful particle accelerator, where protons and lead ions [17] are employed as primary colliding particles.

The Standard Model predicts several mechanisms through which the Higgs boson can be produced in proton-proton collisions. However, the probability (cross section) of creating a Higgs boson in any single event is extremely low — at the LHC, it is estimated that, on average, only one Higgs boson is produced in approximately ten billion collisions.

2.1.1 Gluon-gluon fusion

Gluon-gluon fusion (ggF) is the primary mechanism for Higgs boson production at hadron colliders such as LHC, accounting for approximately 87% of all Higgs bosons created in proton-proton collisions [18]. In this process, two gluons — carriers of the strong nuclear force that 'glue' quarks inside protons — interact to produce the particle. However, since gluons are massless and do not couple directly to the Higgs boson [19], this interaction occurs indirectly through a quantum loop involving virtual heavy quarks, primarily the top quark and, to a lesser extent, the bottom quark. These virtual particles momentarily exist during the interaction, facilitating the gluon-Higgs coupling. The likelihood of such interactions is higher for heavier quarks due to their stronger coupling to the Higgs field. This coupling strength is proportional to the particle's mass, making the top quark's contribution dominant in the ggF process [20, 21]. This mechanism is illustrated as a Feynman diagram in Figure 2.2.

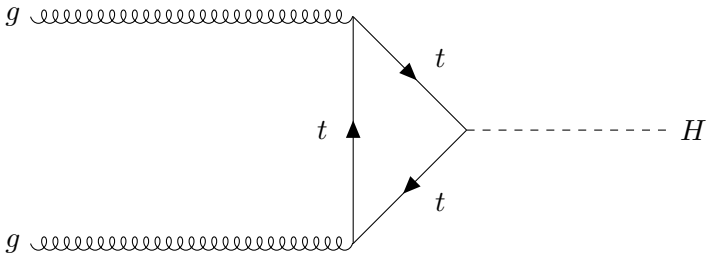

Figure 2.2 Feynman diagram of Higgs boson production via ggF. The gluons couple to a virtual top quark loop, which produces the Higgs boson [21]

2.1.2 Higgs Strahlung process

Higgs Strahlung (German for 'radiation'), also known as associated production, is a process where a Higgs boson is emitted during the collision of a fermion and its corresponding anti-fermion, such as an electron and a positron. Two particles annihilate, and with enough energy, they can briefly form a virtual W or Z boson (carriers of the weak nuclear force), which then 'radiates' a Higgs boson [22]. In the result there is a final state containing both a Higgs and a real Z or W boson [21]. This mechanism dominated Higgs production at older colliders like the Large Electron-Positron Collider (LEP) and was significant at the Tevatron (proton-antiproton collider). However, at the LHC — which collides protons with protons — quark-antiquark pairs are rarer, making Higgs Strahlung responsible for just 4% of Higgs bosons today [18]. A Feynman diagram can be found in Figure 2.3.

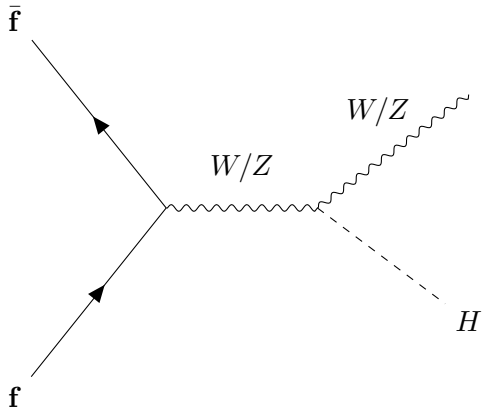

Figure 2.3 Feynman diagram of Higgs boson production via Higgs Strahlung process [21]

Despite its lesser role at the LHC, Higgs Strahlung remains valuable for precision studies. When the Higgs is produced alongside a W or Z boson (written as WH or ZH), scientists can exploit the leptonic decays of the W/Z to isolate clean signals from background noise [23]. Additionally, Higgs Strahlung helps test theoretical predictions: deviations in its production rate could hint at new physics affecting the Higgs' couplings to weak bosons [24]. While overshadowed by gluon fusion (the dominant production mode), Higgs Strahlung acts as a precision tool.

2.1.3 Weak boson fusion

Weak Boson Fusion (WBF) is the second most significant Higgs boson production mechanism at the LHC, contributing approximately 7% of the total Higgs yield, following ggF [18]. In this process, two incoming quarks from the colliding protons emit a virtual weak boson — either a W or Z boson (Figure 2.4).

These bosons then interact to produce a Higgs boson. At the same time, the colliding quarks lose little energy, recoiling smoothly as they exchange weak bosons.

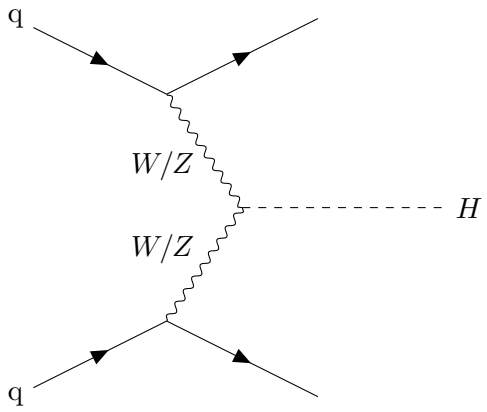

Figure 2.4 Feynman diagram of Higgs boson production via WBF [21]

This results in a distinct experimental signature: two high-energy jets emitted in opposite directions ('tagging jets') in the forward regions of the detector and minimal particle activity in the area between them [25]. This signature makes WBF events relatively easy to identify amidst the complex environment of proton-proton collisions. It is particularly valuable for studying the Higgs boson's couplings to weak bosons due to its clean experimental signature [26, 21].

2.1.4 Top quark fusion

Top quark fusion is an extremely rare Higgs boson production mechanism at the LHC, occurring when two gluons — each splitting into a top quark-antiquark pair — collide and combine to form a Higgs boson (Figure 2.5). This process, sometimes called '$t\bar{t}H$' (though distinct from associated production), relies on the Higgs' strong coupling to top quarks due to their immense mass of about 173 GeV. However, its predicted production rate is about 1% of all Higgs events, making it two orders of magnitude rarer than ggF [18]. The challenge lies in distinguishing it from overwhelming backgrounds like $tt+$ jets, where top pairs are produced alongside unrelated jets.

Despite its rarity, observing top fusion is critical because it provides a direct measurement of the Higgs-top quark coupling. Significant progress was made in the year 2018 when both the ATLAS and CMS collaborations reported the observation of $t\bar{t}H$ production by combining data from multiple Higgs decay channels (e.g., $H \to \gamma\gamma$, $H \to WW^*$): CMS reported a 5.2σ significance for '$t\bar{t}H$' production [27], while ATLAS reached 6.3σ using full 2011–2017 datasets [28]. These results confirmed the Standard Model prediction that the

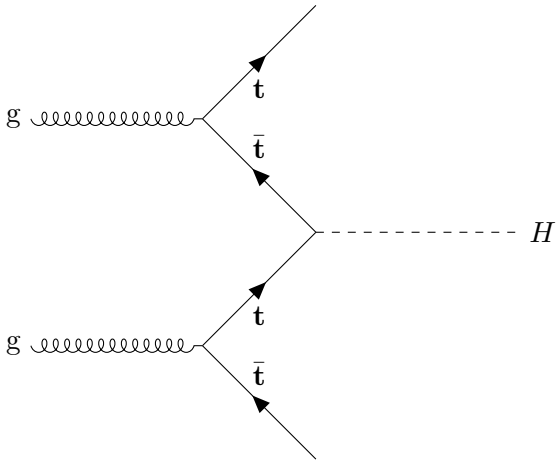

Figure 2.5 Feynman diagram of Higgs boson production via top quark fusion [21]

Higgs boson couple to particles proportionally to their mass.

2.2 Higgs boson decay modes

Quantum mechanics predicts that any unstable particle will eventually decay into lighter particles if such decays are allowed by conservation laws and available phase space. The likelihood of a particular decay occurring depends on factors such as the mass difference between the initial and final states and the strength of the interactions involved. For the Higgs boson, which interacts with all massive elementary particles in the Standard Model (SM) [29], this leads to a variety of possible decay channels, each characterized by a probability known as the branching ratio [30]. The SM provides precise predictions for these ratios as a function of the Higgs mass, and these are commonly visualized in plots showing the expected decay patterns, such as in Figure 2.6 and Figure 2.7. Both plots highlight the dominant decay modes, including fermionic channels such as $H \to b\bar{b}$, $H \to \tau^+\tau^-$, and $H \to c\bar{c}$, as well as bosonic channels like $H \to WW^*$ and $H \to ZZ^*$. Additionally, loop-induced processes such as $H \to gg$, $H \to \gamma\gamma$, and $H \to Z\gamma$ are shown. The plot uses a logarithmic vertical scale to clearly present both frequent and rare decay modes on the same graph. At the observed Higgs mass of 125 GeV, the $b\bar{b}$ final state is the most probable decay channel, followed by decays into WW^*, gg, and $\tau^+\tau^-$. The plot also indicates the total theoretical uncertainty for each decay mode, represented by the shaded bands surrounding each curve.

One common decay path is into a pair of fermion and antifermion. The probability of such a decay is higher for heavier particles, as the coupling strength between the Higgs and a fermion is proportional to the mass of the latter [29]. Thus, while the Higgs could, in principle, decay into a top-antitop quark pair, this is only possible if the Higgs mass exceeds twice the top quark

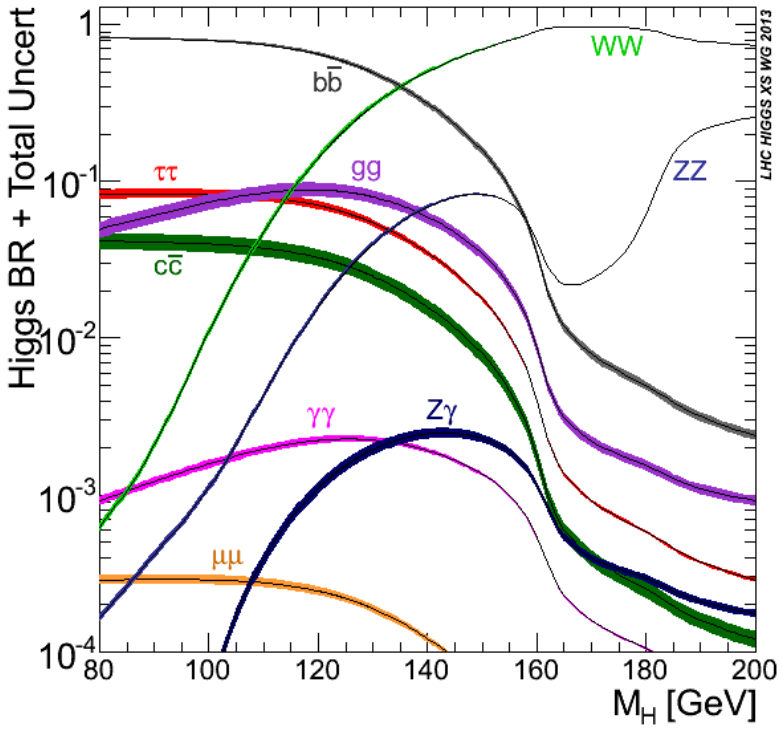

Figure 2.6 Predicted Standard Model branching ratios for the Higgs boson as a function of its mass [31]. The plot covers fermionic decays (such as $b\bar{b}$, $\tau^+\tau^-$, and $c\bar{c}$), bosonic decays ($WW^{(*)}$ and $ZZ^{(*)}$), and loop-induced decays (gg, $\gamma\gamma$, $Z\gamma$). The shaded bands represent the theoretical uncertainties associated with each prediction.

mass (about 346 GeV), which is not the case for the observed Higgs mass of 125 GeV [33]. At this mass, the most common decay is into a pair of bottom and antibottom quarks ($H \to bb$) [34], accounting for approximately 58% of all decays. The next most frequent fermionic decay is into a tau-antitau pair, with a branching ratio of about 6% [18].

The Higgs boson can also decay into pairs of massive gauge bosons. For a Higgs mass of 125 GeV, the decay into a pair of W bosons ($H \to WW^*$) is the most likely bosonic decay, occurring about 22% of the time [18]. Each W boson can further decay into a quark-antiquark pair or a charged lepton and a neutrino, but reconstructing these decays is challenging due to background processes and the undetectable nature of neutrinos [29]. Another important but rarer channel is the decay into a pair of Z bosons ($H \to ZZ^*$), which occurs about 3% of the time at this mass [18]. When both Z bosons decay into easily detectable charged leptons (such as electrons or muons), the resulting signal is particularly clean and useful for Higgs studies.

Decays into gauge bosons as gluons or photons are also possible. For them to happen, quantum loops are needed, involving virtual heavy quarks

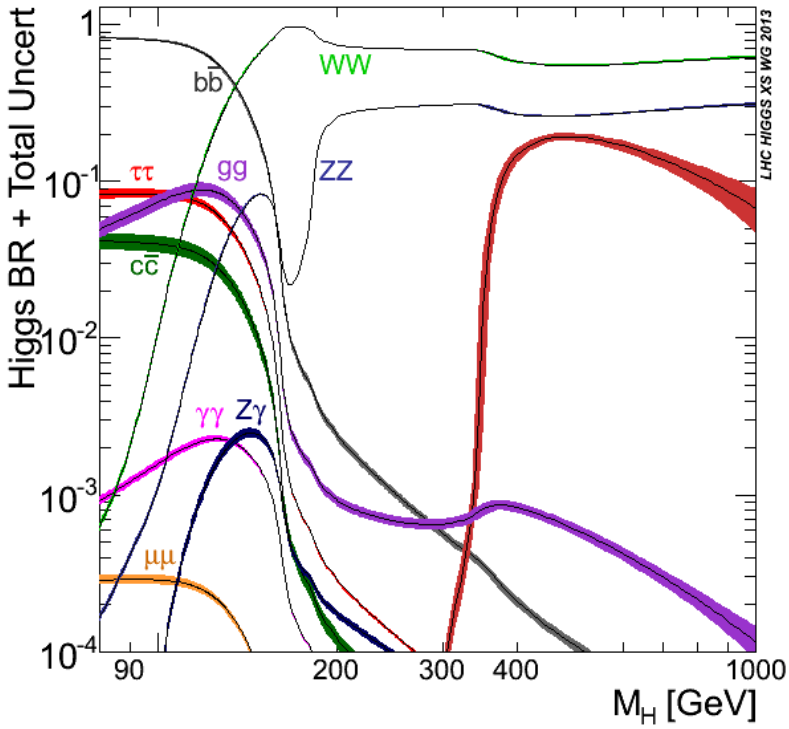

Figure 2.7 Predicted Standard Model branching ratios for the Higgs boson as a function of its mass. Covers wider mass range from 90 GeV to 1000 GeV [32]. The plot covers fermionic decays (such as $b\bar{b}$, $\tau^+\tau^-$, and $c\bar{c}$), bosonic decays ($WW^{(*)}$ and $ZZ^{(*)}$), and loop-induced decays (gg, $\gamma\gamma$, $Z\gamma$). The shaded bands represent the theoretical uncertainties associated with each prediction.

or massive gauge bosons. The decay into two gluons ($H \to gg$) happens about 8.6% [35] of the time, while the much rarer decay into two photons ($H \to \gamma\gamma$) occurs in only about 0.2% of cases [18]. Though rare, the diphoton decay channel is experimentally significant because the photons' energies and directions can be measured with high precision. This allows for an accurate reconstruction of the Higgs boson mass.

2.3 Higgs leptonic decay via W bosons

In this thesis, the focus is placed on the study of the leptonic decay channel of the Higgs boson via W bosons, specifically the process $H \to WW^* \to \ell\nu\ell\nu$, depicted in Figure 2.8. This decay results in a final state consisting of two oppositely charged leptons of different flavors ($l = e, \mu$) and two associated neutrinos.

Two neutrinos in the final state of this decay make the analysis more challenging. They interact only using the weak force and therefore escape direct

detection, leaving no track in the detector. Their presence is inferred indirectly through an imbalance in the transverse momentum of the visible final state particles, known as the missing transverse momentum p_T^{miss} [36].

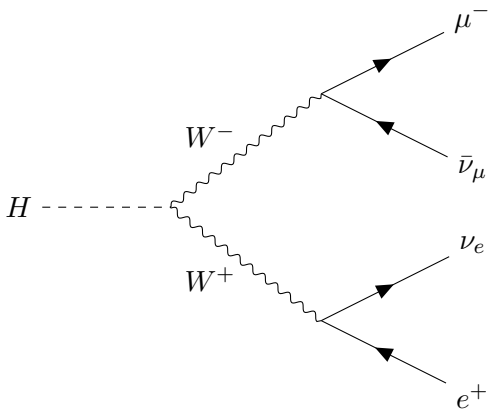

Figure 2.8 Feynman diagram of Higgs boson leptonic decay via W bosons [21]

Chapter 3

Machine Learning

Machine learning (ML) is a field of artificial intelligence whose primary goal is to study and create computational algorithms that are able to learn specific patterns from data and generalize to new information that has never been seen before. This way, they can perform tasks without explicit programming. The term was first used in 1959 by Arthur Samuel, an IBM employee, in his paper [37]. In this work, generalization means a model's ability to yield results as accurately as possible on novel experience (usually called a test set) after being trained on a learning (training) set. The aforementioned data could be in the form of digital sets, gathered and labeled by humans and made available electronically, or other forms of information derived from interaction with the environment. Invariably, the quality of gathered data and its size play a crucial role for the model to make successful predictions [38].

Machine learning proved extremely useful in many fields and its branches, including natural language and speech processing, computer vision, fraud detection, banking, search engines, and others. Given the increasing accessibility of online data and affordable computational resources, more and more companies use ML methods in marketing, healthcare, and even agriculture.

This chapter outlines the essential principles that form the theoretical foundation for the problems and methodologies addressed in the subsequent thesis chapters.

3.1 Learning methods

There are many different scenarios and real-life problems that can be solved (or partially solved) with ML . There are many categories of learning methods based on the type of task, the nature of acquired information, and how the model can learn from the data. Here, two of them will be discussed — supervised and unsupervised learning. Each of them has its own advantages over the others, but there exists no 'Master Algorithm' suitable for each and every

problem. Figure 3.1 is a schematic representation of the first two methods, showing their main differences.

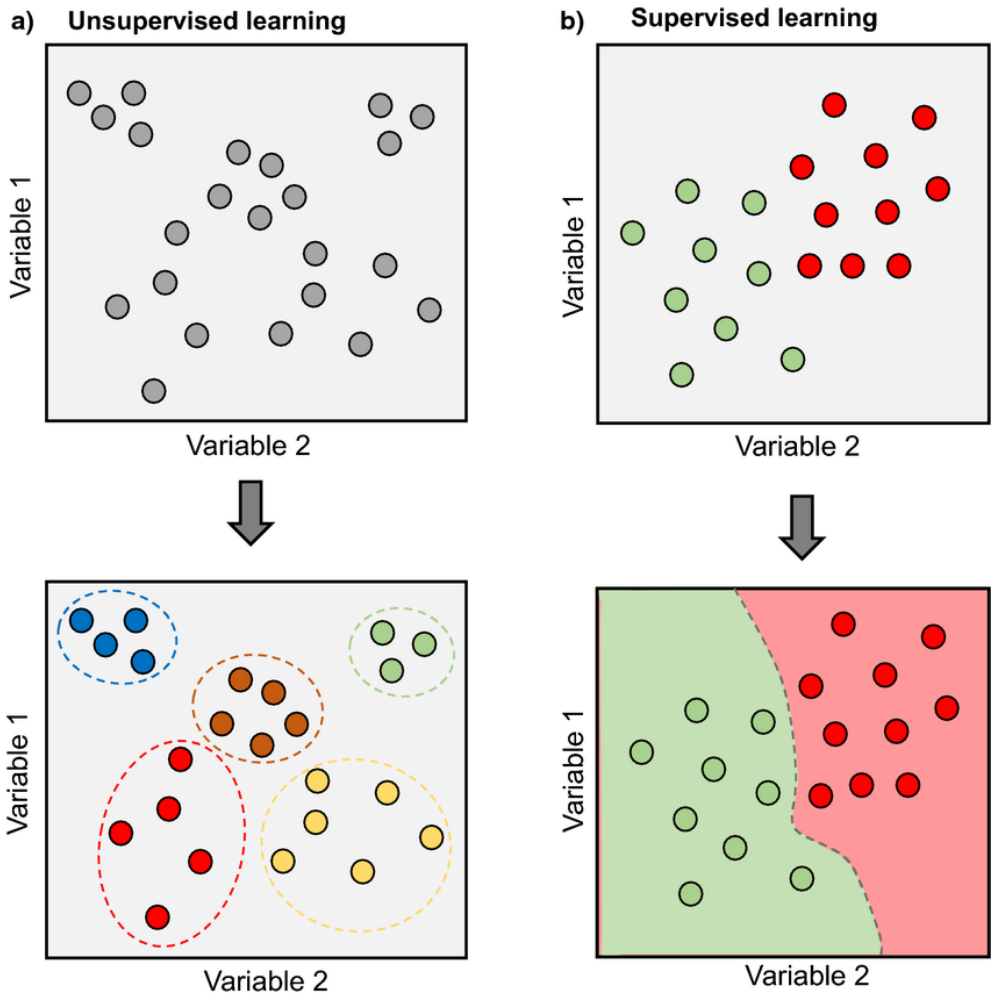

Figure 3.1 Supervised and unsupervised machine learning [39] a) Schematic representation of an unsupervised learning model. Unlabelled data is used in unsupervised learning algorithms for clustering. b) Schematic representation of a supervised learning model. Labeled data is used in supervised learning algorithms for classification.

3.1.1 Supervised learning

In supervised learning, the model is trained on a set of samples that have already been labeled with the correct value. This set is divided into two parts: training and test sets (More on this matter in section Data sets). The learner tries to extract underlying relations in the training set and construct a mapping function from input space to output space [38]. Ideally, a trained model will be able to correctly predict labels not only for samples from the training set

but also for unseen ones — from the test set of entirely new samples from the same problem domain.

Each example is represented by an array or vector called a feature vector. This vector is often dissected into input and output (target) features. The whole data set is represented by a matrix. Labels often come from human input.

Let S be a training data set with N training samples in the form of pairs:

$$S = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \tag{3.1}$$

Where $\mathbf{x}_k$, for $k \in \{1, 2, \ldots, N\}$ is k -th sample (input) vector drawn from m -dimensional input space X^m and $\mathbf{y}_k$ is k -th label (output) vector from n -dimensional output space Y^n. We assume that examples are independently and identically distributed (i.i.d.) according to some fixed but unknown distribution D, where $S \in D$. The model experiences training dataset, learns correct labels for given samples, and constructs a mapping function $f : X^m \to Y^n$ to predict output vector based on feature vector [40]. Then, the performance is evaluated using special metrics.

Supervised ML can be further grouped into two categories based on the type of output label to predict. The first one is classification. In this scenario, each component y_i of the label vector $\mathbf{y} \in Y^n$, for $i \in \{1, 2, \ldots, n\}$, can take only one value from a predefined set of discrete classes. The task then is for the model to correctly assign a label for each input. For example, in the email-filtering problem, it is possible to define a task to filter spam email. In this case, the discrete classes for each sample (email) will be {`spam, non-spam`}. The classification itself is typically performed by estimating the likelihood that the observed sample belongs to each possible class, with the class of highest estimated probability selected as the predicted outcome.

The second one is regression. Here, label vector $\mathbf{y}$ is drawn from n-dimensional real space R^n. Now that the labels are real numbers, one cannot reasonably expect the model to predict the correct label with brilliant precision when it is unique or even exactly its average value. Instead, it is appropriate to demand that the predictions be as close to the correct ones as possible. This is the key difference between regression and classification: the measure of error is quantified by the magnitude of the difference between the predicted and true real-valued labels rather than by their exact equality. In this work, a regression task is performed with multiple outputs required (here, the label vector $\mathbf{y}$ will be taken from an 8-dimensional real space R^8).

3.1.2 Unsupervised learning

In unsupervised learning, algorithms analyze datasets that lack explicit, correct labels. Unlike supervised learning, where the goal is to predict known outcomes, unsupervised learning seeks to uncover hidden structures or relationships within the data itself. More formally, the training data set consists

only of N input vectors:

$$\{(\mathbf{x}_1), (\mathbf{x}_2), \ldots, (\mathbf{x}_N)\}$$

where $\mathbf{x}_k$, for $k \in \{1, 2, \ldots, N\}$ is k -th input vector drawn from m -dimensional input space X^m. The main goal of the model now is to learn patterns from this data and find its own solution, without supervision in the form of labels and evaluation metrics used in supervised learning. This method is used in clustering (e.g., k-means, DBSCAN, hierarchical clustering), density estimation, anomaly detection, and dimensionality reduction (PCA, t-SNE, autoencoders).

Unsupervised models typically learn parameters through optimization of similarity metrics, which somehow explain the underlying structure of the data. For example, clustering algorithms adjust parameters to group similar data points together [41], while dimensionality reduction methods try to find transformations that preserve the essential structure of the dataset in a lower-dimensional space with the best possible accuracy. Different techniques make different assumptions about the data structure. Compactness and connectedness measure how close members of the same cluster are, while separation quantifies how much different clusters are separated [42]. Methods such as the Dunn index combine these aspects, evaluating the ratio between minimum inter-cluster separation and maximum intra-cluster diameter [43]. Other approaches, like density-based clustering, identify clusters as regions of high data density [41].

Unlike supervised learning, there is no universal metric for evaluating the performance of unsupervised models because there are no labels for comparison.

3.2 Data sets

The foundation of any machine learning project is the data. High-quality, diverse, and large datasets are essential for building models that generalize well to new, unseen examples. Data can originate from various sources, including experimental measurements (e.g., detector data), computer simulations, or even artificial datasets crafted by humans.

Preprocessing is important too. Raw data must be thoroughly cleaned, which includes getting rid of errors or missing data, adding new features or removing unnecessary ones. The performance and reliability of a ML model are directly linked to the dataset it experiences. Neglecting this step means leading to reduced model accuracy, increased bias, or misleading results.

One of the critical aspects of preparing data is the train-test split. As mentioned earlier, the models have to correctly predict labels not only for the training data but also for samples they have never encountered before. To understand how good they are, it is important to evaluate each model's performance on a subset of data not used during training. Typically, the dataset

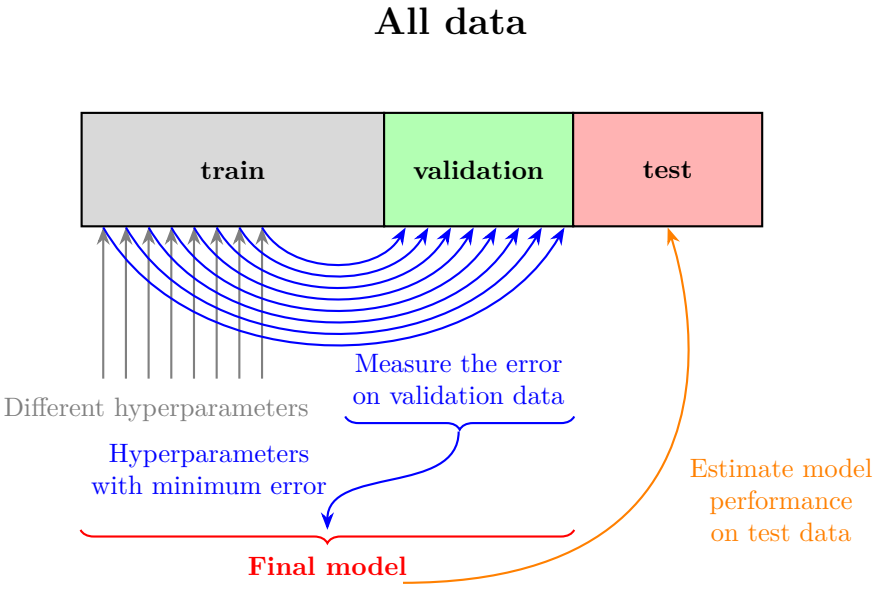

Figure 3.2 How different subsets are used in a training process [44]

is divided into two or three parts: training set (used for the training phase), validation set, and test set.

To further prevent data leakage — where information from the test set inadvertently influences the model and leads to overly optimistic performance estimates — it is important that all preprocessing steps are performed using only the training data, with the test data remaining untouched. Data leakage can arise from improper data handling, such as applying normalization or feature selection across the entire dataset before splitting.

The test set is kept aside until the very end of the process when the final model has been chosen. It is only used to get an estimation of ML algorithm performance and generalization ability on the unseen before data — how good it will be when used in the future — and is never considered in the process of choosing the final model. Otherwise, the resulting test error will be a too-optimistic estimate of reality because the algorithm has been fitted to this data, leading to a distorted and biased choice. It also violates the principle of not touching the test data before obtaining the final model, which is necessary for the test error to be a reasonable estimate of the true error.

All algorithms have internal parameters, which are determined while the model experiences training data, and external parameters, which control the learning algorithm but are not learned from the data itself. Those settings are established before the training begins. They are often called hyperparameters. To determine the optimal set of hyperparameters — to tune them — a model is trained on the training set with different values of those settings and then evaluated on the validation set. Then, based on the values of the metrics, we choose our final model and finally get a realistic estimation of how the model

will perform on future data. Figure 3.2 shows how different sets are used in the training process.

3.3 Evaluation metrics

In a multi-output regression problem with N samples and D output targets (in this case, $D = 8$), the following error metrics are defined by aggregating across all targets. Each metric below is computed by summing or averaging the per-target errors over all N samples. This section describes the evaluation metrics used to calculate the performance of regression models. The overfitting and underfitting processes are described as well.

3.3.1 Mean Squared Error

The **Mean Squared Error** (MSE) measures the average squared difference between the estimated and true value. MSE is commonly used as a training loss for neural networks and is also the default impurity measure in regression trees. For multi-output regression, it is defined as:

$$\text{MSE} = \frac{1}{N}\frac{1}{D}\sum_{i=1}^{N}\sum_{j=1}^{D}(y_{ij} - \hat{y}_{ij})^2\,, \tag{3.2}$$

where y_{ij} is the true value and $\hat{y}_{ij}$ the predicted value of the j-th target feature for sample i.

Advantages:

- Penalizes larger errors more heavily due to squaring, which leads models to focus on reducing big mistakes;
- Differentiable and convex, making it suitable for optimization algorithms like gradient descent.

Disadvantages:

- Sensitive to outliers and large deviations because squaring amplifies large errors;
- The units of MSE are the square of the original units, which can be harder to interpret;
- With multiple targets, if one target has a larger scale, it can disproportionately influence the overall MSE unless targets are normalized.

3.3.2 Mean Absolute Error

The **Mean Absolute Error** (MAE) calculates the average of the absolute differences between predicted values and actual values. It is given by:

$$\text{MAE} = \frac{1}{N}\frac{1}{D}\sum_{i=1}^{N}\sum_{j=1}^{D}|y_{ij} - \hat{y}_{ij}|\,. \tag{3.3}$$

Advantages:

- Unlike MSE, MAE treats all errors linearly, which directly corresponds to the average magnitude of errors;
- Less sensitive to outliers compared to MSE since it does not square the errors. Large deviations do not disproportionately affect the metric;
- Units are the same as the original data, making it more interpretable;
- Can be more informative when typical absolute errors are in priority rather than worst-case errors.

Disadvantages:

- Not differentiable at zero, which can complicate optimization in some algorithms.
- Does not penalize large errors as much as MSE, so models that occasionally make large mistakes (underfitting models) might score better under MAE than under MSE, which can mask tail issues.

3.3.3 Root Mean Squared Error

The **Root Mean Squared Error** (RMSE) is the square root of the MSE, bringing the error metric back to the original units of the target variable. RMSE still penalizes significant errors (due to the underlying squaring), but its scale is comparable to the original data. It is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\frac{1}{D}\sum_{i=1}^{N}\sum_{j=1}^{D}(y_{ij} - \hat{y}_{ij})^2}. \tag{3.4}$$

Advantages:

- Like MSE, it penalizes larger errors more strongly, but its units are interpretable since they match the original data;
- Expressed in the same units as the targets, making error magnitudes directly interpretable (unlike MSE);

- Widely used and understood in many fields.

Disadvantages:

- Still susceptible to outliers due to the squaring of errors;
- Provides no fundamentally new information beyond MSE .

3.3.4 Overfitting and Underfitting

Overfitting is the phenomenon when the algorithm fails to capture the underlying structure of the data and instead memorizes the dataset, including its random noise, outliers, measurement errors, and anomalies. The result is a mapping function that is perfectly fitted to training samples. An overfitted model is excessively complex, having more parameters than available data. It achieves high accuracy if measured on the training set, but performs inadequately on the validation set. Predictions on additional data will be very inaccurate because of the lack of generalization capabilities — the model has learned too much unnecessary information [45]. To mitigate overfitting, one can make use of regularization techniques, increase the size of the training set, or simplify the model in question [46].

On the other hand, underfitting takes place when the model is too simple. It fails to capture significant patterns adequately and thus has low accuracy on training and validation sets. This process is a direct consequence of applying too much regularization, using low-quality sample sets, or training with a small number of epochs (for Neural Networks) [45]. The remedy will be the opposite of overfitting — one can make the model more sophisticated by adding more trainable parameters, try using different values of hyperparameters, or weaken the regularization effect.

Figure 3.3 shows an intuitive example of overfitting and underfitting.

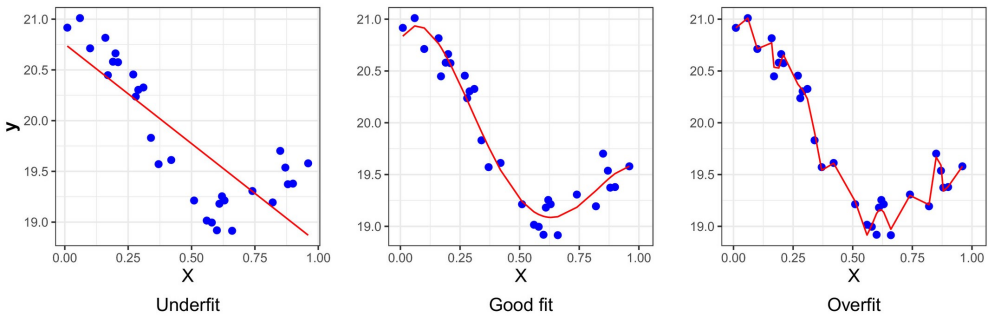

Figure 3.3 Illustration of the underfitting/overfitting issue on a simple regression case. Data points are shown as blue dots, and the model fits as red lines. Underfitting occurs on the left panel, a good fit is shown in the center panel, and overfitting in the right panel [47]

3.4 Hyperparameter optimization

Previously, it was mentioned that the validation set is used to tune the model's hyperparameters. This process is called hyperparameter optimization. It involves searching for the best combination of hyperparameter values that minimize a chosen evaluation metric, typically measured on a validation set. The search spaces are often large and complex. That is why there are a lot of different exploration strategies available — from random search to evolutionary algorithms.

In this thesis, a commonly used method was applied — a grid search. It creates a discrete set of candidate values for each hyperparameter, which forms a multidimensional grid [48]. The algorithm exhaustively evaluates the model performance at every point in this grid. The optimal hyperparameter set is the one yielding the best validation performance.

This concept is simple and easy to implement. However, it suffers from a problem similar to the 'curse of dimensionality': the number of evaluations grows exponentially with the number of hyperparameters in the grid. This makes grid search computationally expensive and often impractical for models with many hyperparameters or costly training procedures [49].

3.5 Ensemble method

Each trained ML model has a unique approach to the problem before it, constructing its own mathematical representation to find a solution. However, it is a good idea to combine predictions from multiple distinct models simultaneously. This way, the variance in estimation is reduced, which might lead to better accuracy and robustness compared to relying on a single model alone [50].

The ensemble method allows to train several ML algorithms independently on the same task and then aggregate all predictions to form a final output. This ensemble consists of a concrete set of alternative models, often called 'weak learners'. They can be constructed using one ML algorithm or several separate ones. Those learners can have different hyperparameter values or be trained on distinct subsets of the training set. Regardless of their construction, they all focus on the same type of task — either classification or regression. In the second case, the final solution can be calculated by averaging the outputs of all learners [51].

There are two main categories of ensemble techniques: parallel and sequential. Parallel methods, such as bagging (bootstrap aggregating), train weak learners independently on different random subsets of the training set and then combine their predictions. Random forest is one of the most popular method of bagging-based ensembles. Sequential methods, like boosting, train base learners iteratively, where each new model is focused on correcting the errors of its predecessors. This work uses one such method — AdaBoost [51].

3.6 Supervised learning approaches

3.6.1 Decision tree

A decision tree is the supervised learning algorithm used to predict output value based on the given sample by learning simple decision rules inferred from the training data features. A regression tree is used if the target is a real number. The main idea is to iteratively split training data samples into distinct and non-overlapping regions, where the output values within each region are as homogeneous as possible. The splits are executed inside each internal node based on the value of a chosen feature (e.g., whether a feature value is greater than a threshold) [52]. The goal is to create two sub-samples, or 'children', with greater purity of the output variable than their 'parent'. For regression tasks, purity means the first child should have samples with high target variable values, and the second should have samples with low values. Each leaf node corresponds to a predicted target value. For example, in CART, one of the decision tree construction algorithms, the final prediction is the average of the target values of training samples falling into the region [52]. Figure 3.4 illustrates a trained decision tree for regression task.

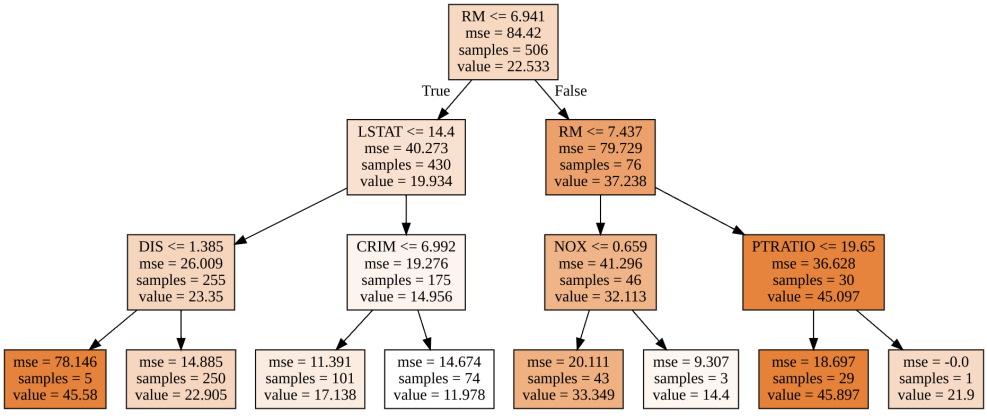

Figure 3.4 Visualization of a regression decision tree trained on the Boston Housing dataset. The tree splits the data based on various features (e.g., RM, LSTAT, CRIM) to predict housing prices. Each node displays the mean squared error, number of samples, and predicted value. Darker colors represent higher predicted values [53]

Building a regression tree involves a top-down, greedy algorithm called recursive binary splitting. Starting with the initial training set and first internal node, the algorithm constructs a rule to split samples into two regions based on one feature only. Let S be a training data set with N training samples in the form of pairs:

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \tag{3.5}$$

where $\mathbf{x}_k$, for $k \in \{1, 2, \ldots, N\}$ is k -th sample (input) vector drawn from

m -dimensional input space X^m and y_k is k -th label (output) value from 1-dimensional output space Y. Then, let x_k^i be an i-th feature of the k-th input sample. If x_k^i is numerical, a rule is defined as:

$$x_k^i \leq t, \tag{3.6}$$

where t is some numerical threshold value. Now, let N_t be the tree's node with index t, then C_t^L C_t^R are the child nodes of N_t, $j \in N_t$ is one the training samples that pass through the node and $\bar{y}_t$ is the mean value of those samples. The loss function — residual sum of squares (RSS) — for N_t is defined as:

$$RSS_t = \sum_{j \in N_t} (y_j - \bar{y}_t)^2. \tag{3.7}$$

The loss function for the whole tree is the RSS across all leaves or across all nodes, which will be split while training the model (called 'buds'). If I_t is the indicator of whether the node t is a bud or a leaf, the total loss for the tree is:

$$RSS_T = \sum_t I_t \cdot RSS_t \tag{3.8}$$

When choosing a feature and threshold to perform a split, the algorithms try to minimize RSS_T. The reduction in RSS_T when splitting bud N_t into children C_t^L and C_t^R is defined as:

$$\begin{aligned} \Delta RSS_T &= RSS_t - \left(RSS_{C_t^L} + RSS_{C_t^R}\right) = \\ &= \sum_{j \in N_t} (y_n - \bar{y}_t)^2 - \left(\sum_{j \in C_t^L} \left(y_n - \bar{y}_t^L\right)^2 + \sum_{j \in C_t^R} \left(y_n - \bar{y}_t^R\right)^2 \right), \end{aligned} \tag{3.9}$$

where $\bar{y}_t^L$ and $\bar{y}_t^R$ is the mean value of samples in the child nodes.

To find an optimal sequence of splits to minimize RSS_T is an NP-hard problem [54]. Instead, The algorithm takes a greedy approach: for each split, it considers all possible combinations of features and thresholds. The values are taken from training data — each value of the chosen feature among the samples in the bud is tested as the threshold. After considering all possible splits, we choose the one with the greatest reduction in bud's RSS .

Advantages:

- Regression trees are simple to understand and interpret. The decision rules can be expressed in if-then statements;
- Naturally capture nonlinear dependencies and interactions between features;
- Requires little data preparation. Some tree and algorithm combinations support missing values [55];

- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree [55];
- Able to handle both numerical and categorical data.
- Able to handle multi-output problems. The modification is simple: Store n output values in leaves instead of one and use splitting criteria that compute the average reduction across all n outputs [55].

Disadvantages:

- The algorithm can create very deep and complex trees, resulting in overfitting and poor generalization capabilities. To avoid this, one can use methods as pruning, set the minimum number of samples required at a leaf node or limit the maximum depth;
- Decision trees can significantly change their structure because of small variations in the training data;
- Predictions of decision trees are piecewise constant approximations;
- Features with many distinct values can dominate the splitting process, potentially biasing the model.

3.6.2 Bootstrap aggregated trees

Boosted trees are an ensemble learning technique that constructs a strong predictive model by sequentially combining multiple weak learners (e.g., shallow decision trees). Each succeeding tree is trained to correct the errors of the previous ensemble, progressively minimizing a predefined loss function. To force models to focus on the 'hard' examples, the boosting algorithm applies weights to the training data and changes them after each iteration accordingly [56]. For regression tasks, the objective is often to reduce MSE or MAE .

Boosting often yields significantly improved accuracy over a single model, and in practice, its ensembles tend to have lower errors than bagging-based ensembles in many settings. In regression tasks, boosting creates a sequence of regression trees, each trying to correct the residual errors of the ensemble so far. Empirical studies show that a boosted regression model can achieve lower prediction error than an individual tree or a simple bagged ensemble [57].

AdaBoost (Adaptive Boosting) is one of the most famous boosting algorithms, introduced by Freund and Schapire [58]. The key idea in AdaBoost is the adaptive reweighting of training examples based on error. In regression, one must convert continuous prediction errors into a form suitable for reweighting. In 1997, Leo Drucker proposed AdaBoost.R2 [59], an adaptation of AdaBoost to regression problems. His modification introduces a weighted median when aggregating predictions of all weak learners.

Let S be a training data set with N training samples in the form of pairs:

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \tag{3.10}$$

where $\mathbf{x}_k$, for $k \in \{1, 2, \ldots, N\}$ is k -th sample (input) vector drawn from m -dimensional input space X^m and y_k is k -th label (output) value from 1 -dimensional output space Y. Let T be the number of weak learners to train.

The following AdaBoost.R2 algorithm was proposed by Drucker [59]:

1. To each training sample $\mathbf{x}_k$ assign a weight w_k for $k \in \{1, 2, \ldots N\}$;
2. Repeat next steps for $t \in \{1, 2, \ldots T\}$ or while $\hat{L}^t$ loss defined below is less than or equal to 0.5:
   - Draw a N samples with replacement and with probability $p_k = \frac{w_k}{\sum_{j=1}^{N}}$ or $k \in \{1, 2, \ldots N\}$ to form new training subset;
   - Train weak learner t to the bootstrapped training subset and predict values for all samples from the original training set. Let those predictions be $f^t(\mathbf{x}_k)$ for $k \in \{1, 2, \ldots, N\}$;
   - Calculate a loss ${L^t}_k$ for $k \in \{1, 2, \ldots, N\}$:

$$D = \sup \left| y_k - f^t(\mathbf{x}_k) \right| \tag{3.11}$$

$${L^t}_k = \frac{\left| y_k - f^t(\mathbf{x}_k) \right|}{D} \tag{3.12}$$

   - Calculate the average loss $\hat{L}^t$:

$$\hat{L}^t = \sum_{k=1}^{N} {L^t}_k \cdot p_k \tag{3.13}$$

   - If $\hat{L}^t \geq 0.5$, end iteration and set $T = t - 1$;
   - Let $\beta_t = \frac{\hat{L}^t}{1 - \hat{L}^t}$ be a measure of confidence in the weak learner. A low value means high confidence in the prediction;
   - Update the weights: $w_k = w_k \cdot (\beta_t)^{1 - {L^t}_k}$. The smaller the loss, the smaller the weight. This reduces the probability of picking this sample to the training subset for the next iteration.
3. To get final output value for sample $\mathbf{x}$, calculate the weighted median of $f^t\mathbf{x}$ for $t \in \{1, 2, \ldots T\}$, using weights $\log(\frac{1}{\beta_t})$ for model t.

Advantages:

- AdaBoost often achieves superior accuracy compared to single models by combining multiple weak learners into a strong ensemble. By sequentially focusing on the samples that are hardest to predict, it adaptively improves the overall model's performance, often outperforming many traditional classifiers and regressors;

- AdaBoost can work with various weak learners, such as decision trees or linear models. One can apply this algorithm across different types of data and problem domains, adapting to the strengths of the chosen base learner;
- Despite its focus on difficult cases, AdaBoost is often resistant to overfitting, especially when using simple weak learners and regularization. This robustness stems from combining many weak models rather than relying on a single complex one, thus controlling variance.

Disadvantages:

- AdaBoost's adaptive weighting scheme can cause it to excessively emphasize noisy or mislabeled samples, as these are repeatedly misclassified and thus receive higher weights. This can degrade model performance and lead to overfitting on noise;
- Because AdaBoost trains weak learners sequentially, it cannot easily be parallelized. This sequential nature increases training time, especially for large datasets or when many iterations are required. The computational cost can become significant compared to methods like random forests;
- AdaBoost requires that each weak learner performs slightly better than random guessing. If weak learners are unstable or too weak, the ensemble may fail to converge to a strong model, resulting in suboptimal performance.

3.7 Artificial Neural Networks

Artificial neural networks (ANNs) are inspired by the structure and function of biological neural networks found in the human brain. In biological systems, neurons communicate through synapses, forming complex networks that process and transmit information via electrical impulses. ANN abstracts these principles by representing artificial neurons as mathematical functions that receive inputs, apply weights, and produce outputs through activation functions. The learning process in an ANN involves adjusting the weights so that the network can approximate complex functions or recognize patterns in data.

The conceptual foundation for ANNs was laid in 1943 when Warren McCulloch and Walter Pitts published a paper introducing the first mathematical model of a neural network [60]. Their model described how simple computational units could be interconnected to perform logical operations, suggesting that networks of such units could, in principle, compute any function that a digital computer could.

The next significant milestone came in the late 1950s when Frank Rosenblatt introduced the perceptron, an early trainable neural network model that could perform binary classification tasks [61]. The perceptron consisted of

a single layer of artificial neurons, each computing a weighted sum of its inputs and passing the result through a threshold activation function.

Neural networks continue to evolve, with applications in healthcare, finance, autonomous vehicles, and scientific discovery.

3.7.1 Feedforward Neural Network

Feedforward Neural Networks (FNNs) are a foundational class of ANNs characterized by their plain data flow. The information propagates from input to output layers without cyclic connections. These networks typically consist of one input layer, one or several hidden layers, and an output layer. The FFN with multiple hidden layers is called a Deep Neural Network (DNN) [62]. Each neuron in a layer computes a weighted sum of inputs from the preceding layer, followed by a nonlinear activation function. Mathematically, for layer i, and neuron j, the output is represented by function ${f_j}^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}$, where n_{i-1} is the number neurons from previous layer. This function is calculated as:

$$ {f_j}^{(i)}(\mathbf{x}) = g(\mathbf{w}^T \cdot \mathbf{x} + w_0), \tag{3.14} $$

where g is activation function and $\mathbf{w}$ is the weight vector of neuron j. Layer i of the FNN can be represented as $\mathbf{f}^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$, where

$$ \mathbf{f}^{(i)} = \left({\mathbf{f_1}}^{(i)}, \mathbf{f_2}^{(i)}, \ldots, {\mathbf{f_{n_i}}}^{(i)} \right)^T \tag{3.15} $$

.

The whole FNN is a function chain $\mathbf{f}^{(l)} \odot \mathbf{f}^{(l-1)} \odot \cdots \odot \mathbf{f}^{(2)} \odot \mathbf{f}^{(1)}$. Figure 3.5 shows an example of FNN . This type of architecture allows FNNs to approximate complex nonlinear mappings. Training process is using backpropagation and gradient-based optimization to minimize a loss function. Gradients are computed via the chain rule, iteratively adjusting weights and biases to reduce prediction errors.

3.7.2 Regularization

Due to the immense amount of parameters in DNNs, they are prone to overfitting. While reducing model complexity (e.g., fewer layers or neurons) can mitigate overfitting, it risks underfitting by simplifying the model. Regularization techniques address this trade-off by constraining the learning process without discarding model capacity. In this thesis, experiments employed L1 and L2 regularization, which penalize large weights by adding a term to the loss function. L1 regularization (Lasso) drives non-critical weights to zero, while L2 (Ridge) shrinks weights uniformly. Dropout and Batch Normalization were also considered in different architectures. An additional technique was used — Gaussian noise injections. It injects random noise sampled from a normal

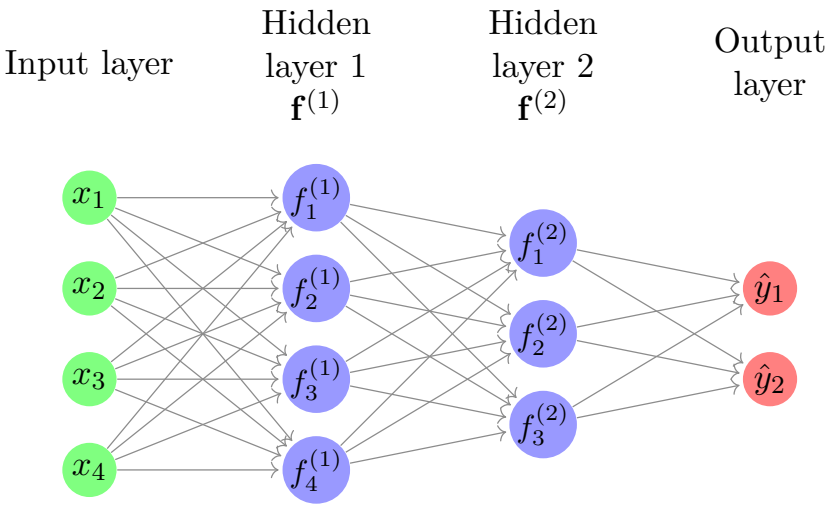

Figure 3.5 An example of FNN with 3 layers. This model is considered as DNN .

distribution $N(0, \sigma^2)$ into hidden layer activations [63]. By corrupting inputs with noise, the model learns robust features invariant to small changes [64].

3.7.3 Learning rate scheduler

The learning rate (LR) is a critical hyperparameter in training deep neural networks, controlling the step size at which model parameters are updated during optimization. Choosing an appropriate LR is essential: too small a value leads to slow convergence and longer training times, often causing the model to get stuck in flat regions of the loss function — a vanishing gradient problem. On the contrary, too large LR can cause unstable updates, resulting in oscillations or divergence, known as the exploding gradient problem. To address these challenges, dynamic learning rate schedules adjust the LR during training to balance convergence speed and stability.

In this work, two learning rate scheduling strategies were experimented with: the multi-step schedule with burn-in and the cosine annealing schedule. The multi-step learning rate schedule reduces the LR at predefined epochs or iterations, allowing the model to make significant initial updates and gradually refine its parameters as training progresses [65]. The burn-in period is a particularly effective enhancement: it starts training with an intentionally low LR that gradually increases during the initial phase (the burn-in steps). This gradual warm-up helps prevent premature convergence to suboptimal value and stabilizes early training dynamics, especially when using large batch sizes or complex architectures [65]. After the burn-in phase, the LR decreases stepwise at specified milestones.

The cosine annealing schedule, introduced by Ilya Loshchilov and Frank Hutter, is a smooth, non-monotonic LR decay strategy inspired by simulated

annealing [66]. The LR at iteration t is computed as:

$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)\ \eta_0, \tag{3.16}$$

where η_0 is the initial LR, T is the total number of training iterations (sometimes epochs) and t is the current one. This schedule gradually decreases the LR from η_0 to zero following a half-cosine curve, allowing the optimizer to explore the parameter space freely at the start and then converge smoothly.

Chapter 4

Implementation

This chapter describes the design and implementation of the training pipeline, which was developed for this research. The main objective behind this pipeline was to provide a modular and reusable environment capable of handling experiments with multiple DNN models in an organized and efficient way. The main priority was given to maintainability, flexibility, and the ability to visualize results to interpret and evaluate model performance in the context of quantum entanglement studies.

At the core of this system is the concept of configuration files. Each model is represented by its own configuration file, which specifies parameters such as:

- The dataset to be used;
- Model architecture (layers, activation functions);
- Training parameters (number of training, validation, and test epochs, optimizer, learning rate, batch size).

This approach makes it very simple to switch between different models, reproduce results, or make minor adjustments during experimentation without altering the code. It also facilitates using the same pipeline for future projects or extended analyses.

Once a training session is initiated, the pipeline performs the following steps:

1. Reads the global configuration for environment settings (paths to datasets, output folders, logging preferences);
2. Loads the dataset from disk and preprocesses it into train, validation, and test subsets;
3. Instantiates a deep learning model based on the selected architecture from the configuration file;

4. Trains the model, logging its progress (loss values, validation metrics) and saving intermediate checkpoints;

5. Evaluation plots are generated, comparing reconstructed quantities to the truth-level data;

6. After training is complete, the model is used to predict the reconstructed 4-momenta of the particles in the test set;

7. Models are saved for later reuse, allowing inference on new data without retraining.

This flow can be represented as the diagram in Figure 4.1.

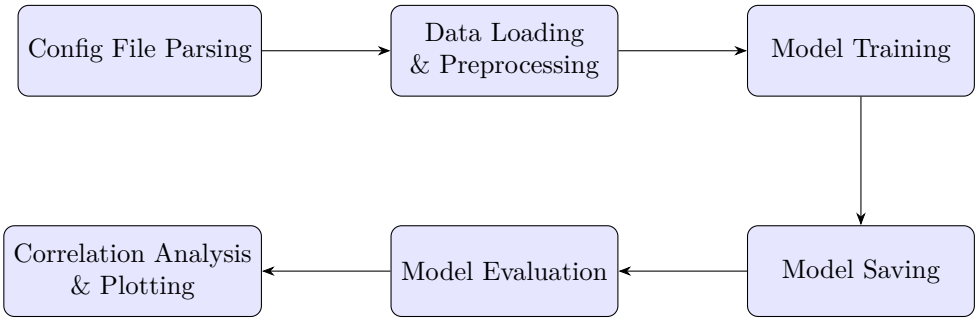

Figure 4.1 Overview of the machine learning pipeline for $H \to WW^* \to \ell\nu\ell\nu$ analysis. The process begins with configuration parsing, continues through data loading, model training, and evaluation, and ends with the generation of physical analysis plots for correlation interpretation.

The pipeline was designed to be both easy to maintain and extendable. Its modular structure allows for new model architectures or data types to be integrated with minimal effort, which is especially valuable for future research or similar experiments.

An important part of the implementation is the set of tools for result analysis. The pipeline can automatically generate plots of Wigner-Weil coefficients c_{ij}, to study how the reconstructed four-momenta affect the quantum correlations between W bosons. Three main types of plots are generated:

- A 3D histogram (visualized as a heatmap) of the c_{ij} coefficients computed from the reconstructed four-momenta;
- The same type of heatmap, but calculated from the truth-level data;
- An array of 8×8 two-dimensional histograms, each corresponding to a specific (i, j) pair of coefficients. These plots compare the reconstructed c_{ij} distributions to their truth counterparts, offering a detailed view of reconstruction accuracy.

In addition to the c_{ij} analysis, the pipeline can produce comprehensive visual comparisons of reconstructed four-momenta versus ground-truth values. These are presented as:

- Two-dimensional histograms,
- Scatter plots,
- Binned scatter plots visualized as heatmaps (including outlier suppression via value clipping to improve plot readability).

Moreover, the system can compute the four-momentum of the W bosons directly from the four-momenta of the decay products (lepton and neutrino) via simple vector summation. These W-boson four-momenta are also compared to their ground-truth counterparts through the same range of visualizations.

4.1 Folder structure description

Before diving deeper into the used technologies, it is worth describing the general folder structure of the project. As the scale of the project has grown, a clear and logical way of organizing files and scripts has become crucial for maintaining readability and reproducibility.

- `configs/` — Contains configuration files for individual experiments. Each file specifies model hyperparameters, dataset paths, optimizer settings, learning rate schedule, and much more;
- `data/` — Holds both input datasets (raw and preprocessed);
- `images/` — Stores images used in this thesis;
- `logs/` — Stores TensorBoard log files of the trained models, such as loss value, evaluation metrics, and training time;
- `plots/` — Stores the visual outputs of the experiments, including scatter plots, histograms, heatmaps, and c_{ij} coefficient analyses;
- `saves/` — Stores Pytorch save files, which holds neural network learnable parameters and architecture;
- `utils/` — Contains high-level Python scripts for training, evaluation, visualization, data preprocessing, and metric calculation;
- `config_classic.yaml` — Configuration file, specifying which classical ML model to train, as well as additional hyperparameters;

- `config_pipeline.yaml` — Configuration file, defines path to save logs, plots and trained DNN models;
- `EDA.ipynb` — Notebook where and Exploratory Data Analysis is performed;
- `NeuralNetwork.py` — A Python script to parse config file of individual experiment and build Neural Network;
- `train_classic.ipynb` — Notebook orchestrating classical ML model training, prediction, and plot generation;
- `train_pipeline.ipynb` — Notebook orchestrating DNN model training, prediction, and plot generation;

This layout ensures that code, data, and results are separated, making the project easy to navigate and scale as new experiments are added.

4.2 Technologies

The pipeline was developed entirely in Python[1], version 3.10.4, which is widely regarded as one of the most versatile and accessible programming languages for machine learning and data analysis [67]. Its large ecosystem of libraries, strong community support, and the availability of interactive computing tools make it a natural choice for this research.

Python's design encourages clear, readable code, while its broad module support streamlines the implementation of common data science workflows such as data preprocessing, visualization, and model training.

All experiments were conducted on an HP ProBook 450 G8 laptop, with an 11th Gen Intel (R) Core (TM) i7–1165G7 CPU running at 2.80 GHz, 16 GB of DDR4 RAM, and an integrated Intel Iris Xe Graphics graphics card.

4.2.1 Jupyter Notebook

The studies were carried out in a Jupyter Notebook[2], an open-source web-based computing environment that combines live code execution, text documentation, and visualization in a single interface. This interactive development style was particularly valuable during the early stages of the project, where it enabled fast prototyping and visualization.

[1] https://www.python.org/
[2] https://jupyter.org/

4.2.2 Pandas

The `Pandas`[3] library was used for efficient manipulation and analysis of structured datasets. Its powerful DataFrame abstraction allowed for seamless loading, filtering, and transformation of tabular data, which was particularly useful for handling large simulation outputs and preparing them for model input.

4.2.3 NumPy

The `NumPy`[4] library was employed for numerical computation throughout this project. Its efficient implementation of multi-dimensional array operations was beneficial for both data transformation and mathematical operations involving four-momenta and probability distributions.

4.2.4 Seaborn and matplotlib

To analyze the results and visualize data in the process of Exploratory Data Analysis (EDA), the pipeline made extensive use of the `matplotlib`[5] and `Seaborn`[6] libraries. These tools were handy for generating scatter plots, histograms, and heatmaps. They helped assess both model performance and the properties of the reconstructed physical quantities.

4.2.5 SciPy

The `SciPy`[7] library provided various scientific computing tools, including statistical tests and interpolation routines, which were used in the project for data preparation and exploration.

4.2.6 Scikit-learn

The `Scikit-learn`[8] library was used primarily for evaluation metrics and data manipulation tasks such as data scaling and train-test splitting. Its straightforward API made integrating the evaluation process into the training pipeline easy.

[3] https://pandas.pydata.org/
[4] https://numpy.org/
[5] https://matplotlib.org/
[6] https://seaborn.pydata.org/
[7] https://scipy.org/
[8] https://scikit-learn.org/stable/

4.2.7 PyTorch

The neural network models were implemented using `PyTorch`[9], a popular open-source machine learning framework. PyTorch's dynamic computational graph model, intuitive syntax, and GPU support made it an ideal choice for fast prototyping and iterative experimentation in this research.

4.2.8 TensorBoard

`TensorBoard`[10] was used to monitor the training progress of the models, visualizing metrics such as loss, accuracy, and learning rate over time. This tool supplied critical insights of the training process and allowed for easier hyperparameter tuning.

[9] https://pytorch.org/
[10] https://www.tensorflow.org/tensorboard

Chapter 5

Dataset

This chapter describes the dataset used to analyze the decay process $H \to WW^* \to \ell\nu\ell\nu$. The dataset consists of simulated proton-proton collision events, which follow SM predictions and were generated using a multi-step simulation chain to replicate realistic LHC conditions.

5.1 Data simulation

The following description is based on 'Laboratory-frame tests of quantum entanglement in $H \to WW$' [68]. The hard scattering process, representing the Higgs production via ggF and its decay into two W bosons, was simulated at leading order (LO) using **MadGraph5_aMC@NLO v5.4.2** [69] Monte Carlo generator. Production mode was modeled via a contact interaction[1]. Both Higgs event signals and background signals (non-resonant events) were generated to simulate the signal that is usually produced in SM processes.

Obtained parton-level events were processed through **PYTHIA 8.3** [70] to simulate:

- Parton showering: emission of additional quarks/gluons via QCD radiation;
- Hadronization: formation of stable particles from quarks/gluons;
- Underlying events: soft interactions between proton remnants not involved in the hard scattering.

A simplified detector response was modeled using **Delphes 3.5.0** [71], which includes:

- Tracking: reconstruction of charged particle trajectories in the inner detector;

[1] An approximation where a complex interaction like a loop process is simplified to a direct coupling at a single spacetime point, ignoring internal details like virtual particles

- Calorimetry: measurement of energy deposits from electrons, photons, and jets;
- Missing transverse momentum p_T^{miss}. Transverse momentum p_T itself is the component of a particle's momentum perpendicular to the beam axis (z-axis). The high value of p_T indicates a particle produced in a high-energy collision, making it more likely to originate from a Higgs decay rather than background processes. The transverse momentum vector sum must be zero after the collision has happened. Therefore, any imbalance in the total transverse momentum (missing transverse momentum) signals the presence of invisible particles.

5.2 Selection cuts

After the simulation, additional selection cuts can be applied to ensure each event contains exactly one Higgs boson, two W bosons, and their respective decay products (leptons and neutrinos). The selection cuts used by the ATLAS Collaboration for the $H \to WW^* \to \ell\nu\ell\nu$ decay channel are designed to isolate Higgs signals from backgrounds [72]. The standard criteria limit the maximum pseudorapidity η of both leptons and the minimum transverse momentum p_T of leading and subleading leptons. In this dataset, two charged leptons are labeled with indices 1 and 2, denoting the **leading** and **subleading** lepton, respectively. This distinction is based on their *transverse momentum* p_T, defined as:

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

The leading lepton is the one with the greater transverse momentum, while the subleading lepton has the lower value. This is also applicable to neutrinos. Table 5.1 lists the exact values applied. Missing transverse momentum cuts ensure the primary lepton has sufficient energy to trigger the detector and suppress low-energy backgrounds while balancing signal retention and background rejection, as softer leptons are still likely in Higgs boson decays. A pseudorapidity cut ensures that leptons are within the tracking acceptance of the ATLAS inner detector, where measurement precision is highest.

Table 5.1 Selection cuts applied to leptons

Parameter	Value
minimum leading lepton p_T	22 GeV
minimum subleading lepton p_T	15 GeV
maximum lepton η	2.5

5.3 Data extraction

The dataset used in this thesis was initially stored in the `ROOT` file format. To perform further analysis using machine learning tools in Python, it was necessary to convert the data into a more accessible tabular format, specifically `CSV`.

For this purpose, a Python script was utilized, which employs the `uproot` library to extract the event information from the `ROOT` files and store it in a `pandas DataFrame`. The script iterates over all entries and separates the relevant sub-quantities. For each particle, the total energy (`fE`) and total momentum magnitude (`fP`) are extracted. The momentum vector is further decomposed into its Cartesian components (`fX`, `fY`, `fZ`). The resulting table is exported and saved in `CSV` format for subsequent analysis.

5.4 Exploratory Data Analysis

This section describes an exploratory analysis of the dataset used in this work. The aim is to understand the data structure, its statistical properties, the nature of the features and the targets, as well as to assess the presence of outliers or any data quality issues that might affect training performance.

5.4.1 Dataset overview

The dataset consists of 1,000,000 simulated events. Each event includes information about two charged leptons and two neutrinos, represented in the form of four-momenta, along with the components of the missing transverse momentum arising from the undetectable neutrinos. All features are real-valued floating-point numbers.

The input features are composed of the four-momenta of the two leptons in the final state (Table 5.2) and the x and y components of the missing transverse momentum (Table 5.5). The target values consist of the four-momenta of the two neutrinos associated with each lepton (Table 5.3). The missing transverse momentum components, `mpx` and `mpy`, are derived as the sum of the transverse components of the two neutrinos' momenta. Properties of the W bosons are described at the Table 5.4.

5.4.2 Statistical summary

Before inspecting statistical properties, the dataset was split into training (80%), validation (10%), and testing (10%) subsets.

Table 5.2 Description of lepton 4-momenta features.

Feature	Description
l1_E	Energy of the leading lepton
l1_p_x	x-component of the leading lepton's momentum
l1_p_y	y-component of the leading lepton's momentum
l1_p_z	z-component of the leading lepton's momentum
l2_E	Energy of the subleading lepton
l2_p_x	x-component of the subleading lepton's momentum
l2_p_y	y-component of the subleading lepton's momentum
l2_p_z	z-component of the subleading lepton's momentum

Table 5.3 Description of neutrino 4-momenta features.

Feature	Description
n1_E	Energy of the neutrino associated with the leading lepton
n1_p_x	x-component of the neutrino associated with the leading lepton
n1_p_y	y-component of the neutrino associated with the leading lepton
n1_p_z	z-component of the neutrino associated with the leading lepton
n2_E	Energy of the neutrino associated with the subleading lepton
n2_p_x	x-component of the neutrino associated with the subleading lepton
n2_p_y	y-component of the neutrino associated with the subleading lepton
n2_p_z	z-component of the neutrino associated with the subleading lepton

Table 5.4 Description of W boson 4-momenta features.

Feature	Description
w1_E	Energy of the W boson which produced the leading lepton
w1_p_x	x-component of the W boson which produced the leading lepton
w1_p_y	y-component of the W boson which produced the leading lepton
w1_p_z	z-component of the W boson which produced the leading lepton
w2_E	Energy of the W boson which produced the subleading lepton
w2_p_x	x-component of the W boson which produced the subleading lepton
w2_p_y	y-component of the W boson which produced the subleading lepton
w2_p_z	z-component of the W boson which produced the subleading lepton

Tables A.1 and A.2 summarize the statistical properties of the input and target training datasets without selection cuts applied, respectively. It is worth noting that the energy and z-momentum components have a significantly

Table 5.5 Description of missing transverse momentum components.

Feature	Description
`mpx`	Missing transverse momentum in the x-direction
`mpy`	Missing transverse momentum in the y-direction

broader range and higher standard deviation compared to the x and y components. This can negatively affect gradient-based optimizers and the convergence time of the models.

As listed in Table A.3 and Table A.4, the application of selection criteria to isolate the $H \to WW^* \to \ell\nu\ell\nu$ signal significantly alters the statistical composition of the dataset. An analysis and comparison of the most significant changes before and after the cuts is presented below:

- Mean value of leading lepton energy (`l1_E`) decreases from 97.1 GeV (uncut) to 63.7 GeV (cut) (-34%);
- Mean value of subleading lepton energy (`l2_E`) drops from 96.8 GeV to 50.2 GeV (-48%);
- Standard deviation of `l1_E` collapses by 72% (146.4 GeV to 41.1 GeV), indicating tighter energy constraints;
- Extreme `l1_E` values (up to ±3457 GeV uncut) are capped at ±355 GeV post-cut. Similar situation occurs to ranges of `l2_E`, `l1_p_z` and `l2_p_z` — all of them have been significantly shortened;
- Standard deviation of `mpx` and `mpy` is decreased by 9% (28.7 → 31.2 GeV), indicating tighter missing energy constraints;
- Mean value of `mpx` remains at zero, as expected for neutrino momentum balance in Higgs decays;
- Cuts reduce the dataset size by 73% (800k → 215k).

This means that the cuts prefer lower-energy leptons, which are consistent with Higgs decays while rejecting high-energy outliers from background processes. This selection criteria effectively isolate Higgs signal events by filtering leptons within detector acceptance and reducing background-induced outliers.

5.4.3 Univariate analysis

Here follows the illustration of the impact of selection cuts on the kinematic distributions of the simulated dataset used for the analysis. The blue histograms correspond to the original, unfiltered dataset, while the orange histograms represent the subset of events that survive the applied selection criteria.

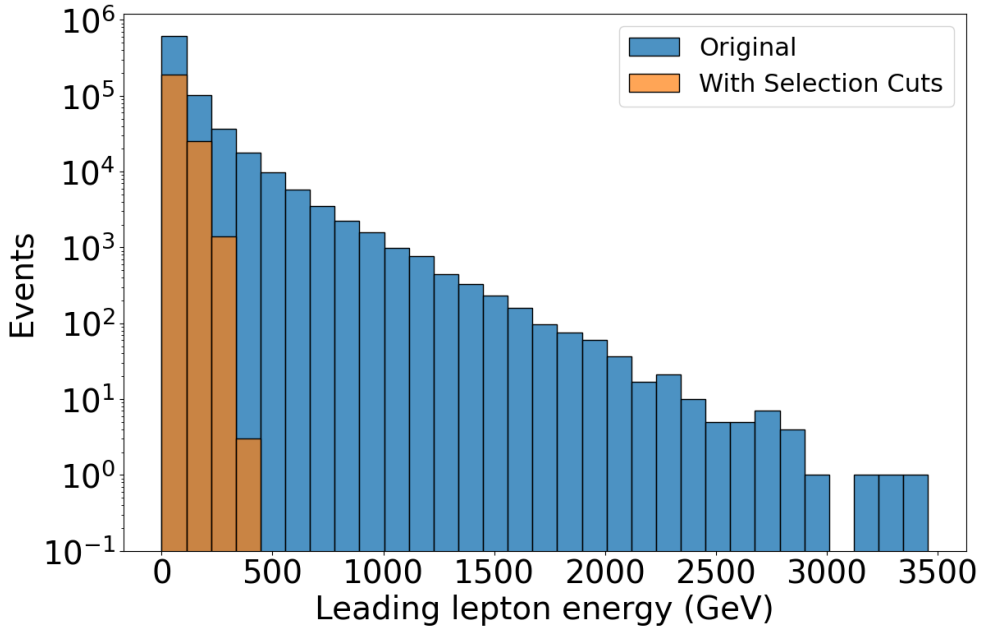

Figure 5.1 Distribution of leading lepton energy before (original) and after selection cuts applied.

Figure 5.1 shows the distribution of the leading lepton energy ($l1_E$). The original dataset spans a broad range of energies, with a long tail extending to several thousand GeV. After selection cuts are applied, the distribution becomes significantly narrower, with high-energy outliers removed and most events concentrated at lower energies.

Figures B.1 and B.2 provide a comprehensive overview of the distributions for all input features: the energies and momentum components of both leptons, as well as the missing transverse momentum components (`mpx`, `mpy`). For each variable, the selection cuts reduce the range and variance, particularly evident in the energy distributions. The cuts also remove extreme values, resulting in more physically plausible events that are within the acceptance of the detector (e.g., `l1_p_x` and `l2_p_y`). Similar behavior observed for target features (Figures B.3 and B.4).

To further investigate the structure and relationships within the dataset, Figure 5.2 shows the correlation matrix for the input dataset. In each cell, the Pearson correlation coefficient is calculated between a pair of features, with the intensity of color indicating the strength and direction of the correlation. Particularly, strong negative correlations are observed between the transverse momentum components of the leptons (`l1_p_x`, `l1_p_y`) and the missing transverse momentum components (`mpx`, `mpy`), as expected from momentum conservation in the transverse plane. Moderate positive correlations are also seen between the energies and longitudinal momenta of the leptons. Se-

lection cuts only slightly change the Pearson correlation coefficients, as shown in Figure 5.3.

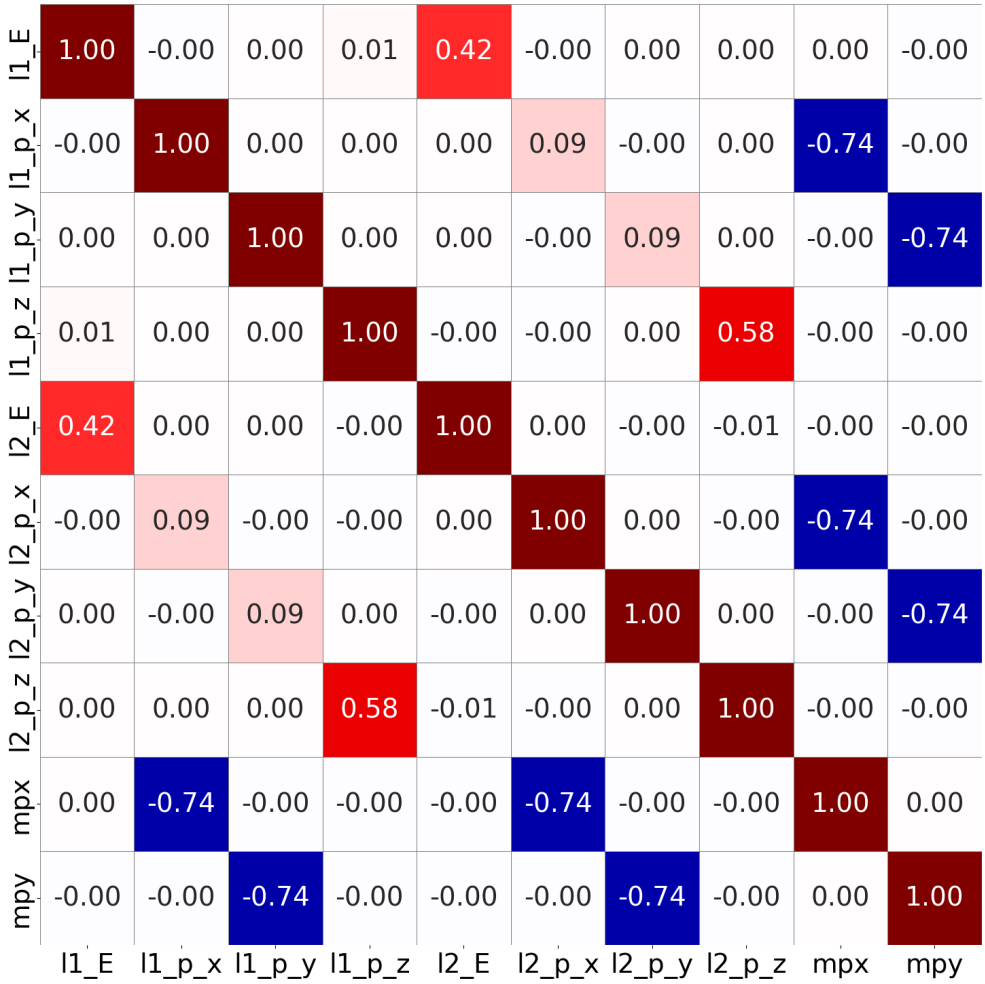

Figure 5.2 Correlation plot of features before selection cuts applied.

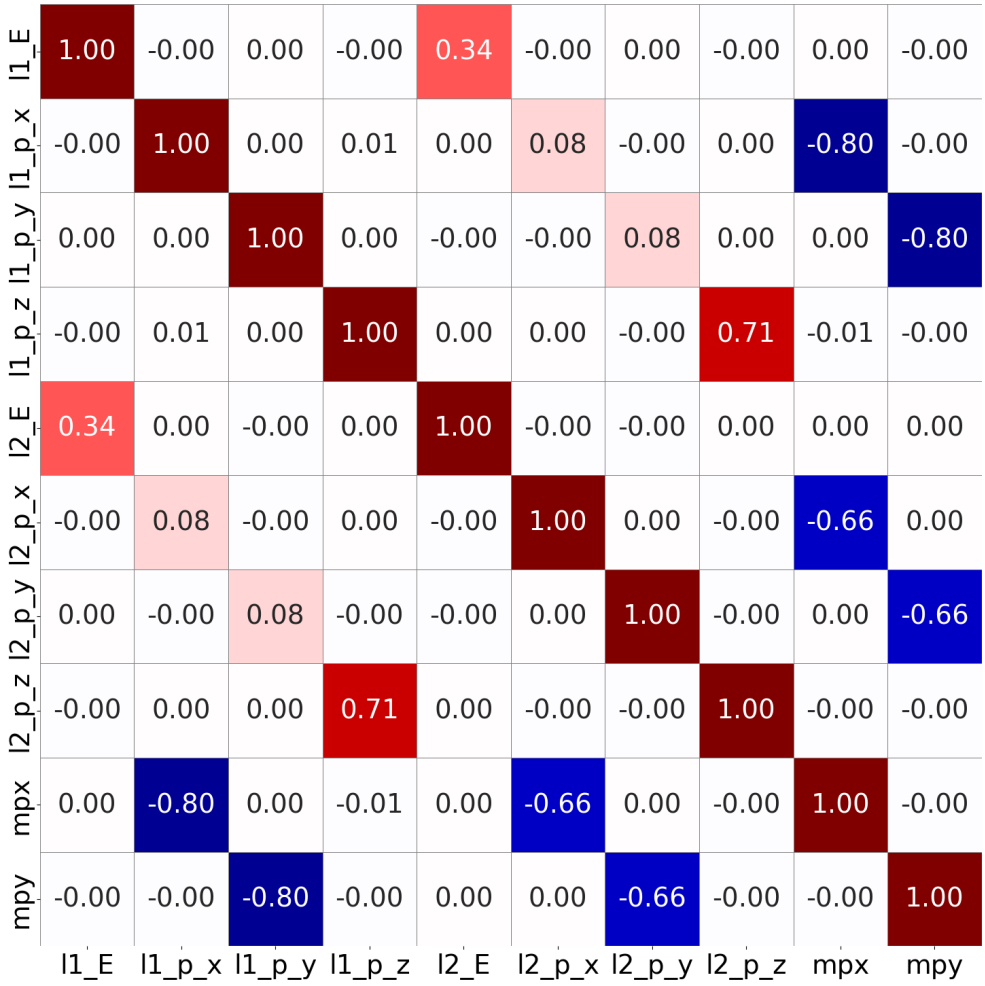

Figure 5.3 Correlation plot of features after selection cuts applied.

Chapter 6

Experiments and Results

This chapter describes the conducted experiments, including hyperparameter tuning and different architectures, and a discussion of their results.

6.1 Classical ML methods

The first experiments were conducted on classical ML models — Random Forest and AdaBoost. For the AdaBoost algorithm, a decision tree regressor was selected as the base estimator due to the nonlinear shape of the data and the ability to partition the feature space into distinct regions.

6.1.1 Random Forest

A simple grid search was used for hyperparameter tuning. Each search space is represented by several values. They are written manually in distinct config files. Hyperparameters that were chosen to try out in the starting configuration were:

- Depths of the decision tree (as the estimator model in Random Forest) — 4, 10, 16, and 22;
- Number of estimators in the model — 2, 8, 14, and 20.

MAE was chosen as a loss function. One can see the value of the loss function for different combinations in Figure 6.1. The best model in this iteration has 20 Decision Trees with a depth of 16.

Further search was split into two directions: either to experiment with estimators' depth or their number. The evaluation shows that while training loss was smaller with deeper trees, there was almost no improvement in validation metrics. Figures 6.2 and 6.3 demonstrate different values of MAE for each hyperparameter combination. Combining both approaches results in a model with 100 Decision Trees with a depth of 20. Values of MAE and RMSE for

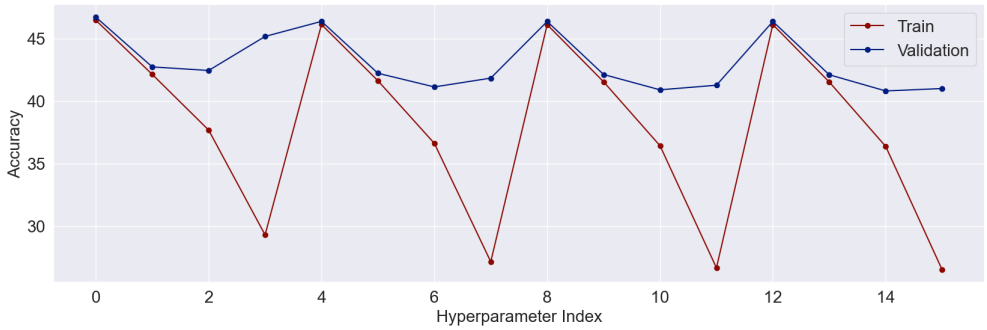

Figure 6.1 Loss function (MAE) for different combinations of hyperparameters (Cartesian product of number of estimators [2, 8, 14, 20] and tree depths [4, 10, 16, 22]) in a first iteration of hyperparameter tuning for Random Forest.

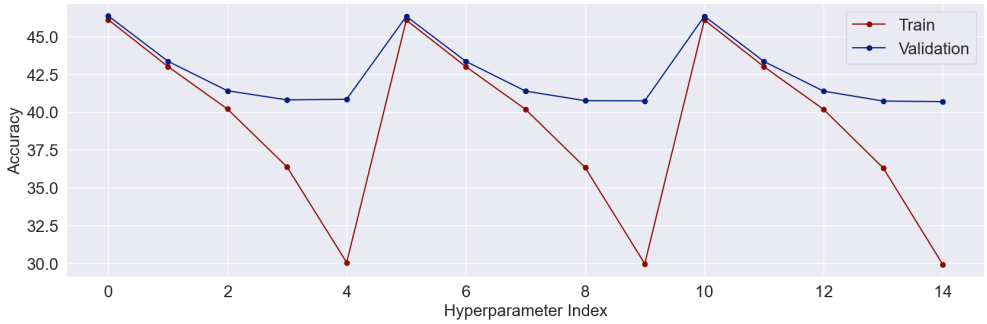

Figure 6.2 Loss function (MAE) for different combinations of hyperparameters (Cartesian product of number of estimators [4, 8, 12, 16, 20] and tree depths [20, 25, 30]) in a depth-focused iteration of hyperparameter tuning for Random Forest.

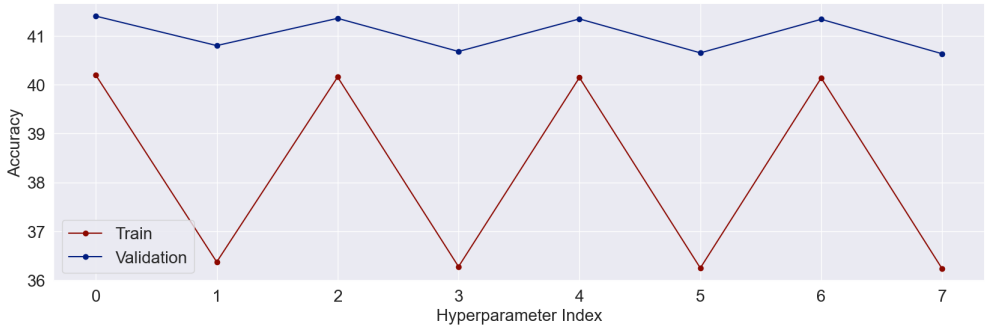

Figure 6.3 Loss function (MAE) for different combinations of hyperparameters (Cartesian product of number of estimators [20, 40, 60, 80] and tree depths [12, 16]) in a breadth-focused iteration of hyperparameter tuning for Random Forest.

each iteration are listed in Table 6.1. Due to the lack of improvement in validation metrics, this model was not pursued further and was retained for later comparison.

Table 6.1 Results of evaluating best Random Forest models from each iteration of hyperparameter tuning with MAE and RMSE for multi-output regression (average across all target features) on the dataset without selection cuts.

Iteration	First		Depth		Breadth		Final	
	Train	Val.	Train	Val.	Train	Val.	Train	Val.
MAE	36.37	40.80	29.91	40.69	36.23	40.64	29.76	40.47
RMSE	-	71.63	-	71.50	-	71.26	-	71.04

6.1.2 AdaBoost

A similar strategy was used in the beginning steps of tuning. The chosen values of the hyperparameters to test were:

- Depths of the decision tree (as the estimator model in Random Forest) — 4, 10, 16, and 22;
- Number of estimators in the model — 8, 14 and 20;
- Learning rate — 1, 0.1 and 0.01.

First AdaBoost model has a learning rate of 0.01, 20 weak learners (Decision Trees) of depth 22. The tuning was split into researching deeper trees (values of 22, 26, and 30 were explored) and more weak learners (20, 26, and 32). The LR value was additionally tested, with the values of 0.01, 0.04, and 0.07.

Again, deeper Trees showed better performance on train data, but validation metrics are even closer to each other. The final AdaBoost model has a learning rate of 0.01 and 32 estimators with a depth of 22. Values of MAE and RMSE for each iteration are listed in Table 6.2. Figures 6.4, 6.5, and 6.6 demonstrate different values of MAE for each hyperparameter combination.

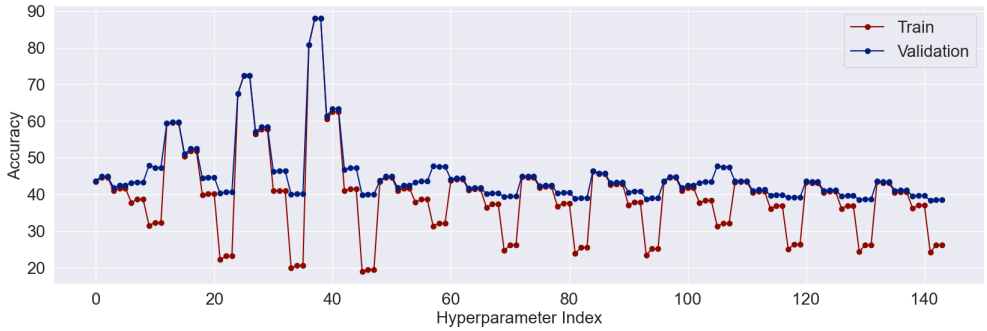

Figure 6.4 Loss function (MAE) for different combinations of hyperparameters (Cartesian product of number of estimators [8, 14, 20], learning rate [1, 0.1, 0.01] and tree depths [10, 16, 22]) in a first iteration of hyperparameter tuning for AdaBoost.

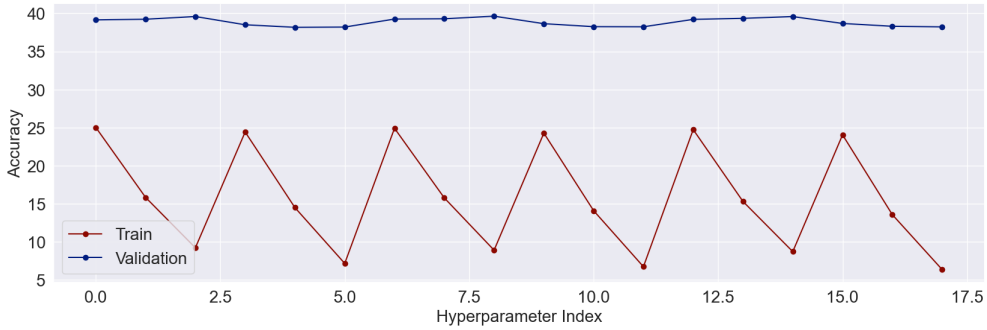

Figure 6.5 Loss function (MAE) for different combinations of hyperparameters (Cartesian product of number of estimators [8, 14], learning rate [0.01, 0.04, 0.07] and tree depths [22, 26, 30]) in a depth-focused iteration of hyperparameter tuning for AdaBoost.

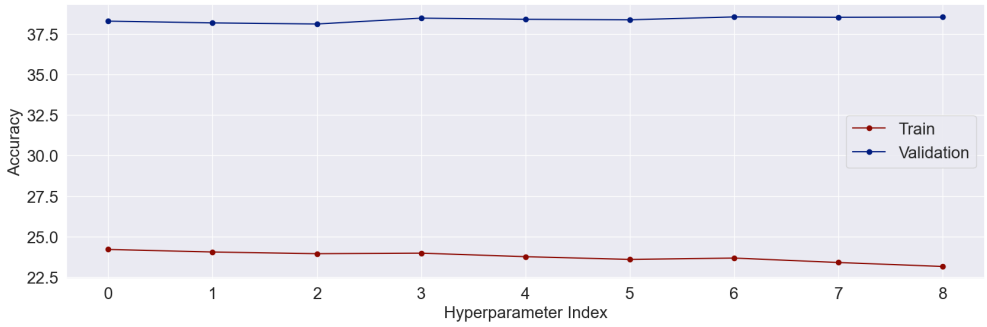

Figure 6.6 Loss function (MAE) for different combinations of hyperparameters (Cartesian product of number of estimators [20, 26. 32], learning rate [0.01, 0.04, 0.07] and tree depths [22]) in a breadth-focused iteration of hyperparameter tuning for AdaBoost.

Table 6.2 Results of evaluating best AdaBoost models from each iteration of hyperparameter tuning with MAE and RMSE for multi-output regression (average across all target features) on the dataset without selection cuts.

Iteration	First		Depth		Breadth		Final	
	Train	Val.	Train	Val.	Train	Val.	Train	Val.
MAE	26.19	38.49	14.49	38.17	23.94	38.12	23.94	38.12
RMSE	-	72.05	-	74.04	-	72.16	-	72.16

6.2 Deep Neural Network

Hyperparameter optimization of DNN was done manually by creating separate config files, which also hold the architecture of the Neural Network — each layer with a number of neurons and other parameters. Throughout the tuning process, the following values were tested:

- Dataset type: Whether to scale data with StandartScaler [73] or RobustScaler [74] or use default set;
- Number of epochs for early stopping: how many epochs to wait before and improvement in validation metric;
- LR type: whether to use static LR or change it in the process of training with the scheduler. If the scheduler was chosen, additional parameters were chosen, such as the number of epochs for the burn-in feature and specific checkpoints (indexes of epochs) to change LR;
- LR : base value, which might be changed with scheduler;
- Activation function: Rectified Linear Unit (RELU) or Leaky RELU were tested;
- Loss function: initially, MSE was chosen, but then changed to MAE for better interpretability;
- Optimizer: Stochastic gradient descent (SGD), SGD with momentum, Root Mean Square Propagation (RMSProp), and Adaptive Moment Estimation (Adam);
- Regularization: L1, L2 norms, Gaussian noise and Dropout;
- Number of neurons in the layers.

During optimization, the architecture was changed several times to contain fewer layers and neurons. The goal was to construct an accurate model that simultaneously contained as few parameters as possible. Following the hyperparameter tuning, the final model consists of 8 fully connected layers.

It uses Adam optimizer with the Leaky RELU activation function. The base value of LR is 0.00004; it is scheduled to increase in the 40-th to 0.002 and in the 60-th epoch to 0.005, and then decrease it in the 80-th and 100-th epochs to previous values, as can be seen in Figure 6.7. The model was trained on the standardized dataset. With a batch size of 128 for training and 256 for validation and testing, the model was trained for 300 epochs, with an early stopping patience of 20 epochs.

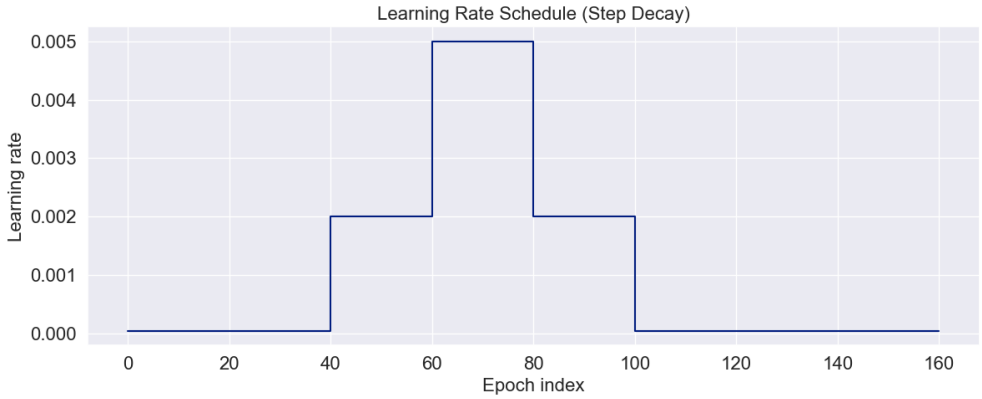

Figure 6.7 The learning rate value throughout training the final model. The X-axis holds the epoch index, and the Y-axis is a real value.

6.3 Effects of selection cuts

After exploring different models on the original dataset, the selection cuts were used on the data. All models were retrained on cut data and evaluated using the same metrics. Tables 6.3 and 6.4 list exact values of MAE and RMSE on the train and validation dataset with selection cuts applied. About 60% decrease in MAE validation error is observed for the best classical models. One can also notice an improvement in histograms of predicted values of neutrinos against the true ones (Figures C.1 and C.2), especially in x and y components.

Table 6.3 Results of evaluating best Random Forest models from each iteration of hyperparameter tuning with MAE and RMSE for multi-output regression (average across all target features) on the dataset with selection cuts.

Iteration	First		Depth		Breadth		Final	
	Train	Val.	Train	Val.	Train	Val.	Train	Val.
MAE	19.55	24.79	19.51	24.72	19.43	24.63	14.35	24.61
RMSE	-	38.61	-	38.46	-	38.32	-	38.25

Table 6.4 Results of evaluating best AdaBoost models from each iteration of hyperparameter tuning with MAE and RMSE for multi-output regression (average across all target features) on the dataset without selection cuts.

Iteration	First		Depth		Breadth		Final	
	Train	Val.	Train	Val.	Train	Val.	Train	Val.
MAE	9.09	23.41	9.85	23.80	4.66	23.52	4.66	23.52
RMSE	-	38.87	-	39.39	-	39.67	-	39.67

6.4 Results

For the final choice, the validation metrics of the three best models were compared. One can see them for an original dataset in Table 6.5, and for the dataset with selection, cuts applied in Table 6.6. It is shown that the Neural Network performs better than classical ML methods. The MAE metric for each target is significantly lower, especially for Energy and z-component features. Therefore, the best DNN was chosen for both datasets. Table 6.7 lists the architecture of the model. The estimated MAE on test data is 35.25 and 22.42 on the original dataset and the one with selection cuts applied, respectively. Figures C.3 and C.4 show the distribution of reconstructed W boson features (using predicted neutrino properties) against the true values.

Table 6.5 MAE for several models calculated on the dataset without selection cuts.

Feature	MAE		
	NN	Random forest	AdaBoost
v1_E	57.563	67.134	63.139
v1_p_x	9.163	9.663	9.264
v1_p_y	9.125	9.645	9.222
v1_p_z	66.454	75.938	71.381
v2_E	57.264	66.645	62.560
v2_p_x	9.171	9.658	9.261
v2_p_y	9.134	9.646	9.225
v2_p_z	66.118	75.467	70.892
Average	**35.499**	40.475	38.118

Table 6.6 MAE for several models calculated on the dataset with selection cuts applied.

Feature	MAE		
	NN	Random forest	AdaBoost
`v1_E`	36.349	40.875	38.761
`v1_p_x`	8.213	8.778	8.357
`v1_p_y`	8.209	8.788	8.342
`v1_p_z`	44.326	48.551	47.007
`v2_E`	29.512	33.578	31.203
`v2_p_x`	8.210	8.717	8.345
`v2_p_y`	8.205	8.697	8.343
`v2_p_z`	35.289	38.911	36.985
Average	**22.289**	24.612	23.418

Table 6.7 Summary of the best DNN architecture.

Type of layer	Number of neurons
`input`	10
`linear`	16
`linear`	32
`linear`	64
`linear`	128
`linear`	64
`linear`	32
`linear`	16
`linear`	8
`identity`	8

Chapter 7

Conclusion

In this thesis, the primary objective was to reconstruct the kinematic properties of W bosons in the $H \to WW^* \to \ell\nu\ell\nu$ final state. Accurate reconstruction of these properties is essential for testing Bell inequalities and studying quantum entanglement in the Higgs boson and W boson systems. The work was divided into several tasks to achieve this goal, from theoretical background research to developing, optimizing, and evaluating machine learning models.

First, a theoretical overview of the Higgs boson production and decay mechanisms was provided, focusing on gluon-gluon fusion and the subsequent decay into W bosons. The following chapter covered the basics of machine learning theory as well as methods, metrics, and additional tools, which were used in this thesis to achieve the stated goal. It also contains explanations of key terms for the reader to better understand the proposed solutions in this thesis's later stages. A modular deep-learning pipeline developed for this research is described. The pipeline handles the complete training workflow, from data loading to evaluation, and includes tools for advanced visualization of both regression performance and quantum correlation observables. Next, a detailed analysis of the simulated dataset was carried out, including the extraction of features and targets from ROOT files and an extensive exploratory data analysis to understand the statistical properties and structure of the data.

Two classical machine learning algorithms — Random Forest Regression and AdaBoost Regression — were implemented and evaluated for the reconstruction task. These models provided a baseline for further comparisons. In parallel, several DNN models were trained, fine-tuned, and tested. They achieve strong performance in matching the true momenta of the W bosons, demonstrating the feasibility of applying machine learning techniques to this challenging task. All aspects of the thesis assignment have been fulfilled, addressing the physics objectives and the application of advanced methods.

Testing basic DNNs and optimizing specific hyperparameters of their structures led to the creation of an architecture that delivers excellent results while remaining relatively simple and compact. The additional result of this work

is a fully operational, flexible, and extensible pipeline for the regression-based reconstruction of W boson kinematics in leptonic Higgs decay. The developed framework provides a solid foundation for future studies in reconstructing particle kinematics.

While the models developed in this thesis achieved promising results, there remains much room for future improvements. More sophisticated approaches, such as Conditional Denoising Diffusion Model [75] or unfolding techniques like OmniFold [76], could further enhance the quality of the reconstruction.

Appendix A

Summary statistics

Table A.1 Summary statistics of the input features set (in GeV). The table includes basic descriptive statistics — mean, standard deviation (Std), minimum (Min), 25th percentile (25%), median (50%), 75th percentile (75%), and maximum (Max) — for each input feature. The features correspond to the four-momentum components of two leptons from the Higgs boson decay and missing transverse momentum in x and y directions.

Feature	l1_E	l1_p_x	l1_p_y	l1_p_z	l2_E	l2_p_x	l2_p_y	l2_p_z	mpx	mpy
Mean	97.098	0.013	-0.031	0.276	96.789	-0.021	-0.016	0.035	0.005	0.047
Std	146.362	19.453	19.431	173.476	144.790	19.440	19.420	171.980	28.748	28.699
Min	0.279	-60.545	-61.542	-3457.412	0.219	-60.347	-62.005	-3818.028	-64.661	-63.799
25%	27.268	-13.231	-13.269	-36.613	27.321	-13.270	-13.240	-36.692	-23.806	-23.703
50%	47.991	-0.007	-0.037	0.032	48.003	-0.031	-0.037	-0.014	0.012	0.105
75%	101.634	13.243	13.199	36.908	101.572	13.234	13.223	36.893	23.819	23.837
max	3457.443	60.957	60.382	3123.345	3818.037	61.440	61.582	3489.573	63.392	63.894

Table A.2 Summary statistics of the target features set (in GeV). The table includes basic descriptive statistics — mean, standard deviation (Std), minimum (Min), 25th percentile (25%), median (50%), 75th percentile (75%), and maximum (Max) — for each target feature. They correspond to the four-momentum components of two neutrinos from the Higgs boson decay.

Feature	v1_E	v1_p_x	v1_p_y	v1_p_z	v2_E	v2_p_x	v2_p_y	v2_p_z
Mean	96.597	-0.004	-0.006	0.108	96.601	0.008	0.053	0.193
Std	144.568	19.478	19.441	171.679	144.978	19.438	19.426	172.032
Min	0.111	-60.639	-60.913	-3178.377	0.095	-61.518	-61.144	-2976.134
25%	27.265	-13.272	-13.234	-36.652	27.223	-13.219	-13.207	-36.551
50%	47.999	0.006	-0.015	0.040	47.892	-0.025	0.047	0.025
75%	101.303	13.254	13.233	36.910	101.064	13.247	13.259	36.741
max	3178.535	60.631	61.506	3055.023	3542.116	61.101	62.540	3542.108

Table A.3 Summary statistics of the input features set with selection cuts applied (in GeV). The table includes basic descriptive statistics — mean, standard deviation (Std), minimum (Min), 25th percentile (25%), median (50%), 75th percentile (75%), and maximum (Max) — for each input feature. The features correspond to the four-momentum components of two leptons from the Higgs boson decay and missing transverse momentum in x and y directions.

Feature	l1_E	l1_p_x	l1_p_y	l1_p_z	l2_E	l2_p_x	l2_p_y	l2_p_z	mpx	mpy
Mean	63.695	0.030	-0.069	0.062	50.203	-0.016	0.025	0.133	-0.014	0.049
Std	41.081	23.506	23.538	68.104	34.793	18.785	18.770	55.006	31.226	31.243
Min	22.005	-60.168	-60.760	-351.123	15.010	-59.753	-60.282	-322.844	-62.848	-63.799
25%	35.407	-21.181	-21.280	-34.579	26.594	-15.666	-15.586	-26.614	-26.901	-26.851
50%	48.992	0.061	-0.199	0.119	38.428	0.010	-0.002	-0.009	-0.041	0.105
75%	78.213	21.190	21.183	34.691	61.664	15.632	15.635	27.001	26.899	26.910
max	355.910	60.141	60.382	329.702	334.947	59.094	60.356	330.299	63.392	63.227

Table A.4 Summary statistics of the target features set with selection cuts applied (in GeV). The table includes basic descriptive statistics — mean, standard deviation (Std), minimum (Min), 25th percentile (25%), median (50%), 75th percentile (75%), and maximum (Max) — for each target feature. They correspond to the four-momentum components of two neutrinos from the Higgs boson decay.

Feature	v1_E	v1_p_x	v1_p_y	v1_p_z	v2_E	v2_p_x	v2_p_y	v2_p_z
Mean	69.775	0.014	0.005	0.217	55.157	-0.027	0.044	0.170
Std	78.405	22.087	22.102	100.198	65.264	17.886	17.866	81.625
Min	0.470	-59.655	-60.312	-1211.536	0.298	-60.851	-61.139	-1154.824
25%	27.221	-14.484	-14.497	-29.547	19.423	-10.738	-10.691	-22.887
50%	46.474	-0.024	0.004	0.042	35.199	-0.026	0.023	-0.009
75%	79.708	14.544	14.569	29.642	63.159	10.687	10.764	23.026
max	1388.941	60.631	60.713	1388.813	1378.474	61.101	62.540	1378.362

Appendix B

Univariate analysis

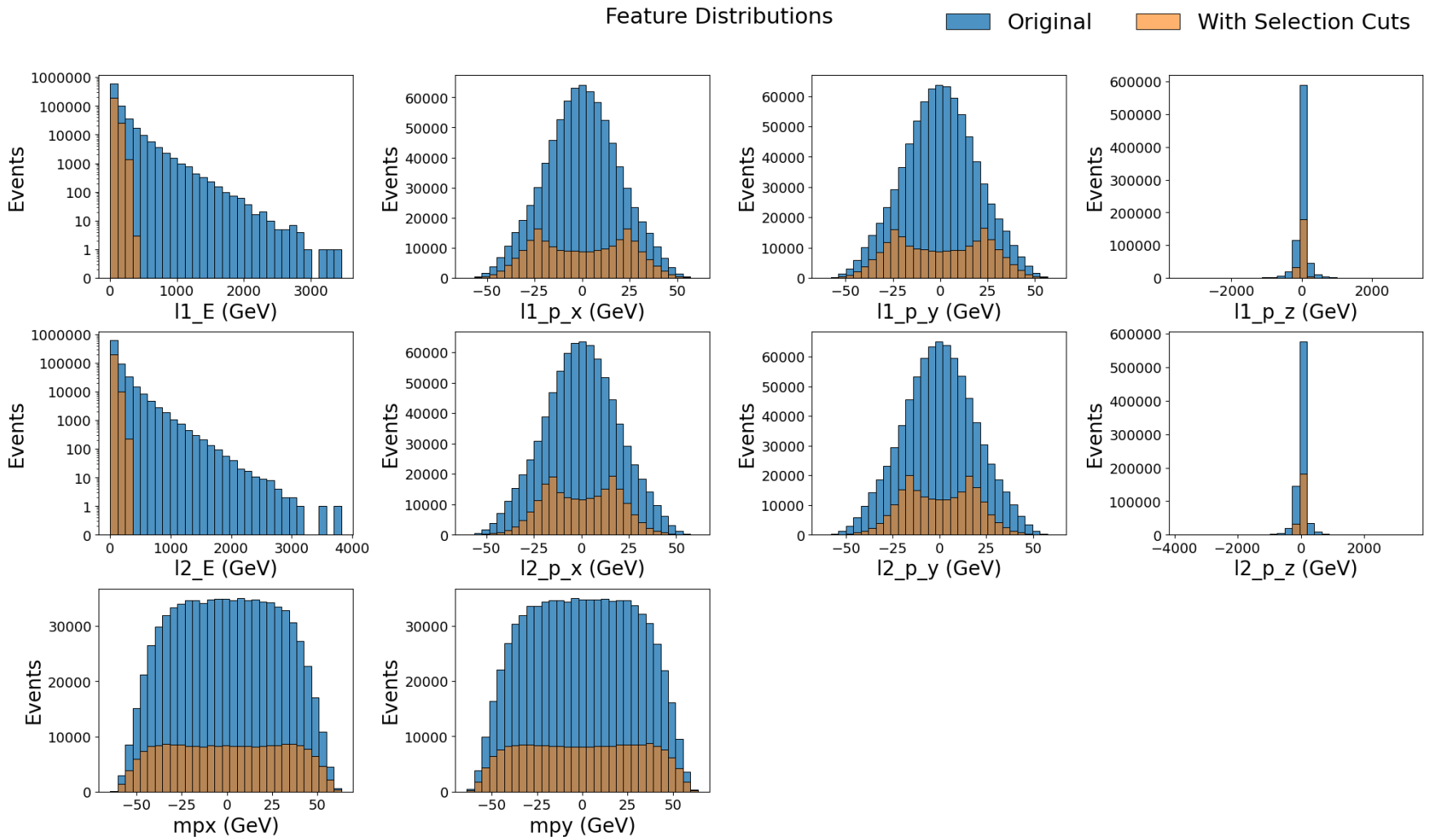

Figure B.1 Distributions of features before and after selection cuts applied.

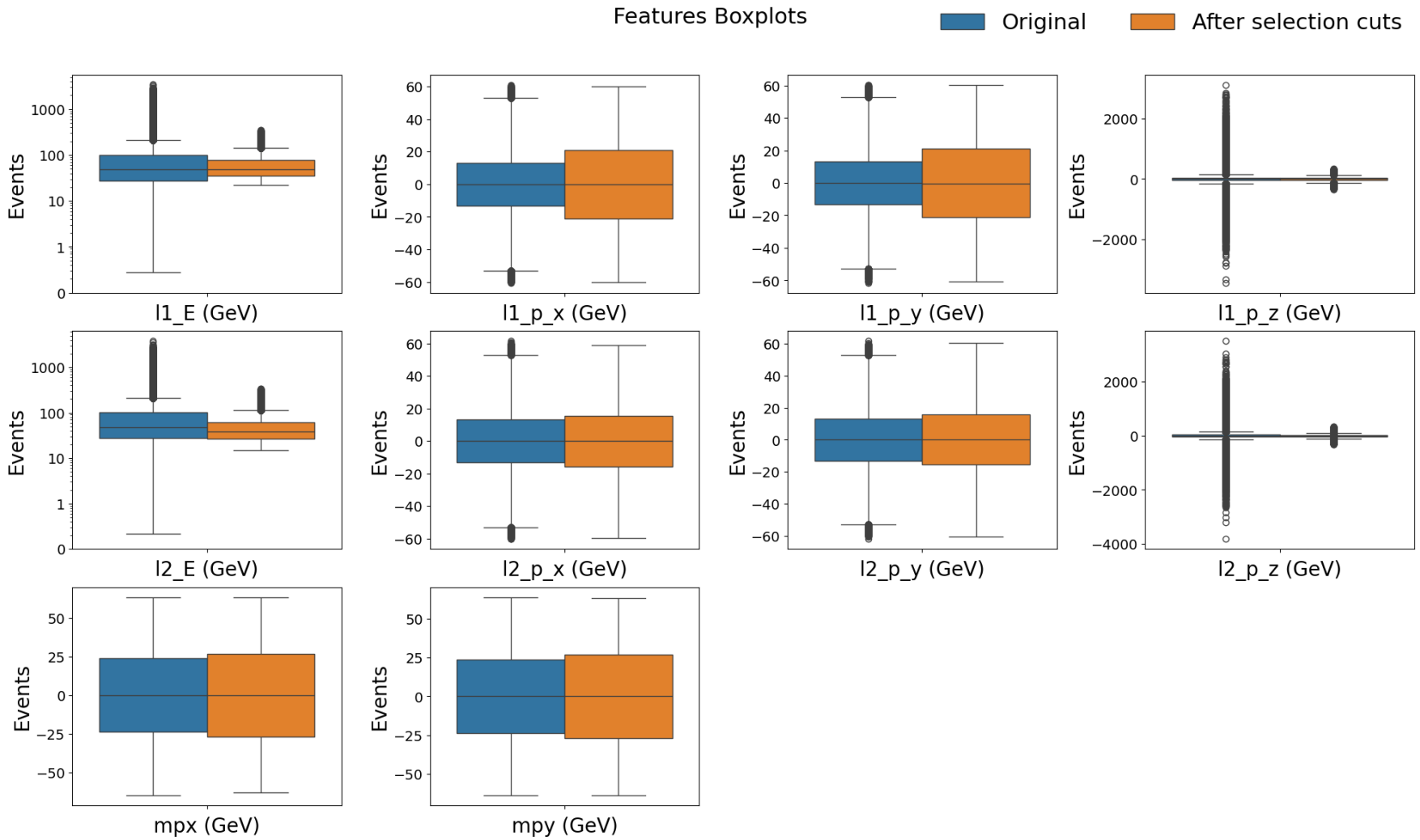

Figure B.2 Boxplots of input features before and after selection cuts applied. The vertical axis (log-scaled for energy plots) indicates the number of events, the horizontal axis — corresponding input feature. The central line in each box marks the median, while the box edges denote the interquartile range (25th to 75th percentiles). The whiskers extend to the most extreme data points within $1.5x$ the interquartile range. Outliers beyond this range are shown as individual points.

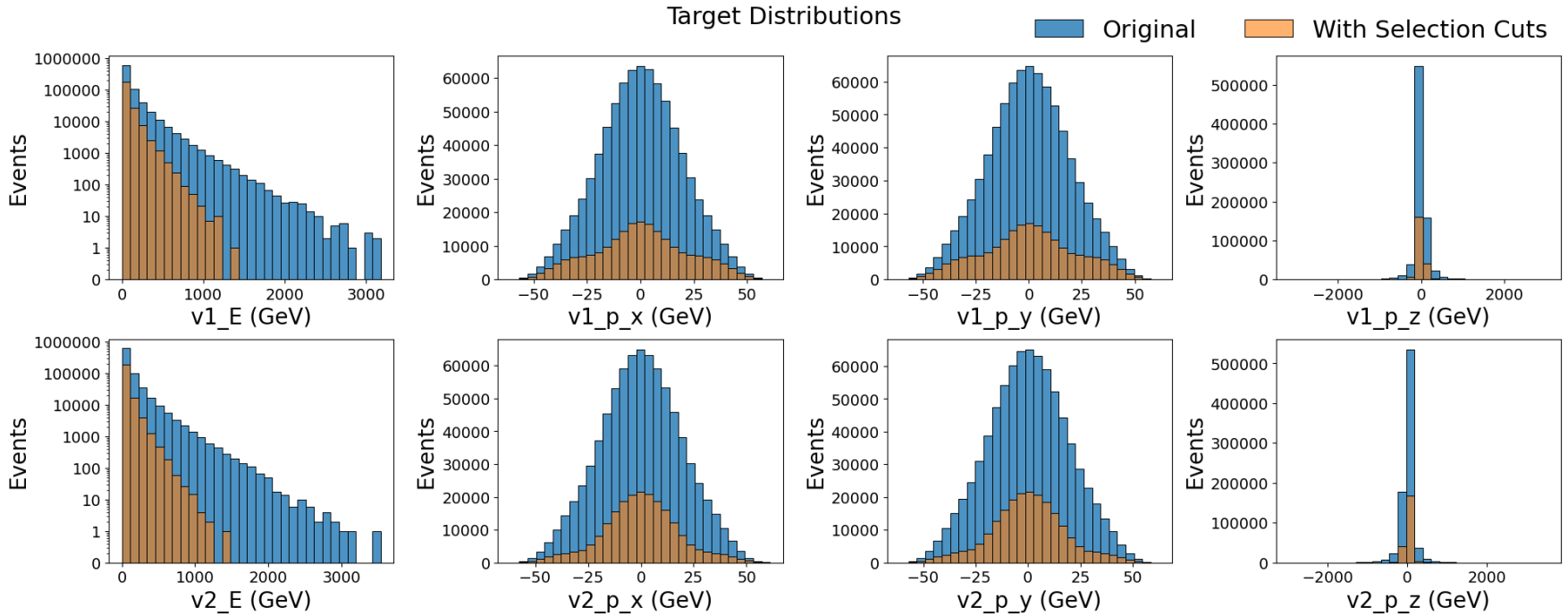

Figure B.3 Distributions of targets before and after selection cuts applied

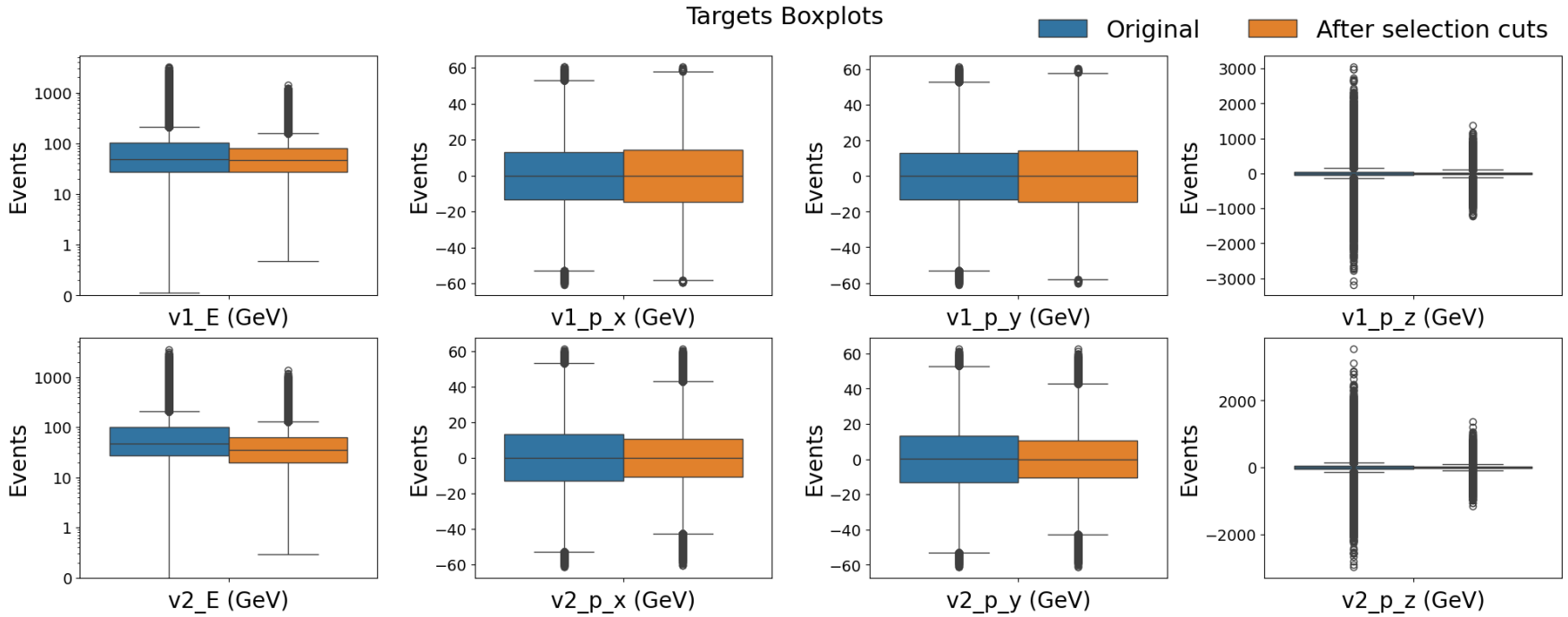

Figure B.4 Boxplots of targets before and after selection cuts applied. The vertical axis (log-scaled for energy plots) indicates the number of events, the horizontal axis — corresponding target feature. The central line in each box marks the median, while the box edges denote the interquartile range (25th to 75th percentiles). The whiskers extend to the most extreme data points within $1.5x$ the interquartile range. Outliers beyond this range are shown as individual points.

Appendix C

Results

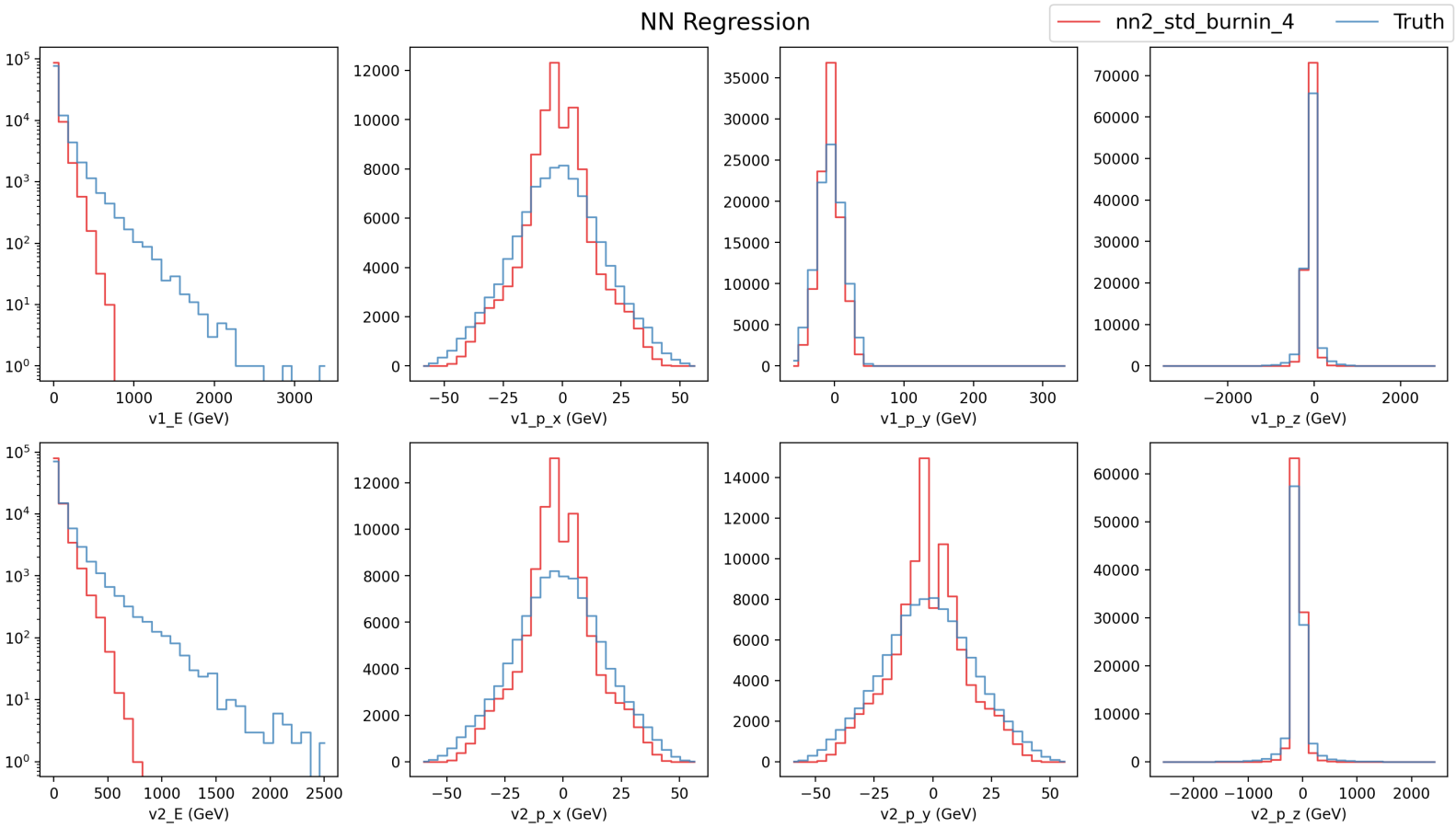

Figure C.1 Histogram of target features of neutrino (in red) predicted by the best DNN model (code name while training was '`nn2_std_burnin_4`') against true values (in blue), plotted on original dataset.

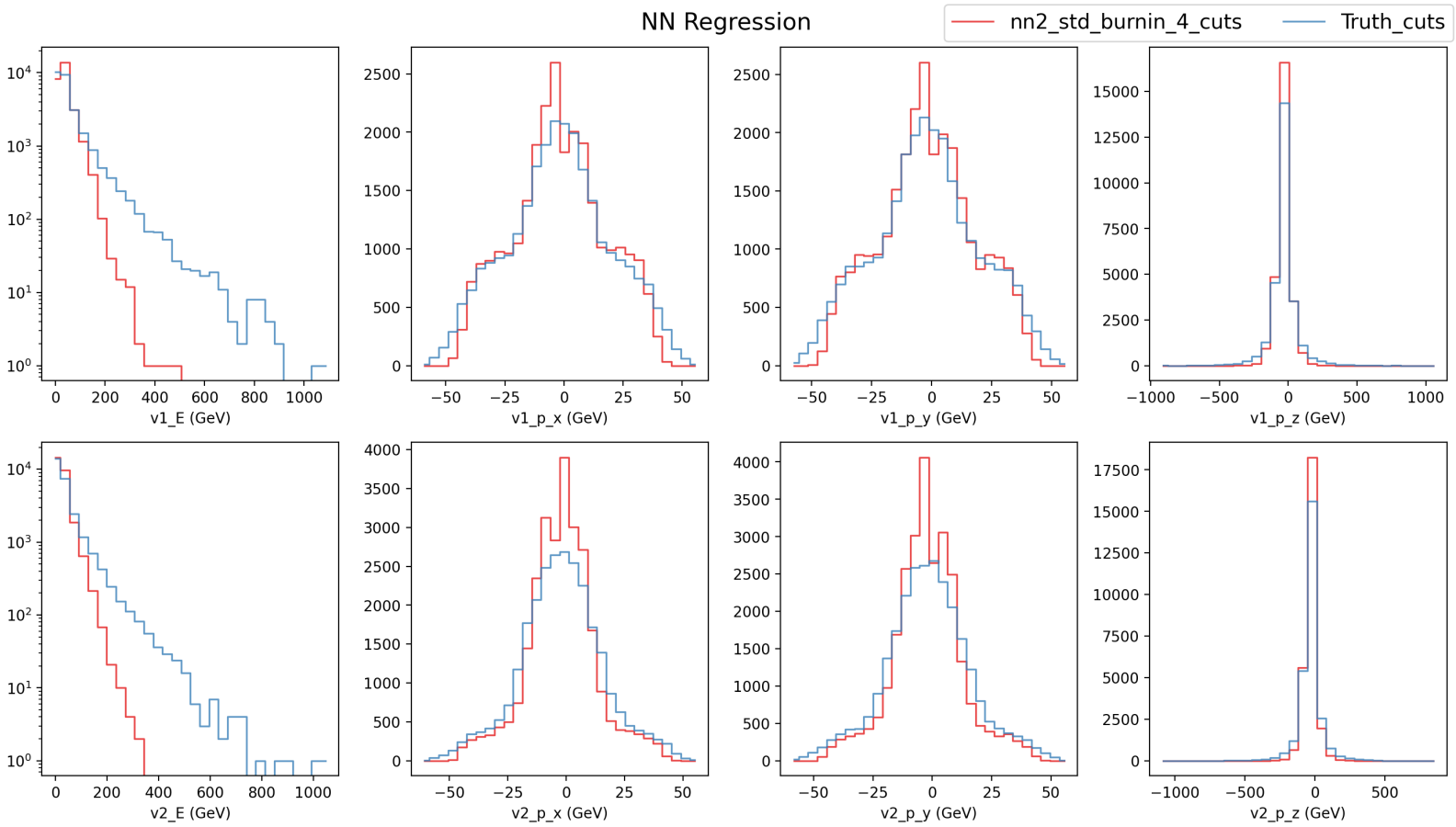

Figure C.2 Histogram of target features of neutrino (in red) predicted by the best DNN model (code name while training was '`nn2_std_burnin_4_cuts`') against true values (in blue), plotted on the dataset with selection cuts applied.

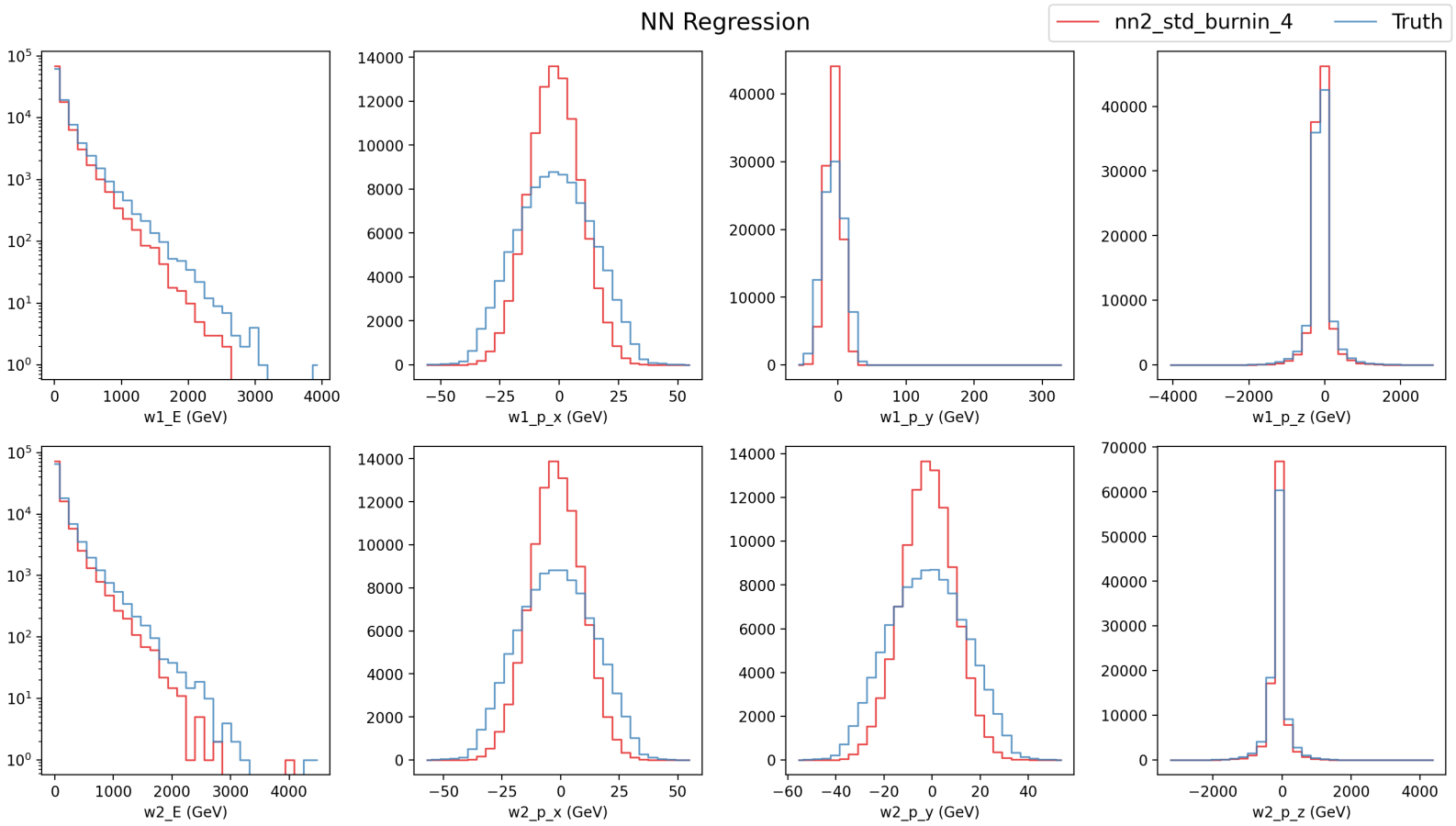

Figure C.3 Histogram of target features of W bosons (in red) predicted by the best DNN model (code name while training was '`nn2_std_burnin_4`') against true values (in blue), plotted on original dataset.

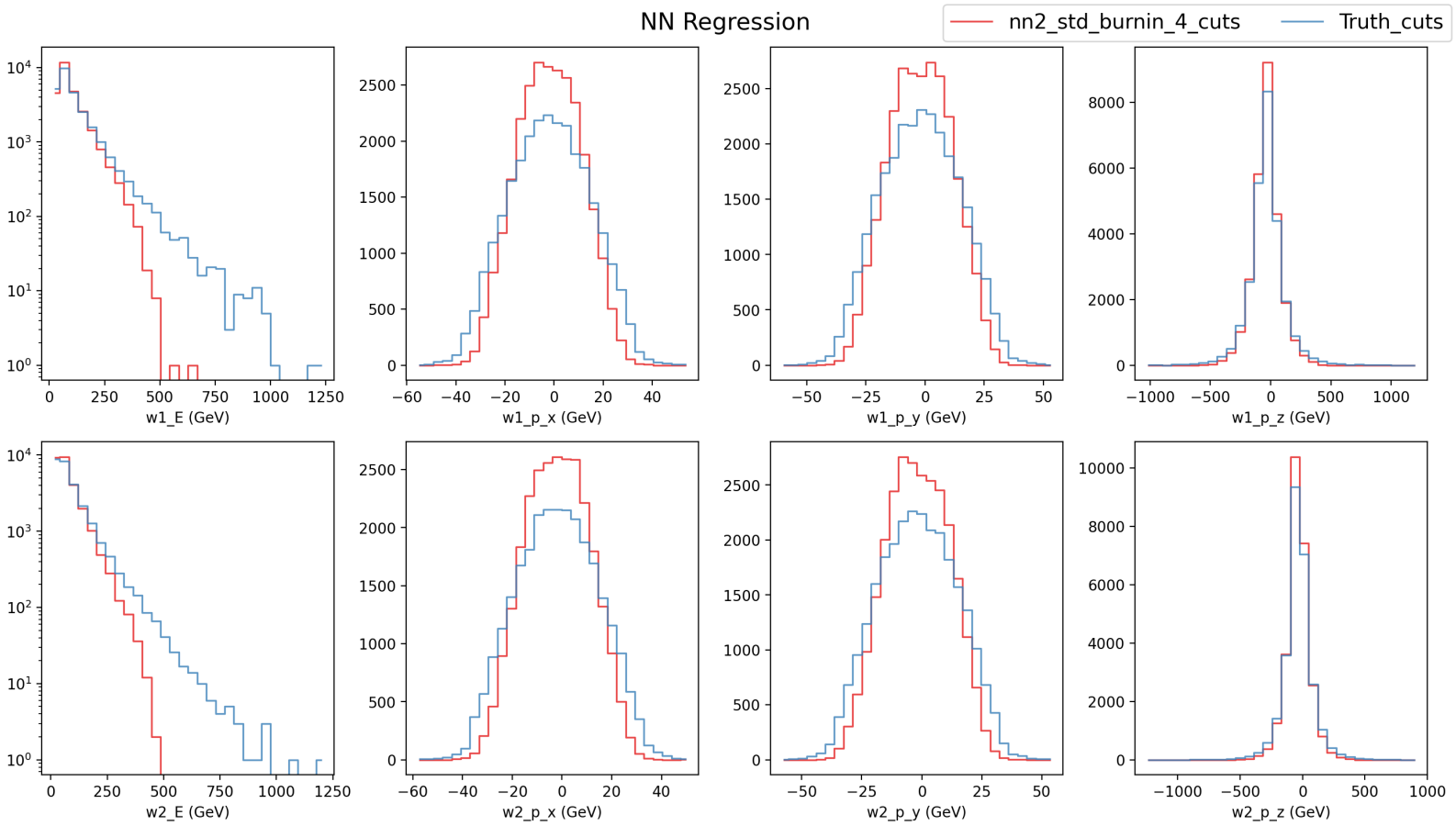

Figure C.4 Histogram of target features of W bosons (in red) predicted by the best DNN model (code name while training was '`nn2_std_burnin_4_cuts`') against true values (in blue), plotted on the dataset with selection cuts applied.

Bibliography

1. PANDEY, R.; SRIVASTAVA, N.; SINGH, N. K.; TYAGI, K. *Quantum Computing: A Shift from Bits to Qubits*. Vol. 1085 [online]. Springer, 2023. Available also from: https://doi.org/10.1007/978-981-19-9530-9. [Accessed 14-05-2025].
2. EINSTEIN, A.; PODOLSKY, B.; ROSEN, N. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.* [Online]. 1935, vol. 47, pp. 777–780. Available from DOI: 10.1103/PhysRev.47.777. [Accessed 13-04-2025].
3. BELL, J. S. On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika* [online]. 1964, vol. 1, pp. 195–200. Available from DOI: 10.1103/PhysicsPhysiqueFizika.1.195. [Accessed 13-04-2025].
4. LEE, K. C.; SPRAGUE, M. R.; SUSSMAN, B. J.; NUNN, J.; LANGFORD, N. K.; JIN, X.-M.; CHAMPION, T.; MICHELBERGER, P.; REIM, K. F.; ENGLAND, D.; JAKSCH, D.; WALMSLEY, I. A. Entangling Macroscopic Diamonds at Room Temperature. *Science* [online]. 2011, vol. 334, no. 6060, pp. 1253–1256. Available from DOI: 10.1126/science.1211914. [Accessed 13-04-2025].
5. ATLAS COLLABORATION. Observation of quantum entanglement with top quarks at the ATLAS detector. *Nature* [online]. 2024, vol. 633, no. 8030, pp. 542–547. ISSN 1476-4687. Available from DOI: 10.1038/s41586-024-07824-z. [Accessed 13-04-2025].
6. BARR, A. J. Testing Bell inequalities in Higgs boson decays. *Physics Letters B* [online]. 2022, vol. 825, p. 136866. ISSN 0370-2693. Available from DOI: https://doi.org/10.1016/j.physletb.2021.136866. [Accessed 13-04-2025].
7. HERNÁNDEZ, J. P. M. *Study of electromagnetic shower events with high granularity calorimeter prototypes and search of new physics involving heavy quark final states in future lepton colliders* [online]. 2024. Available also from: https://www.proquest.com/openview/b0f5bab2fac5b676

d762b9ce671fadb3/1?cbl=2026366&diss=y&pq-origsite=gscholar. PhD thesis. Universitat de Valencia (Spain). [Accessed 14-05-2025].

8. BURGARD, C.; GALBRAITH, D. *Standard model of physics* [online]. 2016. Available also from: https://texample.net/tikz/examples/model-physics/. [Accessed 16-04-2025].

9. PARTICLE DATA GROUP et al. Review of Particle Physics. *Progress of Theoretical and Experimental Physics* [online]. 2022, vol. 2022, no. 8, p. 083C01. ISSN 2050-3911. Available from DOI: 10.1093/ptep/ptac097. [Accessed 17-04-2025].

10. SMITH, R. E. C. *The Development of a Novel Machine Learning Algorithm for Jet Flavor Classification and a Search for Exotic Decays of the Higgs Boson Using the Atlas Detector at* $\sqrt{S}$ *= 13 TeV* [online]. 2025. Available also from: https://www.proquest.com/openview/838905fefd53ebb2cde622aa39c00a33/1?cbl=18750&diss=y&pq-origsite=gscholar. PhD thesis. Stanford University. [Accessed 14-05-2025].

11. GENG, W. *CP Violation in Single Top Quark Production* [online]. 2012. Tech. rep. Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States). Available also from: https://www.osti.gov/biblio/1128088. [Accessed 14-05-2025].

12. GLASHOW, S. L. Partial-symmetries of weak interactions. *Nuclear Physics* [online]. 1961, vol. 22, no. 4, pp. 579–588. ISSN 0029-5582. Available from DOI: https://doi.org/10.1016/0029-5582(61)90469-2. [Accessed 14-05-2025].

13. HIGGS, P. W. Broken Symmetries and the Masses of Gauge Bosons. *Phys. Rev. Lett.* [Online]. 1964, vol. 13, pp. 508–509. Available from DOI: 10.1103/PhysRevLett.13.508. [Accessed 17-04-2025].

14. ENGLERT, F.; BROUT, R. Broken Symmetry and the Mass of Gauge Vector Mesons. *Phys. Rev. Lett.* [Online]. 1964, vol. 13, pp. 321–323. Available from DOI: 10.1103/PhysRevLett.13.321. [Accessed 17-04-2025].

15. GURALNIK, G. S.; HAGEN, C. R.; KIBBLE, T. W. B. Global Conservation Laws and Massless Particles. *Phys. Rev. Lett.* [Online]. 1964, vol. 13, pp. 585–587. Available from DOI: 10.1103/PhysRevLett.13.585. [Accessed 17-04-2025].

16. ATLAS COLLABORATION et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B* [online]. 2012, vol. 716, no. 1, pp. 1–29. ISSN 0370-2693. Available from DOI: https://doi.org/10.1016/j.physletb.2012.08.020. [Accessed 17-04-2025].

17. EVANS, L.; BRYANT, P. LHC Machine. *Journal of Instrumentation* [online]. 2008, vol. 3, no. 08, S08001. Available from DOI: 10.1088/1748-0221/3/08/S08001. [Accessed 17-04-2025].

18. JAVURKOVA, M. Precision measurements of Higgs boson properties with the ATLAS experiment [online]. 2024. Available also from: https://cds.cern.ch/record/2892697. [Accessed 17-04-2025].

19. LAUDRAIN, A. *Layer Intercalibration of the ATLAS Electromagnetic Calorimeter and CP-odd Higgs Boson Couplings Measurements in the Four-Lepton Decay Channel with Run 2 Data of the LHC* [online]. 2019. Available also from: https://theses.hal.science/tel-03058335. Theses. Université Paris Saclay (COmUE). [Accessed 14-05-2025].

20. BENTVELSEN, S.; LAENEN, E.; MOTYLINSKI, P. Higgs production through gluon fusion at leading order. *NIKHEF* [online]. 2005, vol. 7, p. 2005. [Accessed 18-04-2025].

21. DJOUADI, A. The anatomy of electroweak symmetry breaking. *Physics Reports* [online]. 2008, vol. 457, no. 1-4, pp. 1–216. ISSN 0370-1573. Available from DOI: 10.1016/j.physrep.2007.10.004. [Accessed 18-04-2025].

22. MANZONI, S. *Physics with Photons Using the ATLAS Run 2 Data: Calibration and Identification, Measurement of the Higgs Boson Mass and Search for Supersymmetry in Di-Photon Final State* [online]. Springer, 2019. Available also from: https://link.springer.com/book/10.1007/978-3-030-24370-8#bibliographic-information. [Accessed 14-05-2025].

23. CUEVAS, J.; OVIEDO, U. *SM and BSM Higgs physics in CMS* [online]. 2019. Available also from: https://indico.ictp.it/event/8679/session/68/contribution/217/material/slides/0.pdf. [Accessed 18-04-2025].

24. CARENA, M. S. et. al. Higgs physics at LEP2 [online]. 1996. Available also from: https://cds.cern.ch/record/295749. [Accessed 18-04-2025].

25. ATLAS COLLABORATION. *First observation of Z-boson production via weak-boson fusion* [online]. 2014. Available also from: https://atlas.cern/updates/briefing/first-observation-z-boson-production-weak-boson-fusion. [Accessed 18-04-2025].

26. HAN, T.; VALENCIA, G.; WILLENBROCK, S. Structure-function approach to vector-boson scattering in pp collisions. *Phys. Rev. Lett.* [Online]. 1992, vol. 69, pp. 3274–3277. Available from DOI: 10.1103/PhysRevLett.69.3274. [Accessed 18-04-2025].

27. SIRUNYAN, A. M. et al. Observation of $t\bar{t}H$ Production. *Phys. Rev. Lett.* [Online]. 2018, vol. 120, p. 231801. Available from DOI: 10.1103/PhysRevLett.120.231801. [Accessed 18-04-2025].

28. AABOUD, M. et al. Observation of Higgs boson production in association with a top quark pair at the LHC with the ATLAS detector. *Physics Letters B* [online]. 2018, vol. 784, pp. 173–191. ISSN 0370-2693. Available from DOI: https://doi.org/10.1016/j.physletb.2018.07.035. [Accessed 18-04-2025].

29. CHOI, S. Y.; LEE, J. S.; PARK, J. Decays of Higgs bosons in the Standard Model and beyond. *Progress in Particle and Nuclear Physics* [online]. 2021, vol. 120, p. 103880. ISSN 0146-6410. Available from DOI: 10.1016/j.ppnp.2021.103880. [Accessed 20-04-2025].

30. DJOUADI, A. *The Standard Model of particle physics and Beyond* [online]. Clases de Master FisyMat, Desarrollos Actuales, 2021. Available also from: https://www.ugr.es/~adjouadi/Course4.pdf. [Accessed 20-04-2025].

31. TANAKA, R. *SH Higgs Branching ratios* [online]. 2013. Available also from: https://twiki.cern.ch/twiki/pub/LHCPhysics/LHCHWGCrossSectionsFigures/Higgs_BR_LM.png. [Accessed 20-04-2025].

32. TANAKA, R. *SH Higgs Branching ratios* [online]. 2013. Available also from: https://twiki.cern.ch/twiki/pub/LHCPhysics/LHCHWGCrossSectionsFigures/Higgs_BR.png. [Accessed 20-04-2025].

33. ATLAS COLLABORATION. *ATLAS measures Higgs boson mass with unprecedented precision* [online]. 2023. Available also from: https://atlas.cern/Updates/Briefing/Run2-Higgs-Mass. [Accessed 14-05-2025].

34. AABOUD, M. et al. Observation of H→bb$^-$ decays and VH production with the ATLAS detector. *Physics Letters B* [online]. 2018, vol. 786, pp. 59–86. ISSN 0370-2693. Available from DOI: https://doi.org/10.1016/j.physletb.2018.09.013. [Accessed 20-04-2025].

35. MA, X.; WU, Z.; WU, J.; HUANG, Y.; LI, G.; RUAN, M.; ALVES, F. L.; JIN, S.; SHAO, L. Measurements of decay branching fractions of the Higgs boson to hadronic final states at the CEPC*. *Chinese Physics C* [online]. 2025, vol. 49, no. 5, p. 053001. Available from DOI: 10.1088/1674-1137/adacc5. [Accessed 20-04-2025].

36. THOMSON, M. *Modern particle physics* [online]. Cambridge University Press, 2013. Available also from: https://books.google.cz/books?hl=en&lr=&id=BV1sAAAAQBAJ&oi=fnd&pg=PR13&dq=Modern+Particle+Physics&ots=Er0t_0Vl5A&sig=FsUdUB9eagFuI6KWBXpqwmzVlTY&redir_esc=y#v=onepage&q=Modern%20Particle%20Physics&f=false. [Accessed 14-05-2025].

37. SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* [online]. 1959, vol. 3, no. 3, pp. 210–229. Available from DOI: 10.1147/rd.33.0210. [Accessed 28-04-2025].

38. M., Mehryar; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of machine learning* [online]. The MIT Press, Cambridge, 2018. Available also from: https://cs.nyu.edu/~mohri/mlbook/. [Accessed 07-05-2025].

39. MORIMOTO, J.; PONTON, F. Virtual reality in biology: could we become virtual naturalists? *Evolution: Education and Outreach* [online]. 2021, vol. 14. Available from DOI: 10.1186/s12052-021-00147-x. [Accessed 07-05-2025].

40. ZHANG, X.-D. *A Matrix Algebra Approach to Artificial Intelligence* [online]. 1st ed. Springer Singapore, 2020. ISBN 978-981-15-2770-8. Available from DOI: https://doi.org/10.1007/978-981-15-2770-8. [Accessed 07-05-2025].

41. ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd* [online]. 1996, vol. 96, pp. 226–231. No. 34. [Accessed 08-05-2025].

42. HANDL, J.; KNOWLES, J.; KELL, D. B. Computational cluster validation in post-genomic data analysis. *Bioinformatics* [online]. 2005, vol. 21, no. 15, pp. 3201–3212. ISSN 1367-4803. Available from DOI: 10.1093/bioinformatics/bti517. [Accessed 08-05-2025].

43. DUNN, J. C. Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics* [online]. 1974, vol. 4, no. 1, pp. 95–104. Available from DOI: 10.1080/01969727408546059. [Accessed 08-05-2025].

44. VAŠATA, D.; KLOUDA, K. *Supervizované učení, klasifikační úloha, rozhodovací stromy* [BI-ML1.21 lecture 2] [online]. 2024. Available also from: https://courses.fit.cvut.cz/BI-ML1/@master/lectures/files/BI-ML1-02-cs-slides.pdf. [Accessed 08-05-2025].

45. BILMES, J. Underfitting and overfitting in machine learning. *UW ECE course notes* [online]. 2020, vol. 5. Available also from: https://people.ece.uw.edu/bilmes/classes/ee511/ee511_spring_2020/overfitting_underfitting.pdf. [Accessed 09-05-2025].

46. YING, X. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series* [online]. 2019, vol. 1168, no. 2, p. 022022. Available from DOI: 10.1088/1742-6596/1168/2/022022. [Accessed 09-05-2025].

47. BADILLO, S.; BANFAI, B.; BIRZELE, F.; DAVYDOV, I. I.; HUTCHINSON, L.; KAM-THONG, T.; SIEBOURG-POLSTER, J.; STEIERT, B.; ZHANG, J. D. An Introduction to Machine Learning. *Clinical Pharmacology & Therapeutics* [online]. 2020, vol. 107, no. 4, pp. 871–885. Available from DOI: https://doi.org/10.1002/cpt.1796. [Accessed 09-05-2025].

48. KOCHENDERFER, M. J.; WHEELER, T. A. *Algorithms for optimization* [online]. Mit Press, 2019. Available also from: https://books.google.cz/books?hl=en&lr=&id=oxyNDwAAQBAJ&oi=fnd&pg=PR7&dq=Algorithms+for+Optimization+(Mit+Press)&ots=hxhLs-q7uy&sig=NAR3g3FiB_NwkkvT3_FZoOKu_lg&redir_esc=y#v=onepage&q=Algorithms%20for%20Optimization%20(Mit%20Press)&f=false. [Accessed 09-05-2025].

49. YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* [online]. 2020, vol. 415, pp. 295–316. ISSN 0925-2312. Available from DOI: https://doi.org/10.1016/j.neucom.2020.07.061. [Accessed 09-05-2025].

50. P. CARVALHO, Halcyon D; C. A. LIMA, Marília N; SANTOS, Wylliams B; A.FAGUNDE, Roberta A de. Ensemble Regression Models for Software Development Effort Estimation: A Comparative Study. *International Journal of Software Engineering & Applications* [online]. 2020, vol. 11, no. 3. ISSN 0976-2221. Available from DOI: 10.5121/ijsea.2020.11305. [Accessed 09-05-2025].

51. MIENYE, I. D.; SUN, Y. A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access* [online]. 2022, vol. 10. Available from DOI: 10.1109/ACCESS.2022.3207287. [Accessed 09-05-2025].

52. BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.A.; STONE, C.J. *Classification and Regression Trees* [online]. 1st ed. Chapman and Hall/CRC, 1984. Available also from: https://doi.org/10.1201/9781315139470. [Accessed 09-05-2025].

53. PŁOŃSKI, P. *Visualize a Decision Tree in 5 Ways with Scikit-Learn and Python* [online]. 2020. Available also from: https://mljar.com/blog/visualize-decision-tree/. [Accessed 10-05-2025].

54. LAURENT, H.; RIVEST, R. L. Constructing optimal binary decision trees is NP-complete. *Information processing letters* [online]. 1976, vol. 5, no. 1, pp. 15–17. Available also from: https://people.csail.mit.edu/rivest/pubs/HR76.pdf. [Accessed 10-05-2025].

55. DEVELOPERS, scikit-learn. *Decision Trees* [online]. 2025. Available also from: https://scikit-learn.org/stable/modules/tree.html. [Accessed 10-05-2025].

56. SCHAPIRE, R. E. The Boosting Approach to Machine Learning: An Overview. In: *Nonlinear Estimation and Classification* [online]. Ed. by DENISON, D. D.; HANSEN, M. H.; HOLMES, C. C.; MALLICK, B.; YU, B. New York, NY: Springer New York, 2003, pp. 149–171. ISBN 978-0-387-21579-2. Available from DOI: 10.1007/978-0-387-21579-2_9. [Accessed 10-05-2025].

57. DRUCKER, H. Improving Regressors Using Boosting Techniques. *Proceedings of the 14th International Conference on Machine Learning* [online]. 1997. Available also from: https://www.researchgate.net/publication/2424244_Improving_Regressors_Using_Boosting_Techniques. [Accessed 10-05-2025].

58. FREUND, Y.; SCHAPIRE, R. E., et al. Experiments with a new boosting algorithm. In: *icml* [online]. Citeseer, 1996, vol. 96, pp. 148–156. Available also from: https://cseweb.ucsd.edu/~yfreund/papers/boostingexperiments.pdf#:~:text=%E2%80%9CBoosting%E2%80%9D%20is%20a%20general%20method,we%20present%20such%20an%20experimental. [Accessed 10-05-2025].

59. DRUCKER, H. Improving Regressors using Boosting Techniques. In: *International Conference on Machine Learning* [online]. 1997. Available also from: https://api.semanticscholar.org/CorpusID:16242966. [Accessed 10-05-2025].

60. MCCULLOCH, W.S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* [online]. 1943, vol. 5, pp. 115–133. Available also from: https://doi.org/10.1007/BF02478259. [Accessed 12-05-2025].

61. ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* [online]. 1958, pp. 386–408. Available also from: https://doi.org/10.1037/h0042519. [Accessed 12-05-2025].

62. SZE, V.; CHEN, Y.-H.; YANG, T.-J.; EMER, J. S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE* [online]. 2017, vol. 105, no. 12, pp. 2295–2329. Available from DOI: 10.1109/JPROC.2017.2761740. [Accessed 12-05-2025].

63. BISHOP, C. M. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation* [online]. 1995, vol. 7, no. 1, pp. 108–116. Available from DOI: 10.1162/neco.1995.7.1.108. [Accessed 12-05-2025].

64. ZUR, R.; JIANG, Y.; PESCE, L.; DRUKKER, K. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical physics* [online]. 2009, vol. 36, pp. 4810–8. Available from DOI: 10.1118/1.3213517. [Accessed 12-05-2025].

65. GOYAL, P.; DOLLÁR, P.; GIRSHICK, R.; NOORDHUIS, P.; WESOLOWSKI, L.; KYROLA, A.; TULLOCH, A.; JIA, Y.; HE, K. *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour* [online]. 2018. Available from arXiv: 1706.02677 [cs.CV]. [Accessed 12-05-2025].

66. LOSHCHILOV, I.; HUTTER, F. *SGDR: Stochastic Gradient Descent with Warm Restarts* [online]. 2017. Available from arXiv: 1608.03983 [cs.LG]. [Accessed 12-05-2025].

67. TAMBAD, S.; NANDWANI, R.; MCINTOSH, S. K. Analyzing programming languages by community characteristics on Github and StackOverflow [online]. 2020. Available from arXiv: 2006.01351 [cs.SE]. [Accessed 21-04-2025].

68. AGUILAR-SAAVEDRA, J. A. Laboratory-frame tests of quantum entanglement in $H \to WW$. *Physical Review D* [online]. 2023, vol. 107, no. 7. ISSN 2470-0029. Available from DOI: 10.1103/physrevd.107.076016. [Accessed 27-04-2025].

69. ALWALL, J.; FREDERIX, R.; FRIXIONE, S.; HIRSCHI, V.; MALTONI, F.; MATTELAER, O.; SHAO, H.-S.; STELZER, T.; TORRIELLI, P.; ZARO, M. 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* [online]. 2014, vol. 2014, no. 7. ISSN 1029-8479. Available from DOI: 10.1007/jhep07(2014)079. [Accessed 27-04-2025].

70. SJÖSTRAND, T.; ASK, S.; CHRISTIANSEN, J. R.; CORKE, R.; DESAI, N.; ILTEN, P.; MRENNA, S.; PRESTEL, S.; RASMUSSEN, C. O.; SKANDS, P. Z. An introduction to PYTHIA 8.2. *Computer Physics Communications* [online]. 2015, vol. 191, pp. 159–177. ISSN 0010-4655. Available from DOI: https://doi.org/10.1016/j.cpc.2015.01.024. [Accessed 27-04-2025].

71. FAVEREAU, J. de; DELAERE, C.; DEMIN, P.; GIAMMANCO, A.; LEMAÎTRE, V.; MERTENS, A.; SELVAGGI, M. DELPHES 3: a modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics* [online]. 2014, vol. 2014, no. 2. ISSN 1029-8479. Available from DOI: 10.1007/jhep02(2014)057. [Accessed 27-04-2025].

72. ATLAS. *Measurements of the properties of the Higgs-like boson in the* $WW^{(*)} \to \ell\nu\ell\nu$ *decay channel with the ATLAS detector using 25* fb^{-1} *of proton-proton collision data* [online]. Geneva, 2013. Tech. rep. CERN. Available also from: https://cds.cern.ch/record/1527126. [Accessed 27-04-2025].

73. DEVELOPERS, scikit-learn. *StandardScaler* [online]. 2025. Available also from: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. [Accessed 14-05-2025].

74. DEVELOPERS, scikit-learn. *RobustScaler* [online]. 2025. Available also from: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html. [Accessed 14-05-2025].

75. PAZOS, C.; AERON, S.; BEAUCHEMIN, P.-H.; CROFT, V.; HUAN, Z.; KLASSEN, M.; WONGJIRAD, T. Towards Universal Unfolding of Detector Effects in High-Energy Physics using Denoising Diffusion Probabilistic Models. *Methods* [online]. 2024, vol. 4, p. 4. Available also from: https://arxiv.org/abs/2406.01507. [Accessed 28-04-2025].

76. ANDREASSEN, A.; KOMISKE, P. T.; METODIEV, E. M.; NACHMAN, B.; THALER, J. OmniFold: A Method to Simultaneously Unfold All Observables. *Phys. Rev. Lett.* [Online]. 2020, vol. 124, p. 182001. Available from DOI: 10.1103/PhysRevLett.124.182001. [Accessed 28-04-2025].

Contents of the attachment

```
/
├── README.md ..... a brief description of the content
├── src
│   └── Higgs_ww_qe ..... A directory with source codes and plots
└── thesis ..... text and code of the thesis
    ├── vakandri-ctufit-thesis.pdf ..... PDF with thesis text
    └── thesis.zip ..... LaTeX source work with code and images
```