GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

Fakultät für Physik

Bachelor's Thesis

Behandlung von Elektronen im Jet in der "boosted" $X \to HH \to b\bar{b}WW^*$ 1-Lepton-Analyse unter Verwendung neuronaler Netze am ATLAS Experiment

Electron in jet treatment in the boosted $X \to HH \to b\bar{b}WW^*$ 1-lepton analysis using Neural Networks with the ATLAS Experiment

prepared by

Josephine Katharina Stanitzok

from Wolfsburg

at the II. Physikalischen Institut

Thesis number: II.Physik-UniGö-BSc-2024/05

Thesis period: 3rd June 2024 until 26th August 2024

First referee: Prof. Dr. Stan Lai

Second referee: Prof. Dr. Steffen Schumann

Abstract

Diese Arbeit behandelt die Herausforderung der Identifizierung von Elektronen, die mit Jets überlappen. Kontext ist die Suche nach resonanter Higgs-Boson-Paarproduktion am ATLAS Experiment, wobei der "boosted" $X \to HH \to b\bar{b}WW^*$ 1-Lepton-Kanal betrachtet wird.
Ein neuronales Netzwerk mit sechs Hidden Layers, ReLU Aktivierung, L2 Regularisierung und Batch Normalisierung wurde entwickelt, um die Identifizierungsgenauigkeit in von Elektronen in Jets zu verbessern. Das Modell wurde mit dem Adam-Optimierer und der categorical Cross-Entropy Verlustfunktion trainiert und nutzte EarlyStopping und Mechanismen zur Reduzierung der Lernrate.
Das trainierte Netzwerk erreichte eine hohe Genauigkeit von 0,9718 und einen Verlust von 0,1353 nach 54 Epochen, mit einer AUC von 0,9918 für sowohl Signal- als auch Hintergrundklassen. Die Konfusionsmatrix zeigte eine hohe Genauigkeit bei der Unterscheidung von Signal- (99,98%) und Hintergrundklassen (94,38%), was die Robustheit des Modells bei der Unterscheidung von Jets, die prompte Elektronen enthalten, und anderen Jets und Leptonen bestätigt.

Stichwörter: Higgs-Boson, resonante Higgs-Paarproduktion, HHbbVV, Neuronale Netze

Abstract

This work addresses the challenge of identifying electrons that overlap with jets. The context is the search for resonant Higgs boson pair production at the ATLAS experiment, where the boosted $X \to HH \to b\bar{b}WW^*$ 1-lepton channel is considered.
A neural network with six hidden layers, ReLU activation, L2 regularisation and batch normalisation was developed to improve the identification accuracy of electrons in jets. The model was trained with the Adam optimiser and the categorical cross-entropy loss function and used early stopping and learning rate reduction mechanisms.
The trained network achieved a high accuracy of 0.9718 and a loss of 0.1353 after 54 epochs, with an AUC of 0.9918 for both signal and background classes. The confusion matrix showed high accuracy in discriminating signal (99.98%) and background classes (94.38%), confirming the robustness of the model in discriminating jets containing prompt electrons and other jets and leptons.

Keywords: Higgs boson, resonant Higgs pair production, HHbbVV, Neural Networks

Contents

1. Introduction

Understanding the universe has always been a profound quest for humanity, particularly for physicists. The Standard Model (SM) of particle physics [1–5] represents a major milestone in this pursuit, offering a comprehensive framework that clarifies the interactions and properties of elementary particles.

The discovery of the Higgs boson in 2012 by the ATLAS and CMS experiments at the Large Hadron Collider (LHC) [6, 7] was a groundbreaking achievement, affirming the validity of the SM. In the realm of particle physics, the Higgs boson is not just another particle; it is a cornerstone of our understanding of the universe. The Higgs boson is intimately connected with the Higgs field, an energy field that pervades the entire universe [8]. As particles interact with this field, they acquire mass [9]. Without the Higgs field and the Higgs boson, particles would move at the speed of light, making the formation of stars, planets, and life as we know it impossible.

A promising method for testing the SM description of Higgs physics is through the study of Higgs boson pair production. While the SM predicts non-resonant pair production, theories beyond the SM (BSM) also predict resonant production, where a heavy scalar particle X decays into two Higgs bosons. This type of resonant production serves as a direct probe for new physics. Since the SM does not include any particles heavy enough to decay into two Higgs bosons, observing such resonant production would signal the presence of new particles or interactions [10].

At very high resonant masses ($m_X \gtrsim 2\,\text{TeV}$), the topology of the Higgs pair production becomes boosted, resulting in collimated decay products. One possible final state is the $b\bar{b}WW^*$. In these scenarios, the decay products of the Higgs boson ($H \to b\bar{b}$) and the hadronically decaying W boson (W_{had}) are often reconstructed as a single jet due to the limitations of detector resolution. Additionally, the lepton from the leptonically decaying W boson may overlap with the W_{had} jet, complicating the reconstruction process [11]. In high-energy environments, it is particularly difficult to distinguish individual particles. This challenge is especially pronounced for electrons compared to muons, which benefit from the muon spectrometer designed to track them more accurately.

This thesis focuses on the challenge of identifying electrons that overlap with jets in the search for boosted $X \to HH \to b\bar{b}WW^*$ in the 1-lepton channel at the ATLAS experiment. The study employs neural networks to enhance both the precision and effectiveness of the identification process.

The structure of the thesis is as follows: Chapter 2 discusses the fundamentals of particle physics, including the Standard Model, the Higgs mechanism, and the properties of the Higgs boson. Chapter 3 describes the Large Hadron Collider and the ATLAS detector. Chapter 4 provides an overview of neural networks, detailing various types, training techniques, and optimization methods. Chapter 5 focuses on the identification of electrons in jets in the boosted $X \to HH \to b\bar{b}WW^*$ 1-lepton analyses, addressing object reconstruction and the application of neural networks for classification. Finally, Chapter 6 presents the conclusion and an outlook on future research.

2. Fundamentals of Particle Physics

2.1. The Standard Model

The Standard Model (SM) of particle physics provides a comprehensive framework for understanding the fundamental particles and their interactions [12]. It categorizes elementary particles into fermions and bosons, each with specific roles and characteristics. Figure 2.1 illustrates the arrangement and classification of these particles within the model.
Twelve of the particles in the SM are fermions. They have a spin of $\frac{1}{2}$ and build up matter. They are further characterized by their mass, charge, colour and weak isospin. Depending on these characteristics, they couple electromagnetically, weakly, strongly and to the Higgs field. They are divided into three generations with equal charge, colour and isospin but with different masses. Each generation contains quarks and leptons. Quarks are either up-type (charge of $+\frac{2}{3}e$) or down-type (charge of $-\frac{1}{3}e$) and carry colour charge, necessary to interact strongly.
Leptons do not interact via the strong force and have an integer charge. They are either charged ($\pm 1e$) or neutral (neutrinos). The first generation of leptons are the electron (e) and the electron neutrino (ν_e). The second generation consists of the muon (μ) and the muon neutrino (ν_μ). The third generation consists of the tau (τ) and the tau neutrino (ν_τ). Leptons become heavier with each generation as well. Neutrinos are basically massless.
Gauge bosons (vector bosons) are the last group of particles of the SM. They mediate the fundamental forces. The four known forces of nature are the strong force, the electromagnetic force, the weak force and gravity. However, gravity has no effect in particle physics because it is too weak. There are five gauge bosons in total. The massless gauge boson for the strong force is the gluon (g), and for the electromagnetic force it is the photon (γ). The weak force has three gauge bosons: two charged bosons (W^+/W^-) with a mass of about 80.4 GeV and a neutral Z^0 boson with a mass of 91.2 GeV [12].
The final boson in the SM is the scalar Higgs boson. More detailed information can be found in Sections 2.3 and 2.4.

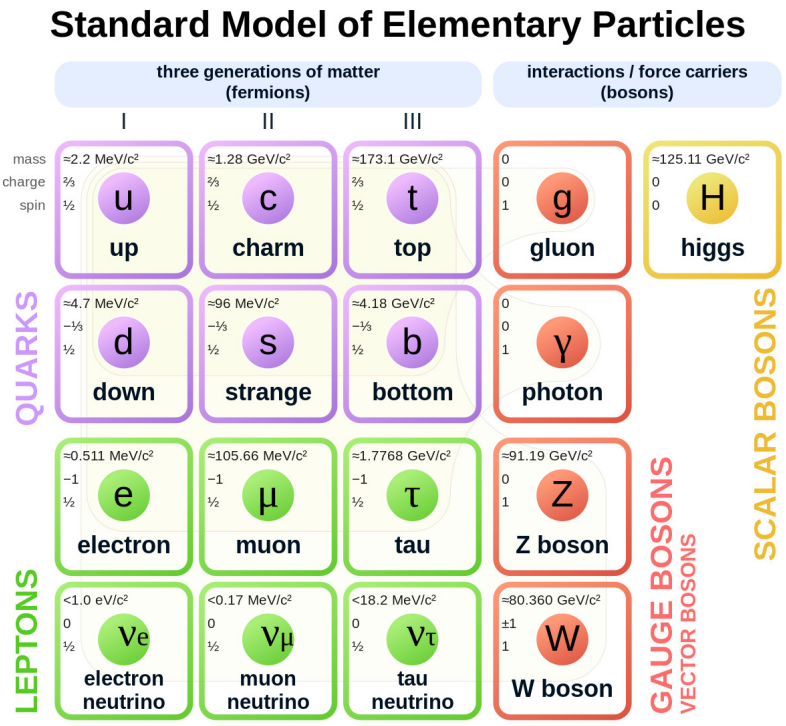


Figure 2.1.: Standard Model of Elemantary Particles with fermions (quarks and leptons) and bosons [13]

The SM does not explain several crucial phenomena, including gravity, dark matter and dark energy [14]. It also fails to explain the observed matter-antimatter asymmetry in the universe [15] and the existence of neutrino masses [16]. These and other limitations suggest the presence of new physics beyond the SM (BSM) [17].

2.2. The Higgs mechanism

The Higgs mechanism, first described by Peter Higgs in 1964, François Englert and Robert Brout in 1964, and independently by Tom W. B. Kibble, Carl R. Hagen, and Gerald Guralnik in 1968 [8, 18, 19], explains how particles acquire mass while preserving gauge symmetries. Generally, mass terms in the Lagrangian conflict with gauge invariance, necessitating the introduction of a field known as the Higgs field. This field permeates all of space and has a non-vanishing vacuum expectation value [8]. Massive particles interact with the Higgs field, which imparts mass to them. The extent of this interaction determines the mass of the particle: particles that interact strongly with the Higgs field have higher masses, while those that do not, such as the photon, remain massless. The Higgs field is represented by a potential function, which is given by Formula 2.1 and

illustrated in Figure 2.2 [8, 18, 19].

$$V(\phi) = \mu^2(\phi^\dagger\phi) + \lambda(\phi^\dagger\phi)^2 \tag{2.1}$$

with

$$\phi = \frac{1}{\sqrt{2}}\begin{pmatrix}\phi_1 + i\phi_2 \\ \phi_3 + i\phi_4\end{pmatrix} \tag{2.2}$$

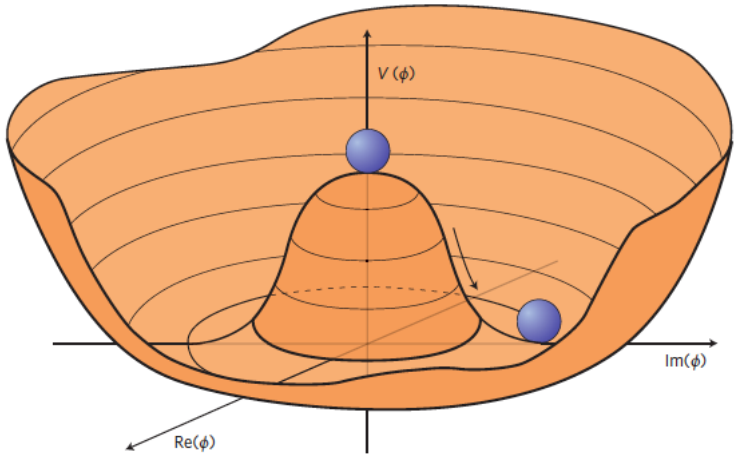


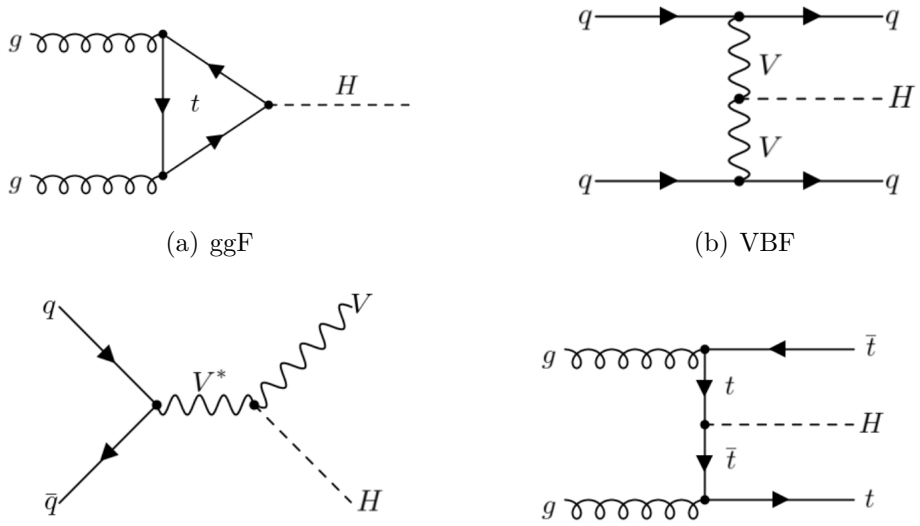
Figure 2.2.: $V(\phi)$ potential for a complex scalar field with $\mu^2 < 0$ and $\lambda > 0$ [20]

After electroweak symmetry breaking, three degrees of freedom are absorbed by the masses of the W and Z bosons, while the photon remains massless, thereby preserving electromagnetic symmetry. Additionally, it explains the mass of fermions, such as electrons, through interactions described by Yukawa coupling. The presence of the Higgs boson is a consequence of the Higgs mechanism, as the Higgs field itself cannot be measured directly.

2.3. Higgs boson properties

The Higgs boson was discovered in 2012 by the ATLAS and CMS experiments at the Large Hadron Collider (LHC) at CERN [6, 7]. It has a spin of 0 and is one of the heaviest particles with a mass of $m_H = 125.09 \pm 0.21(stat) \pm 0.11(syst)$ GeV [21].
The Higgs boson can be produced through several distinct mechanisms at the LHC. These include gluon-gluon fusion (ggF), vector boson fusion (VBF), vector boson to Higgs ($V^* \rightarrow VH$) associated production, and top-antitop pair production associated with

a Higgs boson ($t\bar{t}H$). The lowest-order Feynman diagrams for each of these production processes are illustrated in Figure 2.3.
The production cross sections for pp collisions at a center-of-mass energy of $\sqrt{s} = 13$ TeV are shown in Figure 2.4(a) as a function of the Higgs boson mass m_H. The Higgs boson decays into different particles due to its instability. The branching ratios of the Higgs boson are depicted in Figure 2.4(b) as a function of m_H. Both figures illustrate how these quantities vary with m_H.
The most frequent production mechanism involves the creation of a Higgs boson due to a ggF. The next most common production cross section, which is approximately an order of magnitude lower, is VBF. Once produced, the Higgs boson decays rapidly due to its large mass. The branching ratios indicate that the most dominant decay modes are $H \to b\bar{b}$ and $H \to WW$, followed by $H \to gg$ and $H \to \tau\tau$.



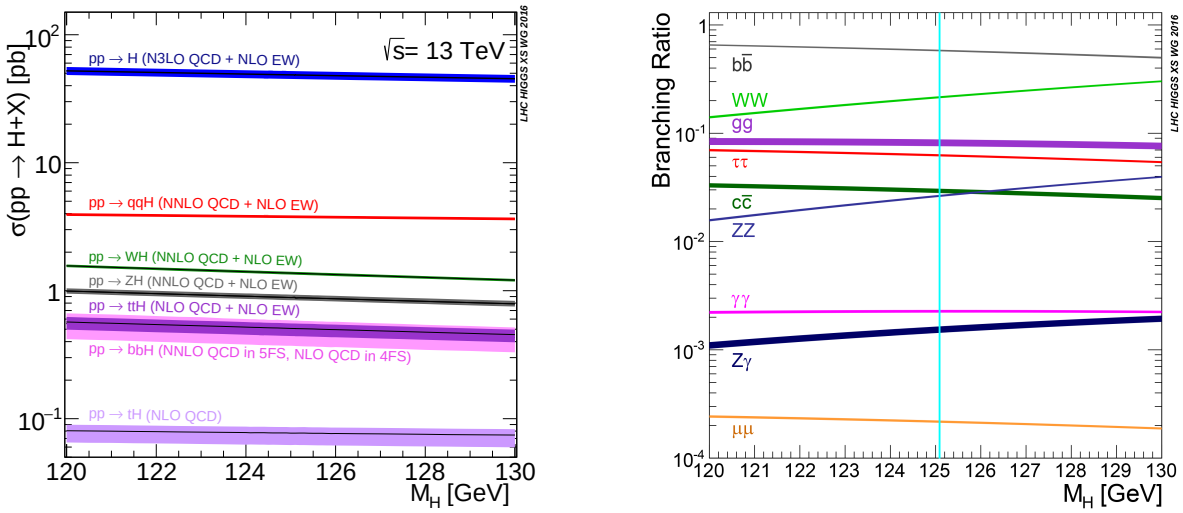
(a) ggF (b) VBF

(c) VH (d) $t\bar{t}H$

Figure 2.3.: Lowest order Feynman diagrams for the production of Higgs bosons at the LHC. (a) gluon-gluon fusion (ggF), (b) vector boson fusion (VBF), (c) VH production, and (d) $t\bar{t}H$ production [11]

2.4. Higgs boson pair production

In addition to single Higgs boson production shown in Figure 2.3, pairs of Higgs bosons can be produced at the LHC. In the SM, such pair production can only occur in a non-resonant mode. However, BSM theories allow for Higgs boson pairs to be produced resonantly by introducing a heavy particle X that can decay into Higgs bosons. Such a heavy scalar is introduced for example in two-Higgs-doublet models, where a second



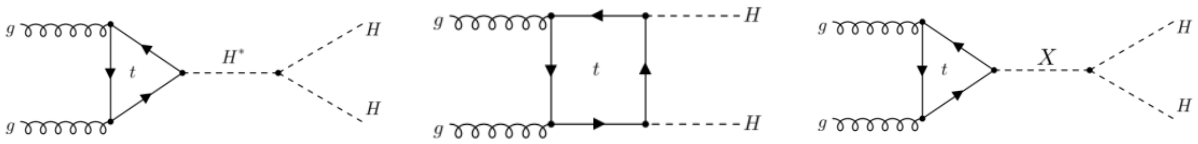
(a) Production cross section

(b) Branching ratio of the Higgs boson

Figure 2.4.: LHC production cross section for pp collisions at a center-of-mass energy of $\sqrt{s} = 13\,\text{TeV}$, measured in pb, and the branching ratio of the Higgs boson, both as functions of the Higgs boson mass. The precision of the calculations is indicated in parentheses [22].

scalar doublet is added to the SM [23] and the two real singlet model (TRSM), where new scalar bosons like X and S can be produced [24]. These models do not specify the masses of these new particles, so searches must explore a wide mass range.

Higgs boson pair production can occur through several mechanisms: self-coupling of Higgs bosons or interactions with the top quark Yukawa coupling as predicted in the SM, within BSM frameworks, it can occur through the mediation of a new heavy scalar particle X. Figure 2.5 illustrates the Feynman diagrams corresponding to these mechanisms of Higgs pair production.



(a) Higgs self-coupling (SM)

(b) top Yukawa-coupling (SM)

(c) resonant Higgs boson pair production via heavy scalar X (BSM)

Figure 2.5.: Feynman diagrams of Higgs pair production [11]

Figure 2.6 shows the branching ratios for the decay of two Higgs bosons. The most common decay modes are either both Higgs bosons decaying into two bottom quarks each (34%) or one Higgs boson decaying into two bottom quarks and the other into two W bosons (25%). The W bosons themselves can decay in various ways, with the most frequent mode being into two quarks, although they can also decay into a lepton and its neutrino.
For this thesis only the $b\bar{b}WW^*$ decay channel is relevant.

	bb	WW	ττ	ZZ	γγ
bb	34%				
WW	25%	4.6%			
ττ	7.3%	2.7%	0.39%		
ZZ	3.1%	1.1%	0.33%	0.07%	
γγ	0.26%	0.1%	0.02%	0.01%	< 0.001%

Figure 2.6.: Branching ratios of a Higgs boson pair with $m_H = 125\,\text{GeV}$ at the LHC [22]

3. The Large Hadron Collider and Atlas Detector

This chapter gives an overview of the Large Hadron Collider in Section 3.1 and describes the Atlas Detector in Section 3.2.

3.1. The Large Hadron Collider

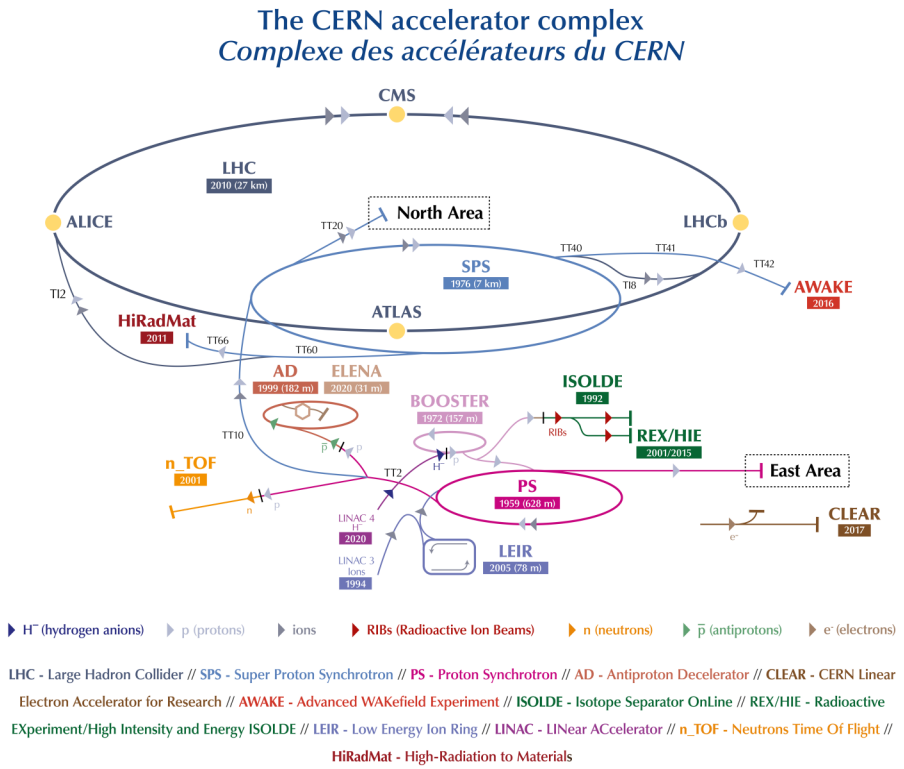


Figure 3.1.: Schematic of the full accelerator complex of the Cern [25]

The Large Hadron Collider, also known as Lhc, is a circular synchrotron accelerator with a circumference of 27 km. It is based at the European Organisation for Nuclear Research

(CERN) in Geneva, Switzerland [26]. The LHC is designed as a proton-proton as well as a heavy ion collider, with a current pp-collision centre-of-mass energy of 13.6 TeV. It is used to study particles of the Standard Model (SM) and to search for particles and interactions beyond the SM. It was built in the old Large Electron-Positron collider (LEP) tunnel, making it the world's largest collider.

To achieve such high energies, the protons are subjected to a series of acceleration stages (shown in Figure 3.1). The process commences with the Linear Accelerator 4 (LINAC4), where negative hydrogen ions are accelerated to about 160 MeV [27]. These ions are then transferred to the Proton Synchrotron Booster (PSB), which strips away the electrons, leaving only the protons. These protons are then further accelerated to 2 GeV and passed to the Proton Synchrotron (PS). In the PS, protons achieve energies of 26 GeV before being injected into the Super Proton Synchrotron (SPS), which propels the protons to energies of 250 GeV. Subsequently, the protons are transferred to the LHC to reach their final energy. At this stage, the protons are travelling at velocities close to the speed of light.

The LHC accelerates two proton beams in opposite directions. The dipole magnets, which are used to guide the protons along a circular path, require field strengths of $\mathrm{B} = 8\,\mathrm{T}$. To achieve this, the magnets are superconducting and must be cooled down to 2 K. The beams are focused using quadrupole and higher order multipole magnets. The protons traverse the accelerator in bunches, with each bunch containing 1.15×10^{11} protons. Bunches follow one another at time intervals of 25 ns. Collisions occur at a rate of 40 MHz. Each time the two beams intersect, numerous interactions take place. For Run-3, there are, on average, 46.5 interactions per bunch crossing.

The luminosity delivered by the LHC can be determined from the frequency of bunch crossings f, the particle number per bunch in both beams N_1 N_2, the number of bunches n_b, and other variables such as the transverse beam dimension $\sigma_{x/y,i}$ and a reduction factor S:

$$\mathcal{L} = \frac{N_1 N_2 n_b f S}{2\pi\sqrt{\sigma_{x,1}^2 + \sigma_{x,2}^2}\sqrt{\sigma_{y,1}^2 + \sigma_{y,2}^2}}. \tag{3.1}$$

During operation, the proton beams lose protons due to collisions and beam losses. After approximately 15 h, the beams are discarded and new beams are generated.

The collision data of the LHC is collected at four main experiments: ATLAS [28], CMS [29], ALICE [30] and LHCb [31]. ATLAS and CMS are the two largest experiments with multipurpose detectors focusing on precise measurements and the search for Beyond Standard Model (BSM) physics. ALICE and LHCb are more specialized. ALICE focuses on lead-lead collisions, simulating the state of the universe shortly after the Big Bang while

LHCb specializes in b-physics.
The LHC does not operate continuously. There are long shutdown periods between run periods during which the machine undergoes maintenance and upgrades are installed. The first run of LHC (Run 1) happened between 2010 and 2013 with a centre-of-mass energy between $\sqrt{s} = 7$ and 8 TeV. For Run 2 (2015-2018), the LHC was upgraded and a energy of $\sqrt{s} = 13$ TeV was reached. After Run 2 the LHC was upgraded again. The LHC Run 3 started again with $\sqrt{s} = 13.6$ TeV in 2022 and is supposed to end in 2025. After another upgrade phase, the High-Luminosity Large Hadron Collider (HL-LHC) will be operated, supposingly delivering an integrated luminosity of 3000 fb^{-1} in Run 4 [32].
This thesis uses data from Run 3.

3.2. The Atlas Detector

The ATLAS (A Toroidal LHC ApparatuS) detector is the largest particle detector at the LHC with a length of 44 m and a diameter of 25 m [28]. It weighs 7000 t and has a cylindrical shape with the beam axis corresponding to its symmetry axis. The detector covers almost the full solid angle and is a general-purpose detector, comprising several detector layers each with different tasks. Figure 3.2 shows a schematic illustration of the ATLAS detector.

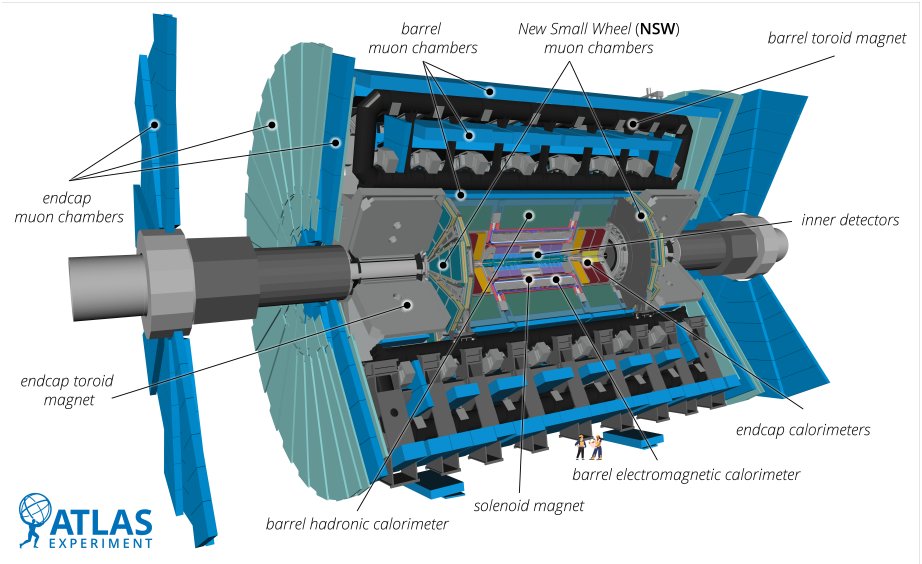


Figure 3.2.: Schematic illustration of the ATLAS detector [33]

The ATLAS experiment at the LHC is designed to measure a wide range of particles from proton-proton as well as heavy ion collisions. It consists of several sub-detectors: the inner detector (ID), the calorimeters (CAL), and the muon spectrometer (MS). They are

arranged cylindrically around the proton beam. This arrangement ensures comprehensive coverage across the full azimuthal range and a pseudorapidity of $|\eta| < 4.9$. The ATLAS experiment at the LHC uses a right-handed coordinate system, with the x-axis pointing towards the center of the LHC, the y-axis pointing upwards, and the z-axis aligned along the beam direction. In this system, the polar angle θ is commonly expressed as pseudorapidity $\eta = -\ln\tan(\theta/2)$, which provides a Lorentz-invariant measure of the angle relative to the beam axis. Differences in pseudorapidity, $\Delta\eta$, are invariant under boosts along the z-axis in the relativistic limit, making them useful for particle identification and separation. The spatial separation between two particles in the detector is quantified using $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$, where $\Delta\phi$ is the difference in azimuthal angle.

The Inner Detector (ID), situated closest to the beam pipe, is designed for precise tracking of charged particles. It consists of the pixel detector, the Semiconductor Tracker (SCT), and the Transition Radiation Tracker (TRT). It is situated within a magnetic field (2 T) aligned with the beam axis. The magnetic field causes charged particles' tracks to bend according to their charge and momentum. The ID tracks these particles, with spatial resolution decreasing with distance from the interaction point. The detector is designed to achieve a transverse momentum resolution of $\sigma_{p_T}/p_T = 0.05\%\, p_T\,[\text{GeV}] \oplus 1\%$, and a transverse impact parameter resolution of 10 μm for high-energy particles in the central η region. The pixel detector provides 3D information, while the SCT offers 2D information, both covering a pseudorapidity range of $|\Delta\eta| < 2.5$. The TRT, used for electron identification, covers $|\Delta\eta| < 2$. To mitigate radiation damage, the ID is cooled to between -5 and $-10\,^{\circ}$C. The ID offers high precision in both $R-\phi$ and z coordinates, enhances electron identification, and aids in heavy-flavour and τ-lepton tagging [28].

After traversing the Inner Detector (ID), particles enter the calorimeter system of the ATLAS experiment, which is designed to measure their energy. The calorimeter system is divided into two main components: the Electromagnetic Calorimeter (ECAL) and the Hadronic Calorimeter (HCAL).

The ECAL, which is crucial for measuring the energy of electrons, positrons, and photons, surrounds the HCAL. The HCAL is primarily responsible for measuring the energy of hadrons. This arrangement ensures accurate energy measurements across a broad range of particles. The energy resolution for the ECAL is given by:

$$\frac{\sigma_E}{E} = \frac{10\%}{\sqrt{E}} \oplus 0.7\%$$

where E is the energy in GeV for electromagnetic showers. For the HCAL, the energy resolution for hadronic showers varies by region. Specifically:

- Barrel and End-Cap Regions:

$$\frac{\sigma_E}{E} = \frac{50\%}{\sqrt{E}} \oplus 3\%$$

- Forward Region:

$$\frac{\sigma_E}{E} = \frac{100\%}{\sqrt{E}} \oplus 10\%$$

In these formulas, σ_E represents the energy resolution, E is the energy in GeV, and the $\oplus$ symbol denotes the addition in quadrature of constant terms.
The ATLAS calorimeter system includes two barrel calorimeters and two end-cap calorimeters, designed to cover a broad pseudorapidity range. It utilizes alternating layers of passive and active materials to measure energy effectively, accommodating both electromagnetic and hadronic showers.
Precision measurements of electrons and photons are achieved through the ECAL, which plays a crucial role in detecting these particles. On the other hand, the HCAL, which consists of the Tile Calorimeter (TileCal) and the Liquid Argon Hadronic End-cap Calorimeter (LAr HEC), is essential for accurate jet reconstruction.
The ATLAS TileCal is a sampling calorimeter that measures the energy of jets and tau leptons. It contributes to the Level 1 trigger system and consists of steel absorbers and scintillating tiles, covering $|\eta| < 1.7$ with approximately 10,000 readout channels. A comprehensive calibration system ensures high data quality [34].
The LAr HEC, which covers up to $|\eta| = 4.9$, consists of several sub-detectors including the Electromagnetic Barrel (EMB), Electromagnetic EndCaps (EMEC), Hadronic EndCaps (HEC), and Forward Calorimeters (FCal). These sub-detectors are maintained at 88 K in cryostats to keep the Liquid Argon in its liquid state [35].
Calorimeters must effectively contain electromagnetic and hadronic showers. Therefore, their depth is crucial. The total thickness of the ATLAS ECAL is more than 22 radiation lengths (X_0) in the barrel and more than 24 X_0 in the end-caps. The HCAL has a thickness of approximately 10 interaction lengths (λ) [28].
Muons (μ) are minimal ionizing particles that are tracked by the Inner Detector (ID) but typically pass through the calorimeters without being fully absorbed due to their high momenta. To achieve accurate muon measurements, especially for those that escape the calorimeters, the ATLAS experiment incorporates an additional tracking detector: the Muon Spectrometer (MS). As the outermost detector in the ATLAS setup, the MS is specifically designed to detect and measure these high-energy muons. It employs gas-filled drift chambers within magnetic fields of 0.5 T (barrel) and 1 T (end-cap) to ensure

precise muon tracking and momentum measurement. The MS uses three large air-core superconducting toroidal magnets to generate a magnetic field of approximately 0.5 T, with separate high-precision tracking and trigger chambers. Monitored Drift Tubes (MDTs) and Cathode Strip Chambers (CSCs) provide precision tracking up to $|\eta| < 2.7$, while Resistive Plate Chambers (RPCs) and Thin Gap Chambers (TGCs) are used for triggering in the barrel and endcap regions, respectively, with momentum resolution better than 4% for most transverse momenta [28, 36]. The ATLAS MS is organized into various chamber layers and sectors, featuring different types of chambers and wheels.

All these sub-detectors of the ATLAS detector produce an overwhelming amount of data, with a recording of up to 3 kHz and a bunch crossing rate of approximately 40 MHz. Most interactions are at low energy scales and not of significant interest, necessitating a real-time filtering system to store only pertinent events. To accomplish this, ATLAS employs a two-stage trigger system [37]. The ATLAS trigger system, consisting of a hardware-based Level-1 (L1) trigger and a software-based High-Level Trigger (HLT), reduces the event rate from 40 MHz to approximately 100 kHz at L1 and further down to 1,000 events per second at the HLT, which uses advanced algorithms on Regions of Interest identified by L1 triggers [38]. Significant upgrades during LS2 enhanced the system's capabilities for Run 3, with the introduction of new features like FPGA-based extractors for improved granularity in calorimeter-triggered events and upgrades to the muon trigger system.

Neutrinos do not interact with the detector material and thus remain undetected. Instead, their presence is inferred from the missing transverse energy, which is the energy required to ensure energy conservation in the plane perpendicular to the beam axis.

4. Fundamentals of Neural Networks

In this chapter, the fundamentals of neural networks are discussed. Over the past few decades, neural networks have emerged as powerful tools across a wide range of applications, from image processing and natural language processing to complex scientific analyses. Their ability to recognize and learn patterns in large and complex datasets makes them particularly valuable for physical research. In this thesis neural networks are used to identify electrons that overlap with jets in the boosted $X \to HH \to b\bar{b}WW^*$ 1-lepton channel.

4.1. Perceptron

The idea of the perceptron was introduced by Frank Rosenblatt in 1958 [39] to mimic the neural structure of the human brain [40]. It is known as the first single-layer neural network and it serves as their foundation. A perceptron can solve linearly separable problems by calculating a weighted sum of inputs plus a bias, which is fed into an activation function to produce a binary output. Each input node has therefore an individual weight. Figure 4.1 shows a diagram of a perceptron.

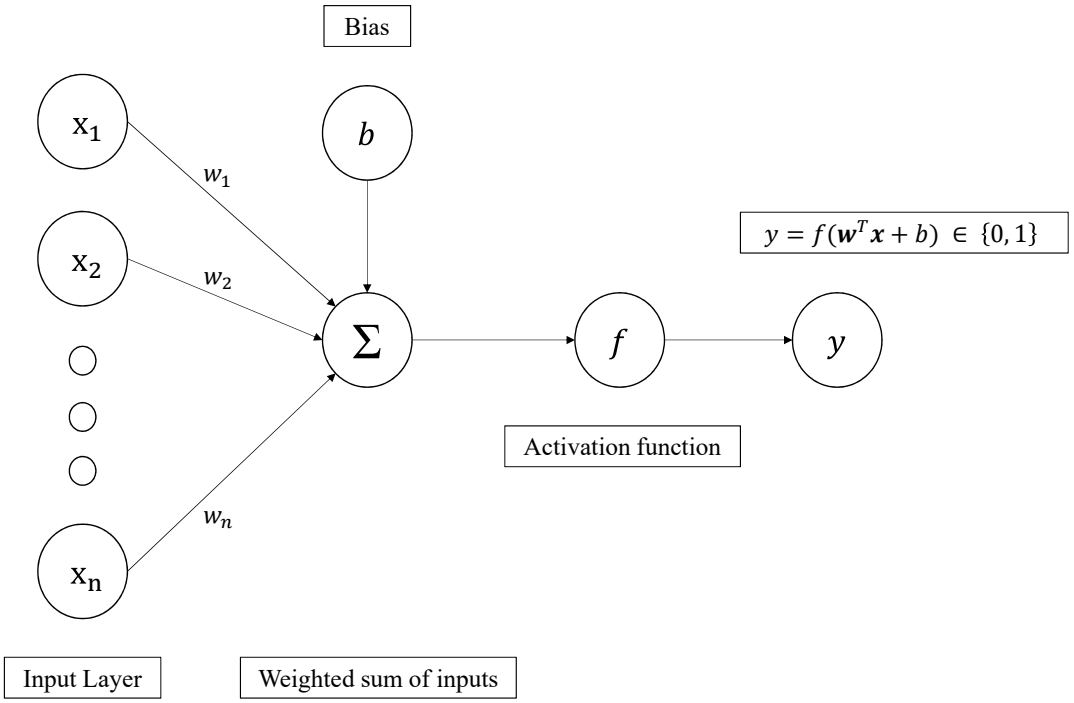


Figure 4.1.: Diagram of a Perceptron showing a weighted sum of inputs creating a binary output y. It consists of the inputs $\mathbf{x}$, their weights $\mathbf{w}$, a bias b, and an activation function f: $y = f(\mathbf{w}^T\mathbf{x} + b) \in \{0, 1\}$

4.1.1. Multi-Layer Perceptron

The extended version of the perceptron is the Multi-Layer perceptron (MLP) also known as Neural Network. The MLP consists of multiple layers: an input layer, one or more hidden layer(s) and an output layer. The number of hidden nodes (z) and layers varies based on the problem's complexity. Typically, it is better to have too many hidden units in a model than too few. If the model has too few hidden units, it may not have the flexibility to capture the nonlinearities in the data. On the other hand, if there are too many hidden units, the extra weights can be minimized toward zero using proper regularization. The number of hidden units generally ranges from 5 to 100, increasing with the number of inputs and training cases.
It is a common strategy to start with a large number of units and apply regularization during training. The number of hidden layers is determined by prior knowledge and experimentation. Each layer plays a role in extracting features from the input for regression or classification tasks. The use of multiple hidden layers enables the creation of hierarchical features at various levels of resolution [40]. A MLP is able to solve non-linear

problems and each unit of a hidden layer is a perceptron as in Figure 4.1. A MLP is a fully connected network. This means that each node is connected to every node in the subsequent layer. Figure 4.2 shows a diagram of a MLP with one hidden layer.

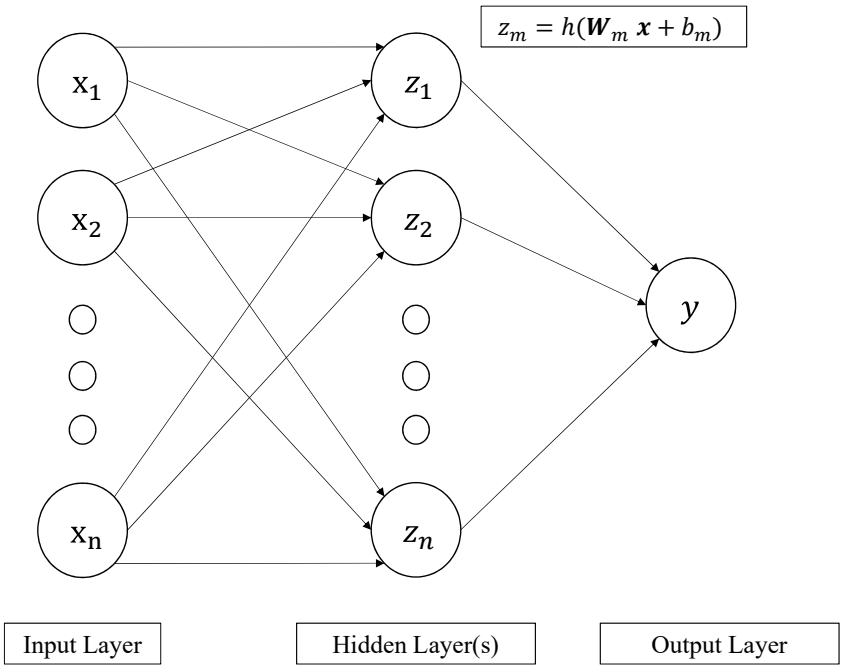


Figure 4.2.: Diagram of a Multi-Layer Perceptron showing an input layer $\mathbf{x}$, hidden layer(s) $\mathbf{z}$ and the output layer y. Each unit z_m of the hidden layer is a perceptron: $z_m = h(\mathbf{W}_m\mathbf{x} + b_m)$

4.2. Types of Neural Networks

In the field of neural networks, there is a wide variety of types. The simplest and most commonly used network type is the Feedforward neural network (FNN). FNNs are a type of artificial neural networks (NN) where connections between units do not form cycles. They're referred to as "feedforward" because information moves in only one direction - forward. It starts at the input nodes, passes through any hidden nodes, and finally reaches the output nodes. There are two main types of Feedforward NNs: single layer FNN and multi-layer FNN [41]. The key difference is that an MLP is a specific type of FNN where each layer is fully connected. In some definitions, the number of nodes in each layer is equal.

Deep Neural Networks (DNNs) are NNs with at least two hidden layers. The field also

includes more advanced forms such as Recurrent Neural Networks (RNNs) [41] and Convolutional Neural Networks (CNNs) [42] but these are not covered in this thesis.

4.3. Training and Optimization

Training neural networks requires considerable skill. These models are often overparameterized, and the optimization problem is non-convex (meaning it has multiple valleys and peaks) and unstable unless certain best practices are followed [40]. The goal of training is to enhance model accuracy and reduce loss.

4.3.1. Activation and Loss Functions

Key aspects include the activation and loss functions of the neural network. Activation functions play a crucial role in artificial neural networks because they convert a neuron's input signal into an output signal that is passed on to the next layer. These functions determine a neuron's output based on the weighted sum of its inputs and their corresponding weights and ensure this output is fed forward to subsequent layers in the network. The choice of activation function is crucial, as it influences the network's ability to learn and represent complex data.
Linear activation functions, while simple, are limited in their ability to capture complex patterns resulting in the neural network behaving like a linear regression model, which often lacks the capability to handle complex tasks [43].
Non-linear activation functions are essential in deep learning because they enable the network to model intricate relationships and adapt to diverse data types, including images, audio and text. This non-linearity is crucial for processing high-dimensional, non-linear datasets with multiple hidden layers and complex architectures. Unlike simple linear models, non-linear activation functions allow the network to approximate complex mappings between inputs and outputs. Additionally, these functions must be differentiable to support backpropagation (see chapter 4.3.2 for a detailed explanation), which optimizes the network's weights by minimizing errors through gradient descent or other optimization techniques [43]. Figure 4.3 illustrates some of the most common activation functions, which will be discussed in more detail in the following. The ranges of the axes have been chosen so that all features are clearly visible.

There is the non-linear sigmoid activation function:

$$\sigma = \frac{1}{1 + e^{-x}}. \tag{4.1}$$

It is the most widely used activation function and transforms values to the range of 0 to 1, with a turning point at x = 0 [43].
Another popular activation function is ReLU, which stands for Rectified Linear Unit. It is a non-linear activation function used in neural networks. One of its key advantages is that it prevents all neurons from being activated simultaneously. Instead, a neuron activates only when the output of the linear transformation is positive. Mathematically, ReLU is defined as:

$$f(x) = max(0, x). \tag{4.2}$$

ReLU is more efficient compared to other activation functions because it activates only a subset of neurons at any given time, reducing computational complexity. However, it can encounter issues where the gradient is zero, causing weights and biases to not be updated during the backpropagation phase of neural network training [43].
The softmax activation function is an extension of the sigmoid function, for multi-class classification problems. It converts the outputs of a network into probabilities that sum to 1 across all possible classes, making it ideal for scenarios where each data point must be classified into one of several categories. Mathematically, the softmax function for a given data point can be expressed as:

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}, \tag{4.3}$$

where x_i is the output for class i, and the denominator is the sum of the exponential functions of all class outputs. This normalization ensures that the output values represents a probability distribution over the classes. When building a model for multi-class classification, the output layer of the network will have a number of neurons equal to the number of classes, and the softmax function is applied to these outputs to obtain the final class probabilities [43].
Other functions used in machine learning are the loss functions. They are essential because they measure how much the predicted outputs of a model deviate from the actual target values. During training, they offer a metric to evaluate model performance and guide

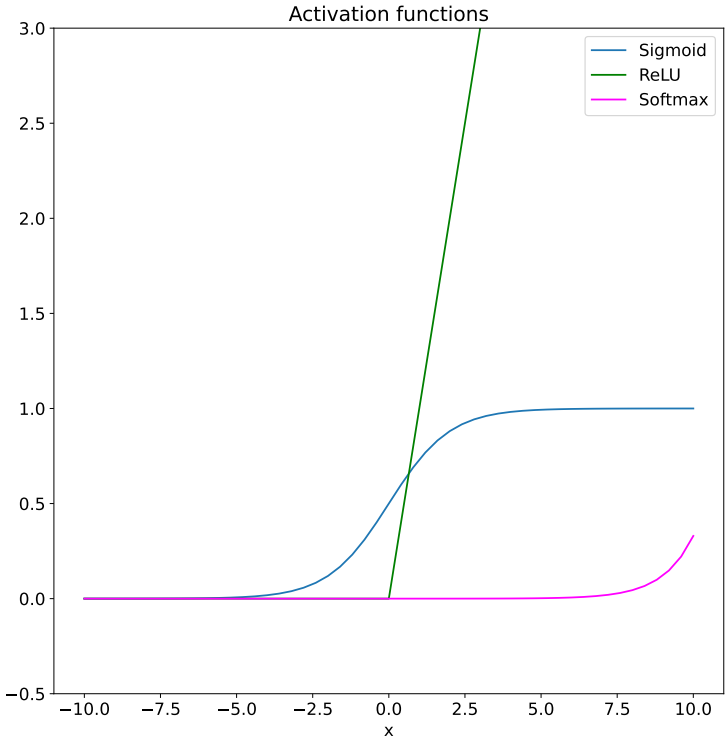


Figure 4.3.: Different activation functions: Sigmoid, ReLU and Softmax

the learning algorithm to adjust the model's parameters, reducing errors and enhancing predictions. By balancing bias and variance, effective loss functions ensure the model can generalize effectively to new, unseen data [40].

There are many different loss functions for machine learning, such as mean squared error or mean absolute error. Cross entropy (CE) is a widely used loss function for classification tasks in neural networks. Originating from adaptive algorithms designed for estimating probabilities in complex stochastic networks, CE has proven effective for combinatorial optimization problems. In deep learning, CE serves as a fundamental loss function for both binary and multi-class classification problems [44]. The CE loss for a classification task is mathematically defined as:

$$\mathrm{CE(j,\ p)} = -\sum_{j} g_j \log(p_j) \tag{4.4}$$

where g_j represents the discrete ground-truth label for class j, and p_j is the predicted probability of class j. In multi-class classification, the predicted probabilities p_j are typically obtained from a softmax layer. This formulation results in the softmax cross entropy

loss, also known as categorical cross entropy loss. For binary classification, probabilities are often generated using a sigmoid layer leading to the sigmoid cross entropy loss, also known as binary cross entropy loss. In practice, CE is preferred for classification tasks because it directly measures the discrepancy between the predicted probabilities and the true labels, leading to more stable and effective training results [44].

4.3.2. Gradient Descent and Backpropagation

Gradient Descent is an optimization algorithm that aims to minimize an objective function $J(\theta)$ parameterized by $\theta \in \mathbb{R}^d$. It updates the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$. The step size towards the minimum is determined by the learning rate η. The process involves moving in the direction of the gradient of the objective function's surface until reaching a local minimum [45].
There are three variants of gradient descent. Depending on the amount of data, a trade-off is made between the accuracy of the parameter update and the time it takes to perform an update. Batch gradient descent, also known as vanilla gradient descent, computes the gradient of the loss function with respect to the parameters θ for the entire training dataset:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{4.5}$$

Batch gradient descent can be slow and impractical for large datasets, as it requires computing the gradient vector for the entire dataset before each update, leading to redundant calculations. This approach also prevents online updates, delaying model adjustments until the full dataset is processed. Parameters are updated in the direction of the gradients, with the learning rate determining the size of the update. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces [45].
Stochastic gradient descent (SGD) performs a parameter update for each training example $x^{(i)}$ and outcome $y^{(i)}$:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{4.6}$$

SGD avoids this redundancy by performing one update per example, which is usually much faster and can support online learning. However, SGD's frequent updates with high variance can cause the objective function to fluctuate heavily, which can complicate convergence to the exact minimum. It has been shown that decreasing the learning rate

over time allows SGD to converge to the same point as batch gradient descent [45].
Mini-batch gradient descent combines the advantages of both batch and stochastic gradient descent by performing an update for every mini-batch of n training examples:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}). \tag{4.7}$$

This approach reduces the variance in parameter updates, leading to more stable convergence, and can make use of highly optimized matrix operations common in state-of-the-art deep learning libraries. Common mini-batch sizes range between 50 and 256 training examples, but can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a neural network, and the term SGD is often used to refer to mini-batch gradient descent as well [45].
The Backpropagation algorithm is essential for training multilayer neural networks. It uses Gradient Descent to minimize the error function by adjusting the network weights. The Backpropagation algorithm consists of two phases: feed-forward and backpropagation. In the feed-forward phase, the input x is fed into the network, and the primitive functions at the nodes and their derivatives are evaluated and stored. In the backpropagation phase, the error signal is propagated back through the network, starting from the output unit. During this process, the gradients of the network function with respect to the weights are calculated using the chain rule. The collected result at the input unit gives the derivative of the network function with respect to the input x.
The error function $E(w)$ measures the number of false classifications made by a perceptron and is defined as:

$$E(w) = \sum_{x \in A} (1 - f_w(x)) + \sum_{x \in B} f_w(x), \tag{4.8}$$

Here, A and B are sets of input vectors in n-dimensional space that need to be separated. The binary function f_w is defined such that $f_w(x) = 1$ for $x \in A$ and $f_w(x) = 0$ for $x \in B$. The function f_w depends on the weight vector w and a threshold. The goal of perceptron learning is to minimize the error function $E(w)$, which is always non-negative, aiming to reach the global minimum where $E(w) = 0$. This is achieved by starting with a random weight vector w and iteratively searching for a better alternative to reduce $E(w)$ at each step.
During the feed-forward step, the output of each unit is stored. In the backpropagation step, the gradient of E with respect to each weight w_{ij} is computed. The weight update is given by $\Delta w_{ij} = -\gamma o_i \delta_j$, where o_i is the output of unit i and δ_j is the backpropagated error at node j. This process transforms the backpropagation algorithm into a learning

method for neural networks [46].

Gradient Problems

In the training process, monitoring the gradients of different parameters is essential, as issues can become evident through the Gradient Descent. There are two main types of gradient issues: vanishing gradients and exploding gradients.
The vanishing gradient problem occurs during backpropagation in deep neural networks, where gradients become progressively smaller as they propagate backward through layers, especially with sigmoid activation functions, whose derivatives can approach zero in saturated regions. As a result, weight updates in earlier layers become negligible, leading to very slow training or halted progress. This issue is less severe in shallow networks but critical in deeper ones, where small gradients decay and hinder model training. The problem can be identified by observing stagnation in model weights, minimal performance improvements, or erratic loss function behavior during training. To address this, techniques like batch normalization, ReLU activation functions, skip connections, Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRUs) architectures [47], and gradient clipping can stabilize gradients and improve training stability. Gradient clipping limits gradient size, while batch normalization scales activations to enhance training efficiency [48].
The exploding gradient problem arises in deep neural networks when gradients of the loss function with respect to the weights become excessively large during backpropagation. Exploding gradients are typically caused by high weight values, leading to large derivatives and substantial deviations in weight updates. This can cause the network to oscillate around local minima, making it difficult to converge to a global minimum. Key indicators of exploding gradients include erratic behavior of the loss function, encountering non-numeric values in calculations, and rapid increases in weight values. Tools like TensorBoard [49] can help visualize gradients and detect the problem. To address exploding gradients, techniques such as gradient clipping and batch normalization are commonly used.

4.3.3. Optimization techniques

Optimization is crucial for developing accurate and efficient neural network models. It involves fine-tuning model parameters to enhance performance and achieve optimal results. The goal is to find the best configuration that enables the model to learn from data and make accurate predictions. As neural networks are used in increasingly complex

applications, effective optimization strategies are essential to realizing their full potential and delivering high-quality results.

Hyperparameters

Hyperparameters are parameters that are set before training and cannot be adjusted during training, such as the number of hidden layers, learning rate, and batch size. These parameters significantly impact the efficiency and accuracy of model training and must be carefully configured. Since manual tuning of hyperparameters is labor-intensive and relies on experience, automated hyperparameter optimization (HPO) is proposed as a solution. HPO aims to minimize human effort while improving training accuracy and efficiency [50].

Optimizer

Optimizers play a crucial role in enhancing the accuracy and speed of training neural networks. They adjust the model's weights based on gradients to minimize the loss function. Key hyperparameters associated with optimizers include the choice of optimizer, mini-batch size, momentum, and learning rate. SGD with momentum accelerates convergence by incorporating momentum into the weight updates. This method calculates an exponentially weighted average of past gradients, smoothing the update process. The momentum term, usually set to values such as 0.9, helps to reduce oscillation and directs the updates in a consistent direction:

$$v_{dw} = \beta v_{dw} + (1 - \beta) dw \tag{4.9}$$

$$w = w - lr \cdot v_{dw} \tag{4.10}$$

where β is the momentum parameter.
Root Mean Square Propagation (RMSprop) adjusts the learning rate based on the magnitude of recent gradients, improving convergence and reducing oscillation. It uses exponential moving averages of squared gradients to adaptively scale the learning rate for each parameter. The update equations are:

$$S_{dw} = \beta S_{dw} + (1 - \beta) dw^2 \tag{4.11}$$

$$w = w - lr \cdot \frac{dw}{\sqrt{S_{dw}}} \tag{4.12}$$

where β is typically set to 0.9.
Adaptive Momentum Estimation (Adam) combines the advantages of both momentum

and RMSprop. It computes adaptive learning rates for each parameter using both the first moment (mean) and second moment (uncentered variance) of the gradients. The update equations for Adam are:

$$v_{dw} = \beta_1 v_{dw} + (1 - \beta_1) dw \tag{4.13}$$

$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2) dw^2 \tag{4.14}$$

$$\hat{v}_{dw} = v_{dw} + (1 - \beta_1^t) \tag{4.15}$$

$$\hat{S}_{dw} = S_{dw} + (1 - \beta_2^t) \tag{4.16}$$

$$\mathbf{w} = \mathbf{w} - \mathbf{lr} \cdot \frac{\hat{\mathbf{v}}_{\mathbf{dw}}}{\sqrt{\hat{\mathbf{S}}_{\mathbf{dw}} + \epsilon}} \tag{4.17}$$

where β_1 is typically 0.9, β_2 is typically 0.999, and ϵ is a small constant (usually 10^{-8}) to prevent division by zero. The equations for the bias are calculated analogously, with the weight w being replaced by the bias b. Adam is widely used due to its effective performance and minimal need for tuning, making it a preferred default optimizer in many deep learning frameworks. While RMSprop and Adam are often favored for their adaptability and efficiency, SGD with momentum may require more time to converge to an optimal solution [50].

Learning Rate Techniques

The learning rate (LR) is a crucial hyper-parameter in training neural networks, determining the step size for updating model weights during optimization. It can be set as a constant value or adjusted through various scheduling methods. Learning rate scheduling techniques, such as linear decay and exponential decay, gradually adjust the LR over time or iterations to improve training efficiency [51]. Another popular method is ReduceLROnPlateau, which lowers the LR when a plateau in model performance is detected, helping the model converge more effectively [52]. Choosing the optimal LR or its schedule is challenging and often requires experimentation or automated tuning [50].

Regularization Techniques

Regularization is essential for managing the neural network's complexity and preventing overfitting, especially when training data is limited. Key regularization techniques include L1 and L2 regularization, dropout and data augmentation. During the training of the neural network, it is decided by trial and error which techniques are really necessary.

L1 regularization, or Lasso Regression, penalizes the sum of absolute weights, promoting sparsity in the model. Conversely, L2 regularization, or Ridge Regression, penalizes the sum of squared weights, typically providing better predictive performance and computational efficiency. While L1 produces simpler models with inherent feature selection, L2 tends to be more robust and capable of handling complex patterns.
Dropout randomly deactivates neurons during training, reducing overfitting by making the model less dependent on specific weights. A dropout rate between 20% and 50% is commonly used, with the rate influencing training dynamics and model performance [50, 53].
Data augmentation artificially creates new data using flips, rotations, crops and more [54].

Batch Normalization

Batch Normalization (BatchNorm) is a widely used technique that accelerates and stabilizes the training of DNNs by applying a transformation to its inputs, aiming to maintain the mean output close to zero and the standard deviation close to one. During training, the layer normalizes its output using the mean and standard deviation of the current batch of inputs:

$$\text{output} = \gamma \cdot \left(\frac{\text{batch} - \text{mean(batch)}}{\sqrt{\text{var(batch)} + \epsilon}} \right) + \beta, \tag{4.18}$$

where ϵ is a small constant to avoid division by zero, γ is a learned scaling factor (initialized as 1), and β is a learned offset factor (initialized as 0) [55]. This process stabilizes activations and addresses Internal Covariate Shift (ICS), which refers to the change in the distribution of network activations caused by adjustments in network parameters during training [50, 56].

Early Stopping

Early stopping is a technique used in HPO and neural network training to efficiently manage computational resources by halting models or trials unlikely to yield better results, thus reallocating resources to more promising configurations. In neural network training, it prevents overfitting by stopping when performance ceases to improve [50]. The mechanism involves monitoring a specific metric, such as loss, during training; if it doesn't improve beyond a specified threshold (min_delta) over a set number of epochs (patience), training is halted early, as seen in frameworks like TensorFlow/Keras [57].

5. Identification of electrons in jets for the boosted $X \to HH \to bbWW^*$ analysis

In this chapter a machine learning approach to distinguish signal electrons overlapping with jets from other jets and leptons is presented. This topology can arise for example in the boosted $X \to HH \to b\bar{b}WW^*$ search in the 1-lepton channel which is the signal considered in this study.

5.1. The $HH \to b\bar{b}WW^*$ channel

In particle physics, the study of new resonances is a crucial aspect of understanding the fundamental forces and particles of the universe. Consider a hypothetical resonance, denoted as X, which decays into two Higgs bosons in a back-to-back configuration. The subsequent decays of these Higgs bosons are of significant interest, particularly the $HH \to b\bar{b}WW^*$ decay channel, which has the second highest branching ratio. This channel is thus highly relevant for experimental studies.
For very high resonant masses ($m_X \gtrsim 2$TeV), the resulting Higgs bosons will inherit a substantial amount of kinetic energy. This kinetic energy leads to a boost. In these scenarios, the decay products of the Higgs boson ($H \to b\bar{b}$) and the hadronically decaying W boson (W_{had}) are collimated and are often reconstructed as a single jet due to detector resolution limits. Additionally, the lepton from the leptonically decaying W boson may overlap with the W_{had} jet as shown in Figure 5.1, complicating the reconstruction. Although this unique topology is not common in background processes and therefore interesting, it is challenging to accurately reconstruct the charged lepton in this dense environment, resulting in a low high signal efficiency during reconstruction [11].
Studying these boosted Higgs bosons and their decay products is not only a promising way to probe the characteristics of the resonance X, but it also enhances our understanding of Higgs physics and potential new physics beyond the Standard Model.

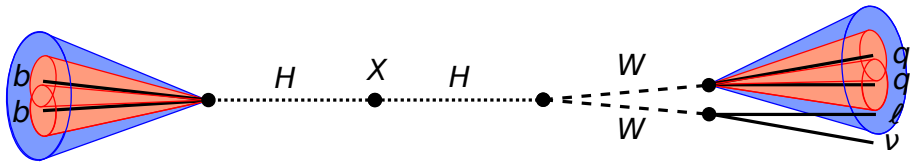


Figure 5.1.: Schematic illustration of the boosted $HH \to b\bar{b}WW^*$ production in the 1-lepton channel [11]

5.2. Object Reconstruction

5.2.1. Lepton Reconstruction

In this thesis, leptons are considered to be either an electron or a muon because τ leptons decay before reaching the detector. This thesis will further focus on the lepton being an electron e. Electrons are reconstructed by matching the track they leave in the Inner Detector with energy deposits in the calorimeters. The tracks are reconstructed with a Gaussian sum filter to account for the energy loss of charged particles in material. Energy deposits are collected in topo-clusters using the EM scale, which accurately represents the energy deposited by electromagnetic showers [58]. In this thesis, electrons only need to fulfill $p_T > 10$ GeV.

5.2.2. Jet Reconstruction

When quarks and gluons are produced in a collision, they hadronize, creating collimated showers of hadrons. These hadrons deposit energy in the calorimeters and, if charged, leave tracks in the Inner Detector. The hadronic objects reconstructed from these constituents are called jets.
For jet reconstruction, various algorithms have been developed. The most commonly used algorithms employ the following formula [59]:

$$d_{ij} = \min\left(p_{ti}^k, p_{tj}^k\right) \frac{\Delta R_{ij}^2}{R^2}, \quad \text{with } k = \begin{cases} -2 & \text{anti-}k_t \\ 0 & \text{Cambridge-Aachen} \\ 2 & k_t \end{cases} \tag{5.1}$$

This formula defines the type of algorithm and the size parameter R. Here, $p_{ti/j}$ represents the transverse momentum of the constituents i and j. The distance between these constituents in detector coordinates, based on rapidity instead of pseudorapidity, is given by

$$\Delta R_{ij} = \sqrt{(y_i - y_j)^2 + (\phi_i - \phi_j)^2}. \tag{5.2}$$

The value of d_{ij} is calculated along with a cut-off value $d_{iB} = p_{ti}^k$ for each pair of constituents. The exponent p influences the prioritization of energy scales relative to the geometric scales. If $d_{ij} < d_{iB}$, the objects i and j are merged into a single entity. Conversely, if $d_{ij} > d_{iB}$, the object i is identified as a jet and removed from the jet reconstruction process. This procedure continues until all objects are assigned to jets [59].
In the ideal case, each jet corresponds to one parton. However, this is not the case in boosted topologies. To determine the number of partons that initiated a jet, it is essential to analyze the distribution of energy within the jet, referred to as its substructure. This can be done using two main types of substructure variables. The first type involves Energy Correlator Functions (ECFs) [60], which are defined as follows:

$$ECF(N, \beta) = \sum_{i_1 < i_2 < \ldots < i_N \in J} \left(\prod_{a=1}^{N} p_{T,i_a} \right) \left(\prod_{b=1}^{N-1} \prod_{c=b+1}^{N} \Delta R_{i_b i_c} \right)^{\beta} \tag{5.3}$$

The ECFs indicate the energy distribution inside the jet. Generalized ECF Ratios as used in the variables L_2 and L_3, which are defined in table 5.1, are defined as follows:

$$\begin{aligned}
1e_3^{\beta=1} &= \sum_{1 \le i < j < k \le n_J} z_i z_j z_k \min\{\theta_{ij}^\beta, \theta_{ik}^\beta, \theta_{jk}^\beta\} \\
3e_3^{\beta=1} &= \sum_{1 \le i < j < k \le n_J} z_i z_j z_k \theta_{ij}^\beta \theta_{ik}^\beta \theta_{jk}^\beta \\
1e_2^{\beta=1} &= \sum_{1 \le i < j \le n_J} z_i z_j \theta_{ij}^\beta
\end{aligned}$$

The ratios for $\beta = 1$

$$C_2 = \frac{ECF_3 \cdot ECF_1}{(ECF_2)^2} \tag{5.4}$$

$$D_2 = \frac{ECF_3 \cdot ECF_1 \cdot (ECF_1)^2}{(ECF_2)^3} \tag{5.5}$$

are useful for distinguishing between two-prong and one-prong jets. Two-prong jets might indicate the presence of a boosted W boson decaying hadronically, while one-prong jets are often associated with background events [61].

The second method uses a quantity called N-subjettiness τ_N which is defined as

$$\tau_N(\alpha) = \frac{1}{d_0(\alpha)} \sum_{i \in J} p_{T,i} \cdot \min(\Delta R_{1i}^{\alpha}, \Delta R_{2i}^{\alpha}, \ldots, \Delta R_{Ni}^{\alpha}), \tag{5.6}$$

with

$$d_0(\alpha) = \sum_{i \in J} p_{T,i} \cdot \Delta R^{\alpha}. \tag{5.7}$$

N-subjettiness quantifies how likely a jet is composed of at least N subjets. The ratio $\tau_{MN} = \frac{\tau_M}{\tau_N}$ with $M > N$ becomes small if a jet contains M or more subjets [62].

Unified Flow Object Jets

Unified Flow Object (UFO) jets utilize a combination of tracking information, Particle Flow objects (PFOs), topoclusters, and Track-CaloClusters (TCCs) as input. These jets are constructed using the anti-k_t algorithm with a large radius of $R = 1.0$, which allows them to capture all hadronic decay products from boosted decays into a single jet. UFO jets in this thesis are W_{had} jets and must meet the criteria of $p_T > 200$ GeV [63, 64].
UFOs combine the advantages of topoclusters, PFOs and TCCs. While topocluster-based jets perform well at low p_T, they face challenges at high p_T and are sensitive to pile-up effects. PFOs enhance performance across a broad p_T range but fall short at high p_T compared to topoclusters. Conversely, TCCs are effective at high p_T but are prone to pile-up issues at low p_T. By integrating the strengths of topoclusters, PFOs and TCCs, UFOs offer improved tagging performance and greater pile-up stability. This integration results in enhanced performance across the entire p_T spectrum, addressing the shortcomings of previous jet definitions and providing a more balanced solution for both low and high p_T regimes [65].

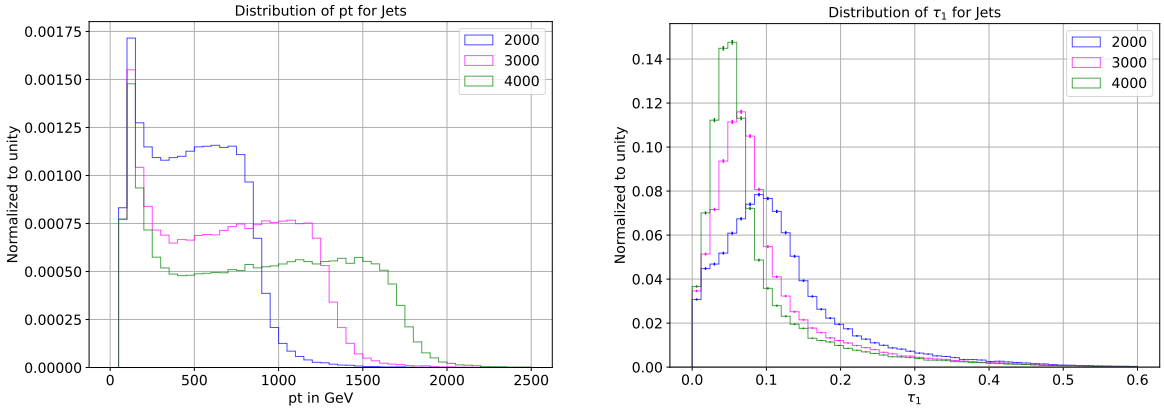
5.3. Monte Carlo Samples and Data Preparation

5.3.1. Monte Carlo Samples

The $X \to HH$ samples used in this study are generated at leading order in α_S using `MadGraph` 3.5.1 [66] with the NNPDF23LO PDF set [67]. `PYTHIA` 8.308 [68] in the A14 tune [69] is used for parton shower simulation with the heavy quark flavour decays being performed by EvtGen 2.1.1 [70]. Samples are produced for $m_X = 2.0, 3.0$ and $4.0\,\text{TeV}$. The full ATLAS detector simulation is used. Filters are applied to enforce $HH \to b\bar{b}WW^*$ decays in 1-lepton final state.

5.3.2. Data Preparation

The initial step in the analysis involved investigating the objects present in the samples with the focus set on UFO jets, electrons and muons. Figure 5.2(a) displays the distribution of the transverse momentum of all jets present in the samples. Figure 5.2(b) shows the variable τ_1.



(a) Distribution of transverse momentum p_T for jets without p_T cut applied

(b) Distribution of τ_1 for jets

Figure 5.2.: Kinematic distributions for jets

The figures clearly shows distinct differences between the various resonance masses. To address this, the mass points are combined into a single dataset to ensure that the neural network's performance remains independent of the specific mass points utilized.
Next, the Truth Labels of the jets and the Truth Types of the electrons are examined. These provide information on the true nature and origin of the reconstructed jets and electrons. The distributions are illustrated in Figure 5.3.
The distribution of the jets' Truth Labels reveals that the majority of jets are categorized as either $H \rightarrow bb$ jets or W_{had} jets ("other_from_Higgs"), which is consistent with the information presented in Figure 5.1. There are also numerous "qcd" jets, which are not associated with top quarks or W/Z bosons and are thus treated as background jets.
The distribution of electron Truth Types shows that only a small fraction are prompt electrons ("IsoElectron"). The majority of electrons originate from background photons ("BkgPhotons") or from the conversion of prompt photons ("BkgElectron"), with some electrons having an unknown origin. Prompt means that the particle is produced directly from the hard scattering process in a high-energy collision, whereas non-prompt particles are produced in secondary processes.

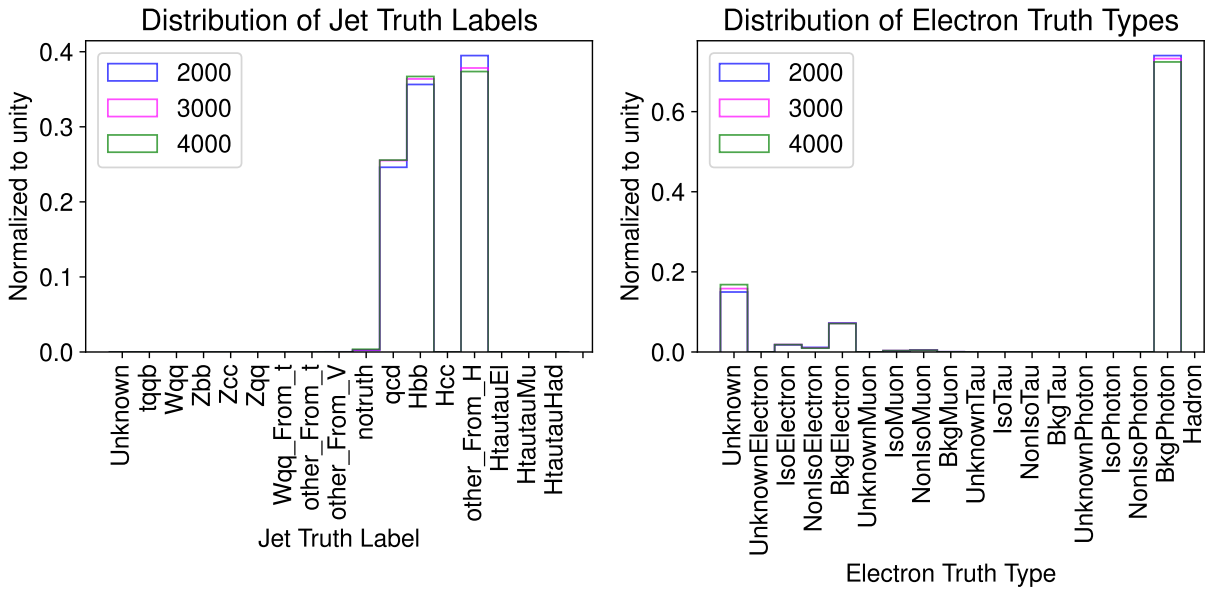
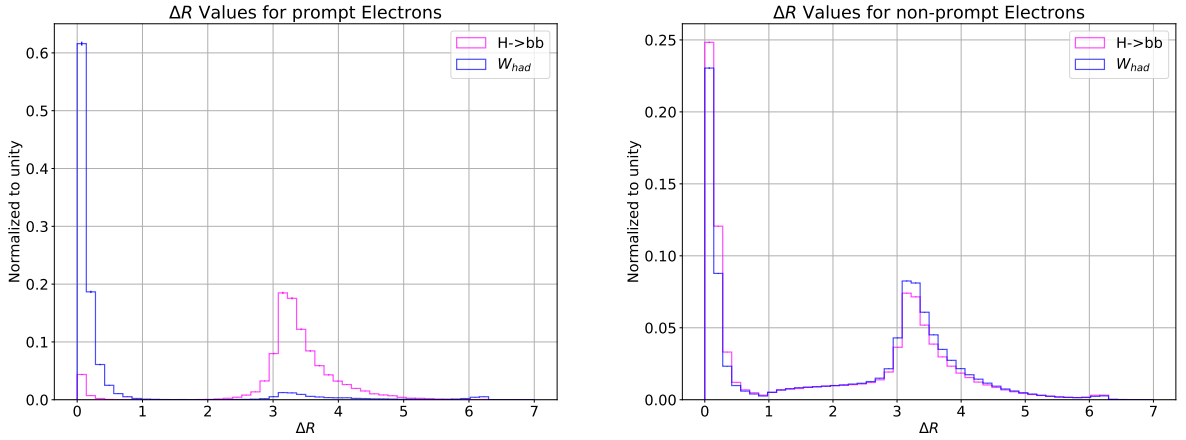


Figure 5.3.: Truth Labels/Types of Jets and Electrons

Figure 5.4 presents the distribution of ΔR between prompt and non-prompt electrons, as well as $H \to bb$ and W_{had} jets. Figure 5.4(a) focuses on the ΔR distribution between prompt electrons and $H \to bb$ and W_{had} jets, with electrons identified as prompt based on truth information. The plot clearly shows that $H \to bb$ jets have a peak in the ΔR values around π, while W_{had} jets peak at values below 0.5. This observation is consistent with Figure 5.1, indicating that prompt electrons are significantly closer to W_{had} jets than to $H \to bb$ jets.
Additionally, Figure 5.4(b) examines the ΔR distributions for non-prompt electrons with $H \to bb$ and W_{had} jets. These distributions reveal peaks at both small ΔR values and around π for non-prompt electrons, irrespective of the jet type. This suggests that there is no substantial difference between the jets for non-prompt electrons, indicating that further separation beyond distinguishing between signal and background classes may be unnecessary.

Classes

The signal class is defined as W_{had} jets paired with prompt electrons where the ΔR value between them is less than 0.5. Conversely, the background class includes all other combinations of jets and leptons with any ΔR value.



(a) Distribution of ΔR for prompt electrons

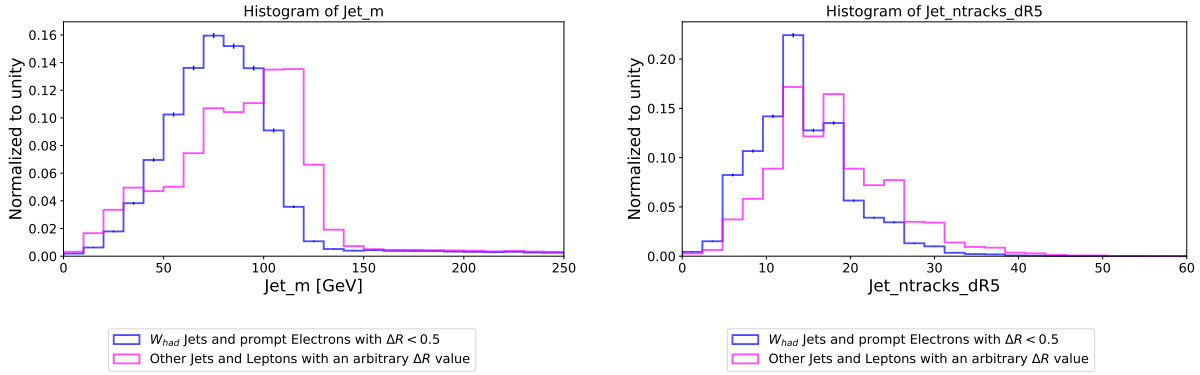
(b) Distribution of ΔR for non-prompt electrons

Figure 5.4.: Comparison of distributions for ΔR of prompt and non-prompt electrons.

5.4. Variables to identify electrons in jets

To select the variables for the neural network used to identify electrons in jets, various properties were analyzed to determine the most distinct features for differentiating between the signal and background classes. The chosen variables based on these analyses are summarized in Table 5.1.

Figures 5.5, 5.6, and 5.7 illustrate the distributions of the selected variables used for training the neural network. Figure 5.5 shows the kinematics of jets. For instance, Figure 5.5(a) displays the jet mass, which clearly separates the signal, peaking around 80 GeV, from the background, which peaks around 110 GeV. Figure 5.5(b) shows the number of tracks within $\Delta R = 0.5$, where the background class is slightly shifted to the right.

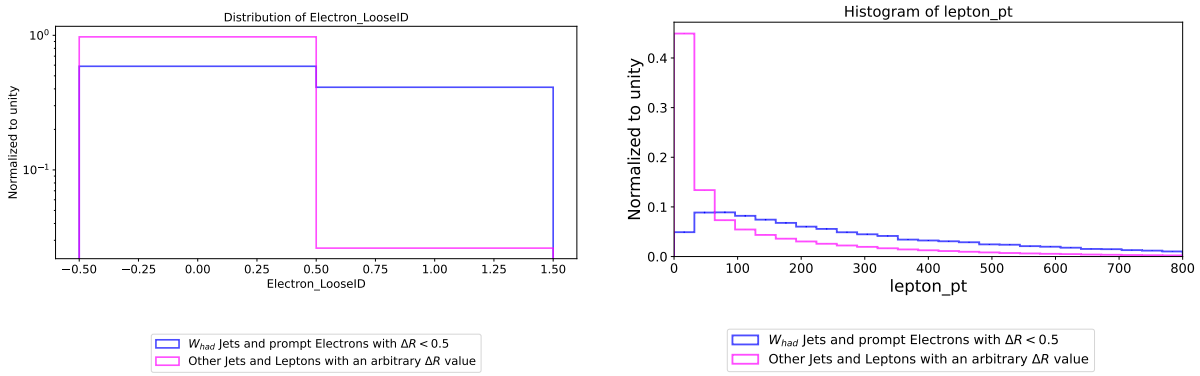


(a) Distribution of the jets mass

(b) Distribution of the number of tracks within $\Delta R = 0.5$

Figure 5.5.: Comparison of distributions for jet kinematics

Figure 5.6 presents two electron-related variables. The Loose ID distribution (Figure 5.6(a)) is relatively even for the signal class, whereas the background class shows a higher concentration in the bin at 0. Additionally, the lepton p_T distribution (Figure 5.6(b)) reveals a peak near 0 for the background class, which is absent in the signal class.



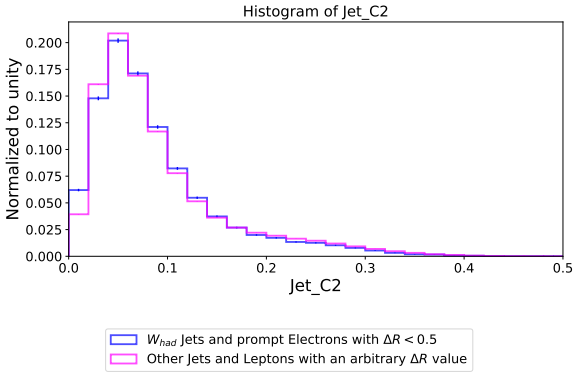
(a) Distribution of the electron Loose ID

(b) Distribution of the lepton p_T in GeV

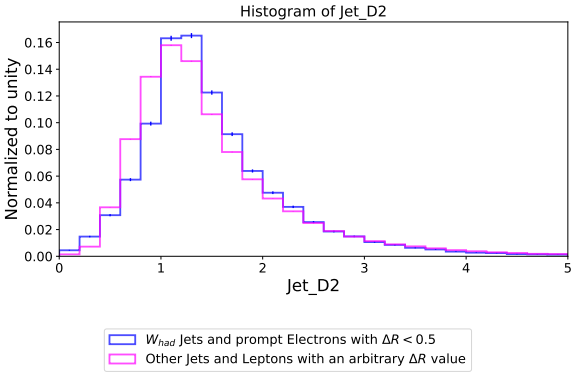
Figure 5.6.: Comparison of the electron Loose ID and lepton p_T distributions.

Figure 5.7 depicts various jet substructures. Figures 5.7(a) and 5.7(b) display the prominent ECF ratios, C_2 and D_2. These plots show subtle differences, with both classes peaking around similar values. On the left side of the peak, the background class is slightly higher, while on the right side, the signal class is slightly more pronounced.
Figures 5.7(c) and 5.7(d) illustrate the splitting scales Split12 and Split23, where the jet constituents are reclustered using the k_t algorithm until exactly two or three subjets, respectively, are formed. A clear separation between the signal and background classes is evident. For Split12, the background peaks lower than the signal, while for Split23, the background peaks higher than the signal.
The generalized ECF ratios L_2 and L_3 are shown in Figures 5.7(e) and 5.7(f). The L_2 distribution shows a more pronounced difference between signal and background, with the background resembling a plateau while the signal peaks around $L_2 \simeq 0.22$. For L_3, the distributions appear similar, although the background class fluctuates slightly, being sometimes lower and sometimes higher than the signal class.
Figures 5.7(g), 5.7(h), and 5.7(i) illustrate the distributions for τ_{21}, τ_{32}, and τ_{42}. These distributions exhibit distinct characteristics: for τ_{21}, the signal peaks around $\tau_{21} \simeq 0.45$, while the background peaks slightly lower at around $\tau_{21} \simeq 0.3$. The τ_{32} distribution forms a plateau around $\tau_{32} \simeq 0.5$ for the signal, whereas the background peaks at approximately $\tau_{32} \simeq 0.75$. For τ_{42}, the signal peaks around $\tau_{42} \simeq 0.3$, while the background is centered around $\tau_{42} \simeq 0.55$. These variations suggest that the signal class likely contains more

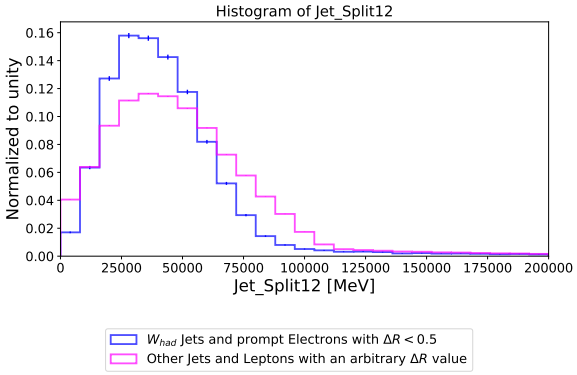
subjets compared to the background class.
Additional plots of other variables, which were found to be less effective due to the lack of distinction between the signal and background classes, are provided in Annex A.



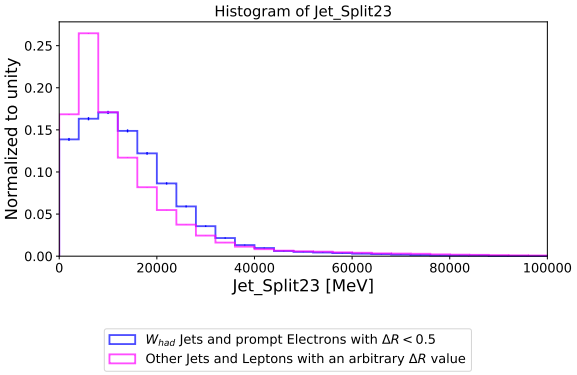
(a) Distribution of the jet C_2



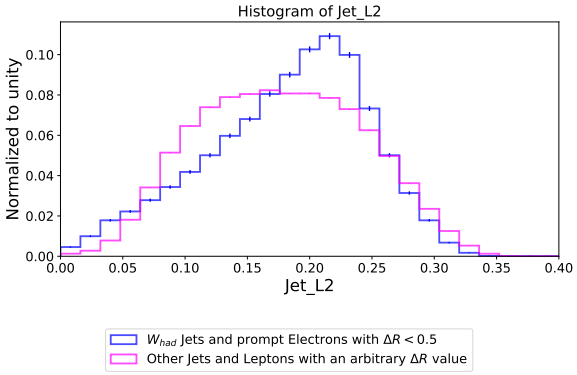
(b) Distribution of the jet D_2



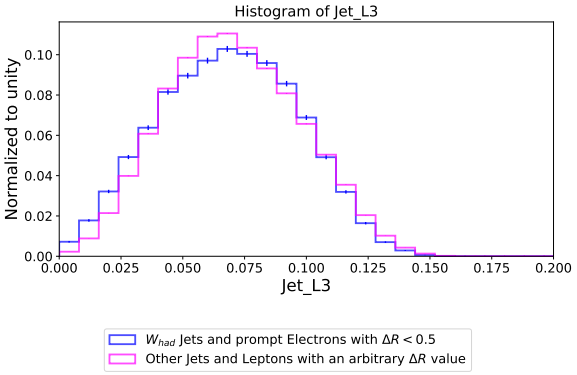
(c) Distribution of the jet Split12



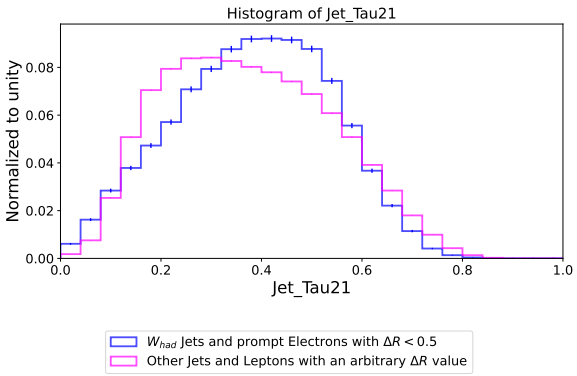
(d) Distribution of the jet Split23



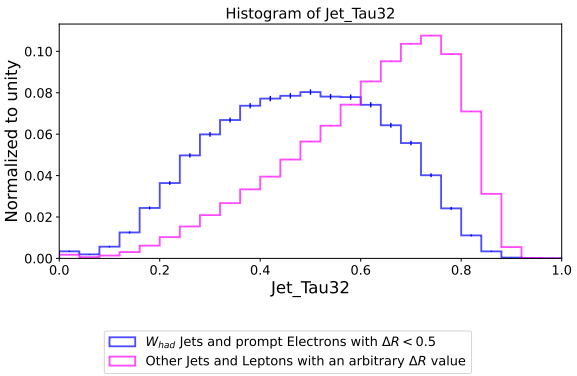
(e) Distribution of the jet L_2



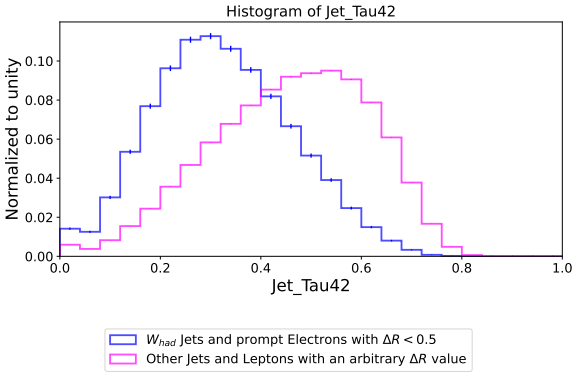
(f) Distribution of the jet L_3



(g) Distribution of the jet τ_{21}



(h) Distribution of the jet τ_{32}



(i) Distribution of the jet τ_{42}

Figure 5.7.: Distributions of various jet substructures.

Property	Description
m	Mass of jets in GeV
p_T of the lepton	Transverse momentum in GeV
τ_{21} τ_{32} τ_{42}	$\tau_{MN} = \frac{\tau_M}{\tau_N}$ with $M > N$, N-subjettiness ratio
C_2	$C_2 = \frac{ECF_3 \cdot ECF_1}{(ECF_2)^2}$
D_2	$D_2 = \frac{ECF_3 \cdot ECF_1 \cdot (ECF_1)^2}{(ECF_2)^3}$
Split12 Split23	$SplitAB = \min(p_T(A), p_T(B)) \cdot \Delta R_{AB}$
Generalized ECF Ratio L_2	$L_2 = \frac{3e_3^{\beta=1}}{(1e_2^{\beta=2})^{3/2}}$
Generalized ECF Ratio L_3	$L_3 = \frac{1e_3^{\beta=1}}{(3e_3^{\beta=1})^{1/3}}$
n_{tracks}	number of tracks within ΔR(track, jet) < 0.5, $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$
Electron_LooseID	Used for identification of the electron

Table 5.1.: Variables used as inputs for the neural network

5.5. Neural Network Setup and Training

5.5.1. Architecture

The neural network architecture is designed with a series of alternating dense and batch normalization layers. The network begins with two dense layers, each containing 512 neurons, followed by two dense layers with 256 neurons, and concludes with two dense layers of 128 neurons. All dense layers utilize the ReLU activation function and are regularized with the L2 regularizer and a regularizer rate of 0.001. Batch normalization layers are placed between each pair of dense layers to normalize the output of the preceding layer, enhancing training stability and performance. The network ends with a dense output layer featuring 2 units, activated by the softmax function to generate classification probabilities. A summary of the architecture is presented in Table 5.2. The optimization process that led to this architecture is described in Appendix B.

Layer (type)	Output Shape	Param #	Activation / Regularizer(Rate)
Dense (512)	(None, 512)	`input * 512 + 512`	ReLU, L2(0.001)
BatchNormalization	(None, 512)	`2048`	
Dense (512)	(None, 512)	`512 * 512 + 512`	ReLU, L2(0.001)
BatchNormalization	(None, 512)	`2048`	
Dense (256)	(None, 256)	`512 * 256 + 256`	ReLU, L2(0.001)
BatchNormalization	(None, 256)	`1024`	
Dense (256)	(None, 256)	`256 * 256 + 256`	ReLU, L2(0.001)
BatchNormalization	(None, 256)	`1024`	
Dense (128)	(None, 128)	`256 * 128 + 128`	ReLU, L2(0.001)
BatchNormalization	(None, 128)	`512`	
Dense (128)	(None, 128)	`128 * 128 + 128`	ReLU, L2(0.001)
BatchNormalization	(None, 128)	`512`	
Dense (2)	(None, 2)	`128 * 2 + 2`	Softmax

Table 5.2.: Final architecture of the Neural Network used for training

5.5.2. Training

The variables listed in Table 5.1 were used for training. The data was divided into two categories: the signal class, which consists of W_{had} jets and prompt electrons with $\Delta R < 0.5$, and the background class, which includes all other jets and leptons with any ΔR value.

Due to the significantly higher number of entries in the background data, it was necessary to resample the data. This means that the entries were artificially increased for the signal class to balance both classes. It is important that the signal and background classes are equally represented to prevent the model from becoming biased toward the more frequent class, which could lead to poor detection of the rarer class. Imbalanced data can also hinder model optimization and evaluation, as standard metrics like accuracy may not accurately reflect the model's true performance.

The model was compiled using the Adam optimizer with a learning rate of 0.0001. For training, categorical crossentropy was used as the loss function, and the performance metrics were accuracy and AUC. During training, early stopping was applied with a patience of 8 epochs, and learning rate reduction on plateau was used with a patience of 4 epochs. The training process was completed after 54 epochs.

5.5.3. Performance

The training curve shows a steady decrease in loss to 0.1353, while the accuracy curve rises continuously to 0.9718. Both curves with training and validation datasets can be seen in Figure 5.8. The values and the similarity of training and validation curves show that the performance of the model is very good. Significant changes due to the learning rate plan are particularly recognisable at epochs 20 and 49.

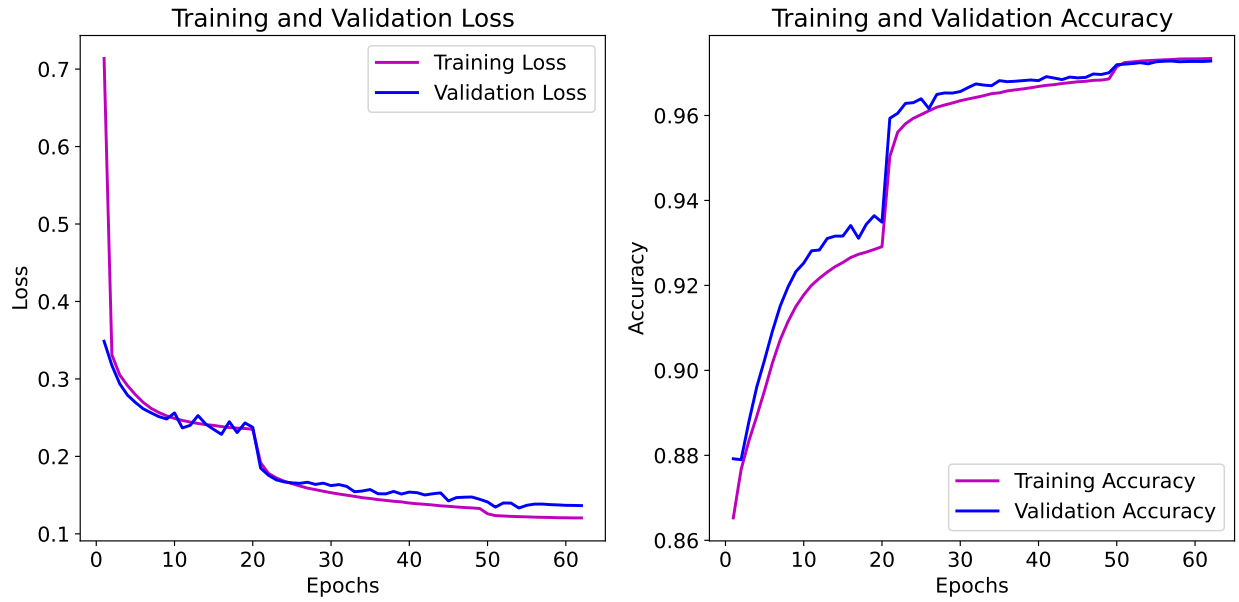


Figure 5.8.: Loss and accuracy curves for training and validation. The loss decreases to 0.1353 while the accuracy rises to 0.9718.

The probability distributions in Figure 5.9 clearly show the expected patterns for both classes (signal on the left, background on the right). The signal class predominantly consists of signal labels, while the background class mainly comprises background labels. For the signal, the peaks in the probability distributions are clearly visible, while for the background class, the distributions are slightly wider. This highlights the ability of the model to discriminate between the two classes and reflect the expected trends in the data. Figure 5.10 shows the same distribution as Figure 5.9 but with a logarithmic scale. With the logarithmic scale, the signal plot (on the left) shows a clear separation between the two classes with only a few outliers that arer so small (around 10) that they are negligible. In the background plot (on the right), peaks of signal can be observed within the background and vice versa (between 10^3 and 10^4). This is not an ideal performance. It illustrates why the signal class performs better.

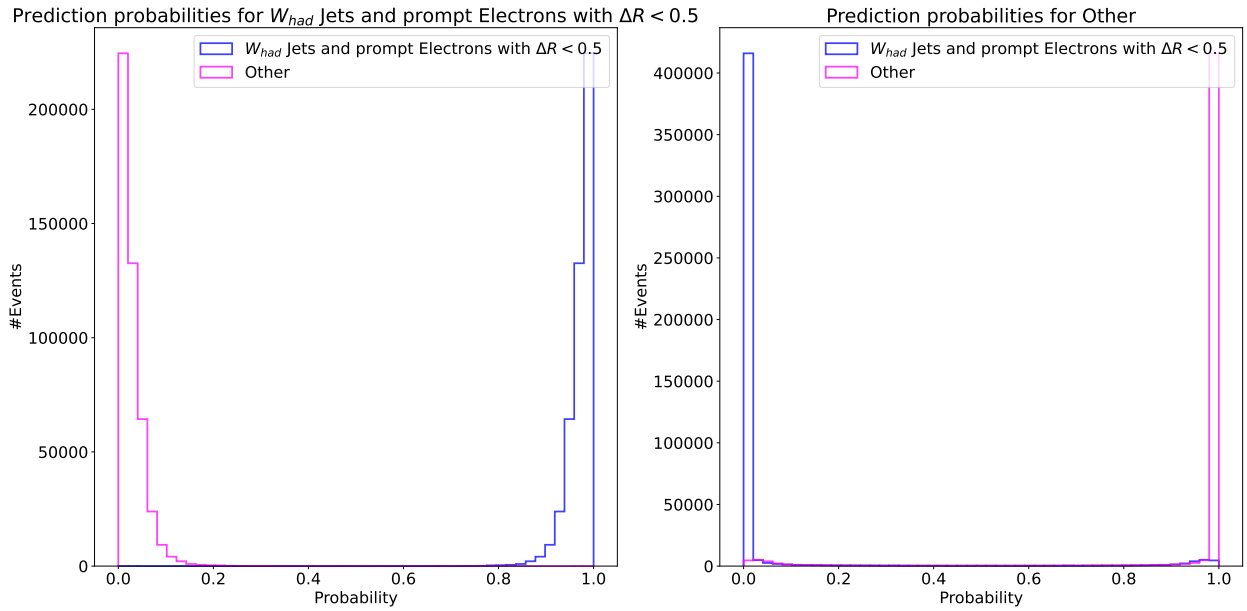


Figure 5.9.: Performance of signal and background showing that the classes can be well separated, indicating that the model works.

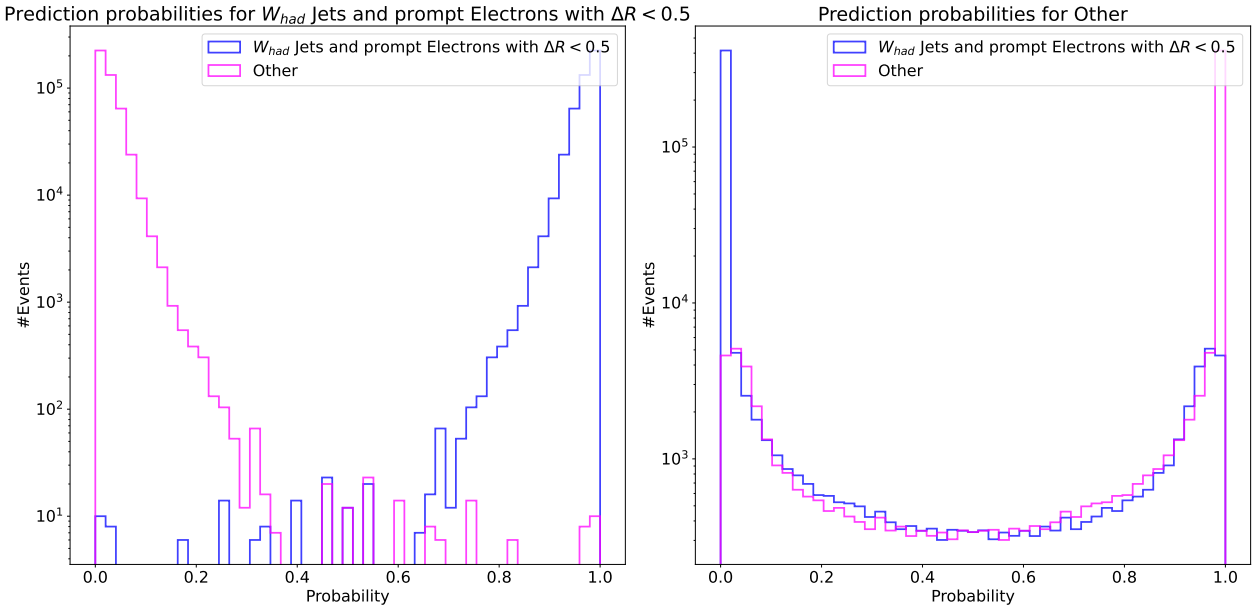


Figure 5.10.: Performance of of signal and background with log-scale providing a more detailed view on the lower range.

Figure 5.11 shows the ROC (Receiver Operating Characteristic) curve for the training data. On the x-axis, the signal efficiency, also known as True Positive Rate (TPR), is displayed. The TPR indicates the proportion of actual positive cases that are correctly identified as positive. On the y-axis, the background rejection, which is equivalent to

1 – False Positive Rate (FPR), is shown. The FPR represents the proportion of negative cases that are incorrectly classified as positive.
The model has an Area Under the Curve (AUC) value of 0.9918 for both signal and background, indicating an excellent ability to discriminate between the two. The AUC quantifies the performance of the model across all classification thresholds, with values closer to 1 reflecting greater accuracy in distinguishing between positive (signal) and negative (background) cases. This high AUC underscores the model's reliability and effectiveness in identifying relevant signals amidst background noise.

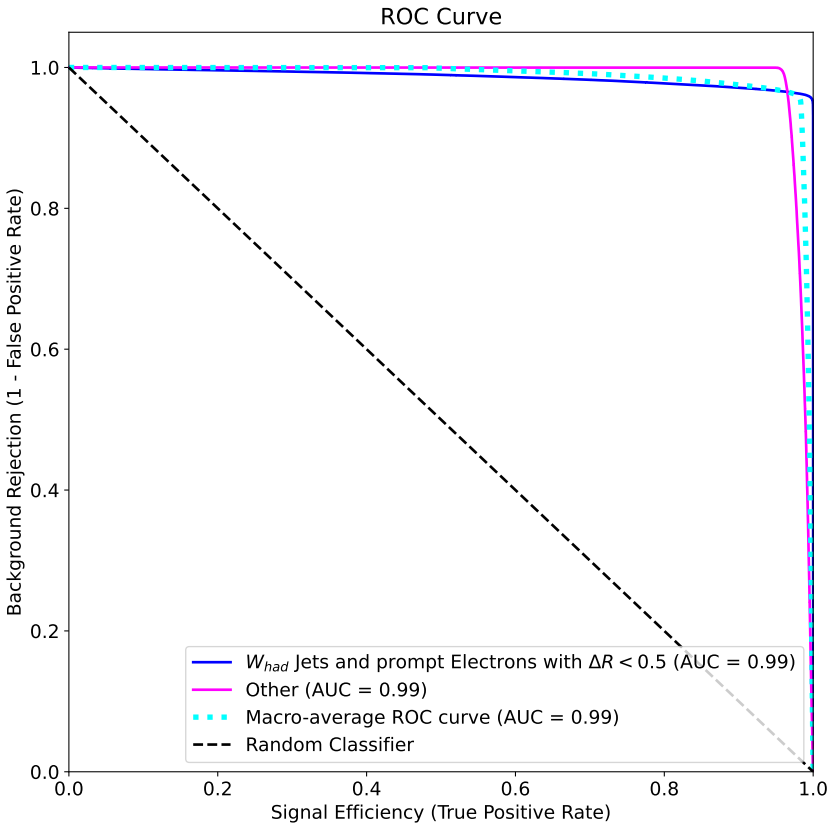


Figure 5.11.: ROC curve for training data. The solid lines represent the ROC curves for signal (blue) and background (magenta), while the dotted cyan line shows the macro-average ROC curve, which is the average performance across all classes. The black dashed line represents the performance of a random classifier, serving as a baseline for comparison. The high AUC values indicate that the model performs significantly better than random guessing.

The confusion matrix in Figure 5.12 shows a high accuracy in identifying both classes. The model correctly classifies the predicted signal as the actual signal with $99.978\% \pm 0.002\%$ accuracy and the predicted background as the actual background with $94.381\% \pm 0.034\%$ accuracy. Bootstrapping [71] was used to calculate the uncertainties.

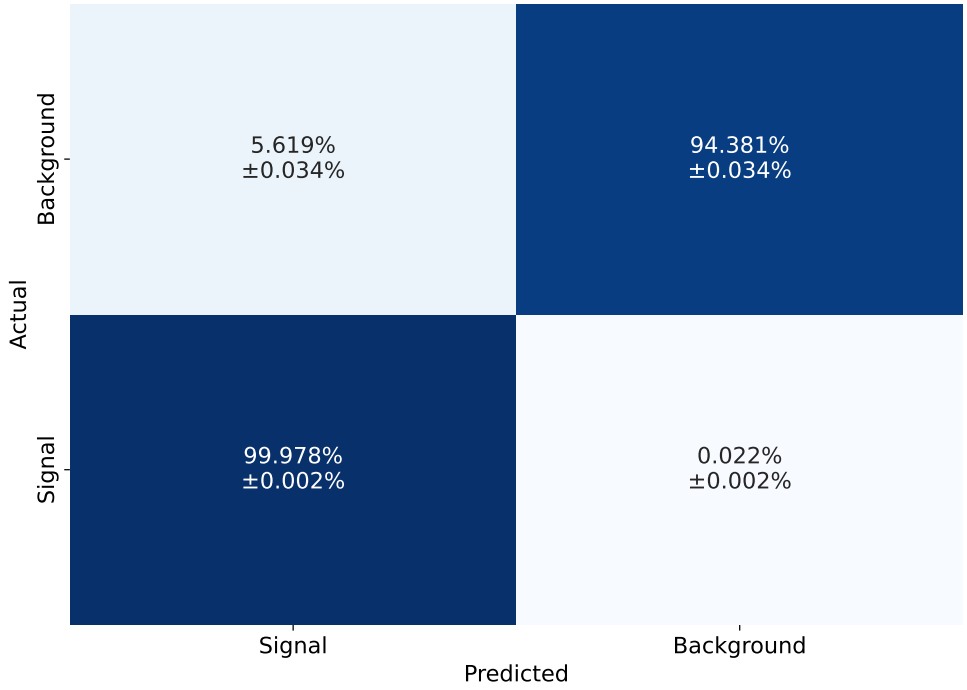


Figure 5.12.: Confusion matrix of the trained NN. The signal performs almost perfectly (99.98%), but also the background accuracy is very good (94.38%).

6. Conclusion and Outlook

This thesis focused on the challenging task of identifying electrons overlapping with jets in the context of the ATLAS search for Higgs boson pair production in the boosted $X \rightarrow HH \rightarrow b\bar{b}WW^*$ 1-lepton channel. The signal particles can be distinctly traced back to the decay of a Higgs boson, while the background particles may originate from various sources, such as QCD events or other heavy bosons. This distinction has a significant impact on several characteristics, one example is illustrated by the ΔR distribution in Figure 5.4 for both prompt and non-prompt electrons, as well as $H \rightarrow bb$ and W_{had} jets. For prompt electrons, the distribution clearly differs between W_{had} (signal) and $H \rightarrow bb$ (background) jets. In contrast, for non-prompt electrons (background), no such distinction is observed.

The developed neural network model, consisting of six hidden layers with alternating dense and batch normalization layers, demonstrated significant improvements in both the precision and effectiveness of the identification process in these high-energy environments. The model architecture utilized ReLU activation functions and L2 regularization, with batch normalization to enhance training stability and performance. Trained with the Adam optimizer and categorical crossentropy as loss function, the model employed early stopping and learning rate reduction mechanisms, resulting in a well-optimized training process.

The network exhibited excellent discriminative capabilities, achieving a final accuracy of 0.9718 and a loss of 0.1353 after 54 epochs. With an AUC of 0.9918 for both signal and background classes, the model showed a strong ability to differentiate between them. The confusion matrix indicated a high accuracy in identifying both signal ($99.978\% \pm 0.002\%$) and background ($94.381\% \pm 0.034\%$) classes. The probability distributions further confirmed the model's effectiveness in distinguishing the classes, with a clear peak for signal and a slightly more spread-out distribution for background. Overall, the model's performance in classifying W_{had} jets containing prompt electrons ($\Delta R(W_{had}, e) < 0.5$) against other jets and leptons highlights its robustness and applicability in high-energy physics experiments.

The promising results of this work suggest the potential for further integration into the

EASYjet framework [72]. This could enhance the framework's capability in classifying and analyzing W_{had} jets and prompt electrons with $\Delta R < 0.5$. Future work could focus on refining the model for broader applications within the framework. Potential extensions include adjusting the ΔR threshold, distinguishing between multiple background classes, or further improving the separation of electrons from W_{had} jets.
Furthermore there are several areas to improve and extend the model's capabilities. One important avenue is the definition of different Working Points (WP). WPs are thresholds applied to the network's output values, determining the acceptance rate of the classified objects. For example a WP of 85% means that 85% of the signal is accepted, while a stricter WP, such as 50%, results in higher purity by accepting only the top 50% of the signal. By exploring different operating points, the balance between signal purity and background rejection can be optimized, potentially leading to improved model performance in different scenarios.
My work facilitates the inclusion of the electron channel in the analysis, which had to be excluded in Run 2 due to insufficient sensitivity. This enhancement results in a significant increase in statistical power, approximately doubling the available data due to the branching ratio, although the selection efficiency still needs to be accounted for. The inclusion of the electron channel in Run 3 not only broadens the scope of the analysis, but also improves its sensitivity, allowing more precise measurements and potentially better constraints on the search for new physics. This increased sensitivity is crucial, especially as the efficiency of the electron selection will play a significant role in the final results.

Bibliography

[1] S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19**, 1264 (1967)

[2] S. L. Glashow, *Partial-symmetries of weak interactions*, Nucl. Phys. **22(4)**, 579 (1961)

[3] A. Salam, *Weak and Electromagnetic Interactions*, Conf. Proc. C **680519**, 367 (1968)

[4] S. L. Glashow, J. Iliopoulos, L. Maiani, *Weak Interactions with Lepton-Hadron Symmetry*, Phys. Rev. D **2**, 1285 (1970)

[5] H. Georgi, S. L. Glashow, *Unified Weak and Electromagnetic Interactions without Neutral Currents*, Phys. Rev. Lett. **28**, 1494 (1972)

[6] ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Phys. Lett. B **716(1)**, 1 (2012)

[7] CMS Collaboration, *Observation of a new boson at a mass of 125* GeV *with the CMS experiment at the LHC*, Phys. Lett. B **716(1)**, 30 (2012)

[8] P. W. Higgs, *Broken symmetries, massless particles and gauge fields*, Phys. Lett. B **12**, 132 (1964)

[9] D. J. Griffiths, *Introduction to elementary particles; 2nd rev. version*, Wiley-VCH, Weinheim (2008)

[10] CERN Courier, *CERN Courier March/April 2024*, CERN Courier **64(2)** (2024)

[11] K. Abeling, *Search for resonant Higgs boson pair production in the $b\bar{b}WW^*$ decay channel in the boosted 1-lepton final state using the full Run 2 ATLAS dataset*, Ph.D. thesis, II.Physik-UniGö-Diss2022/01 (2022)

[12] M. Thomson, *Modern particle physics*, Cambridge University Press, New York (2013)

[13] Cush, *File:Standard Model of Elementary Particles.svg*, Wikipedia (2019), accessed: 27.04.2024, URL `https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg`

[14] P. Langacker, *The standard model and beyond; 1st ed.*, Taylor and Francis, Boca Raton, FL (2010)

[15] M. Dine, A. Kusenko, *Origin of the matter-antimatter asymmetry*, RMP **76(1)**, 1 (2003)

[16] M. Gonzalez-Garcia, M. Maltoni, *Phenomenology with massive neutrinos*, Phys. Rep. **460(1-3)**, 1 (2008)

[17] S. Y. Choi, J. S. Lee, J. Park, *Decays of Higgs bosons in the Standard Model and beyond*, Progress in Particle and Nuclear Physics **120**, 103880 (2021)

[18] F. Englert, R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, Phys. Rev. Lett. **13**, 321 (1964)

[19] G. S. Guralnik, C. R. Hagen, T. W. Kibble, *Broken symmetries and the Goldstone theorem*, Advances in particle physics **2**, 567 (1968)

[20] J. Ellis, *Higgs Physics* (2013), `1312.5672`, URL `https://arxiv.org/abs/1312.5672`

[21] Atlas Collaboration, Cms Collaboration, *Combined Measurement of the Higgs Boson Mass in pp Collisions at $\sqrt{s} = 7$ and 8* TeV *with the Atlas and Cms Experiments*, Phys. Rev. Lett. **114** (2015)

[22] CERN, *CERN Yellow Reports: Monographs, Vol 2 (2017): Handbook of Lhc Higgs cross sections: 4. Deciphering the nature of the Higgs sector* (2017)

[23] G.C. Branco et al., *Theory and phenomenology of two-Higgs-doublet models*, Phys. Rep. **516(1-2)**, 1 (2012)

[24] T. Robens, T. Stefaniak, J. Wittbrodt, *Two-real-scalar-singlet extension of the SM: Lhc phenomenology and benchmark scenarios*, EPJ C **80(2)** (2020)

[25] E. Mobs, *The CERN accelerator complex in 2019. Complexe des accélérateurs du CERN en 2019* (2019), general Photo, Accessed: 23.04.2024, URL `https://cds.cern.ch/record/2684277`

[26] L. R. Evans, P. Bryant, *LHC Machine*, JINST **3** (2008)

[27] CERN, *Linear accelerator 4*, accessed: 24.04.2024, URL `https://home.cern/science/accelerators/linear-accelerator-4`

[28] ATLAS Collaboration (ATLAS), *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3**, S08003 (2008)

[29] CMS Collaboration, *The CMS experiment at the CERN LHC*, JINST **3(08)**, S08004 (2008)

[30] ALICE Collaboration (ALICE), *The ALICE experiment at the CERN LHC*, JINST **3**, S08002 (2008)

[31] LHCb Collaboration (LHCb), *The LHCb Detector at the LHC*, JINST **3**, S08005 (2008)

[32] G. Apollinari, O. Brüning, L. Rossi, *High Luminosity LHC Project Description*, Technical report, CERN, Geneva (2014)

[33] R. M. Bianchi, ATLAS Collaboration, *ATLAS experiment schematic or layout illustration* (2022), general Photo

[34] M. Marjanović, *ATLAS Tile Calorimeter Calibration and Monitoring Systems*, TNS **66(7)**, 1228 (2019)

[35] A. Betti, *ATLAS LAr Calorimeter Commissioning for the LHC Run 3*, NIM-A **6(3)** (2022)

[36] *Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer*, Technical report, CERN, Geneva (2017)

[37] R. Barrue, ATLAS TDAQ Collaboration (ATLAS), *The ATLAS trigger system*, Technical report, CERN, Geneva (2024)

[38] ATLAS Collaboration (ATLAS), *The ATLAS Trigger System for LHC Run 3 and Trigger performance in 2022*, Technical report, CERN, Geneva (2024)

[39] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.*, Psychol. Rev. **65(6)**, 386 (1958)

[40] T. Hastie, R. Tibshirani, J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, Springer, 2 edition (2009)

[41] A.-N. Sharkawy, *Principle of Neural Network and Its Main Types: Review*, JAACM **7**, 8 (2020)

[42] J. Zhang, C. Zong, *Deep Neural Networks in Machine Translation: An Overview.*, IEEE Intell. Syst. **30(5)**, 16 (2015)

[43] S. Sharma, S. Sharma, A. Athaiya, *Activation functions in neural networks*, Towards Data Science **6(12)**, 310 (2017)

[44] Y. Tian et al., *Recent advances on loss functions in deep learning for computer vision*, NC **497**, 129 (2022)

[45] S. Ruder, *An overview of gradient descent optimization algorithms* (2017)

[46] R. Rojas, *Neural Networks - A Systematic Introduction*, Springer, Berlin (1996)

[47] A. Zhang et al., *Dive into Deep Learning* (2023), 2106.11342, URL https://arxiv.org/abs/2106.11342

[48] GeeksforGeeks, *Vanishing and Exploding Gradients Problems in Deep Learning* (2023), accessed: 2024-07-26, URL https://www.geeksforgeeks.org/vanishing-and-exploding-gradients-problems-in-deep-learning/

[49] TensorFlow, *TensorBoard* (2024), accessed: 2024-08-07, URL https://www.tensorflow.org/tensorboard

[50] T. Yu, H. Zhu, *Hyper-Parameter Optimization: A Review of Algorithms and Applications* (2020)

[51] Keras Team, *Learning Rate Scheduler* (2023), accessed: 2024-07-28, URL https://keras.io/api/callbacks/learning_rate_scheduler/

[52] Keras Team, *ReduceLROnPlateau* (2023), accessed: 2024-07-28, URL https://keras.io/api/callbacks/reduce_lr_on_plateau/

[53] Keras Team, *Dropout* (2023), accessed: 2024-07-28, URL https://keras.io/api/layers/regularization_layers/dropout/

[54] K. Team, *Image Augmentation Layers* (2024), accessed: 2024-10-19, URL https://keras.io/api/layers/preprocessing_layers/image_augmentation/

[55] Keras Team, *BatchNormalization* (2023), accessed: 2024-07-28, URL https://keras.io/api/layers/normalization_layers/batch_normalization/

[56] S. Ioffe, C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* (2015), 1502.03167, URL https://arxiv.org/abs/1502.03167

[57] Keras Team, *Early Stopping* (2023), accessed: 2024-07-28, URL https://keras.io/api/callbacks/early_stopping/

[58] ATLAS Collaboration, *Electron and photon energy calibration with the ATLAS detector using LHC Run 2 data*, JINST **19(02)**, P02009 (2024)

[59] M. Cacciari, G. P. Salam, G. Soyez, *The anti-k_t jet clustering algorithm*, JHEP **2008(04)**, 063 (2008)

[60] A. J. Larkoski, G. P. Salam, J. Thaler, *Energy correlation functions for jet substructure*, JHEP **2013(6)** (2013)

[61] ATLAS Collaboration, *Identification of boosted, hadronically decaying W bosons and comparisons with ATLAS data taken at $\sqrt{s} = 8$*TeV, EPJ C **76(3)** (2016)

[62] I. Moult, L. Necib, J. Thaler, *New angles on energy correlation functions*, JHEP **(12)** (2016)

[63] *Improving jet substructure performance in ATLAS using Track-CaloClusters*, Technical report, CERN, Geneva (2017)

[64] ATLAS Collaboration, *In situ calibration of large-radius jet energy and mass in 13* TeV *proton-proton collisions with the ATLAS detector*, EPJ C **79(2)** (2019)

[65] ATLAS Collaboration, *Optimisation of large-radius jet reconstruction for the ATLAS detector in 13* TeV *proton-proton collisions*, EPJ C **81(4)**

[66] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP **2014(79)**, 79 (2014)

[67] R. D. Ball et al., *Parton distributions with LHC data*, Nucl. Phys. B **867(2)**, 244 (2013)

[68] C. Bierlich et al., *A comprehensive guide to the physics and usage of PYTHIA 8.3*, SciPost Phys. Codeb. **2022**, 8 (2022)

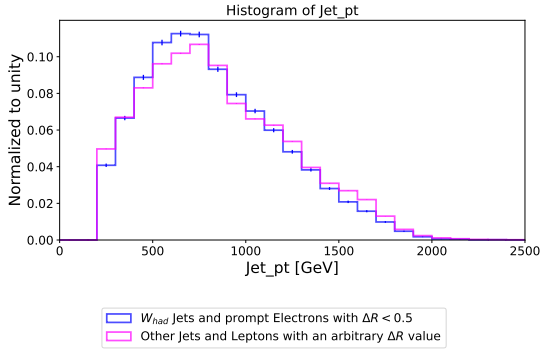
[69] *ATLAS Pythia 8 tunes to 7* TeV *data*, Technical report, CERN, Geneva (2014)

[70] D. J. Lange, *The EvtGen particle decay simulation package*, NIM-A **462(1)**, 152 (2001)

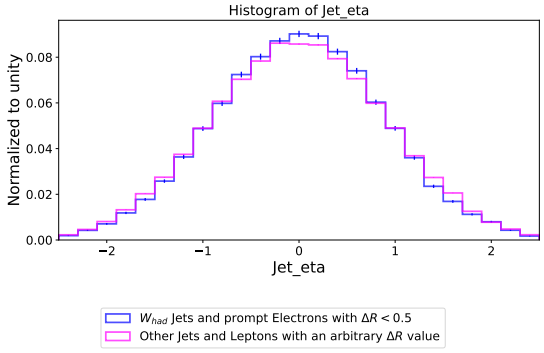
[71] B. Efron, *Bootstrap Methods: Another Look at the Jackknife*, Ann. Stat. **7(1)**, 1 (1979)

[72] EASYjet Team, *EASYjet: GitLab Repository*, `https://gitlab.cern.ch/easyjet/easyjet` (2024), accessed: 2024-08-02

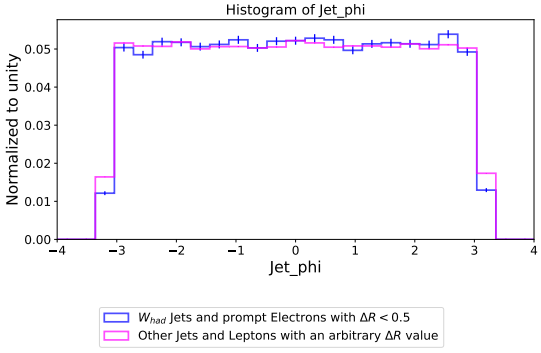
A. Variables for Selecting Training Features



(a) Distribution of the jet p_T

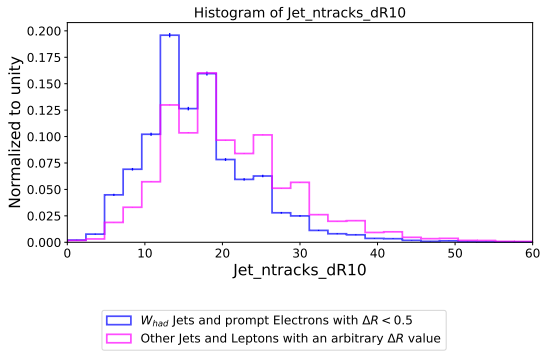


(b) Distribution of the jet η

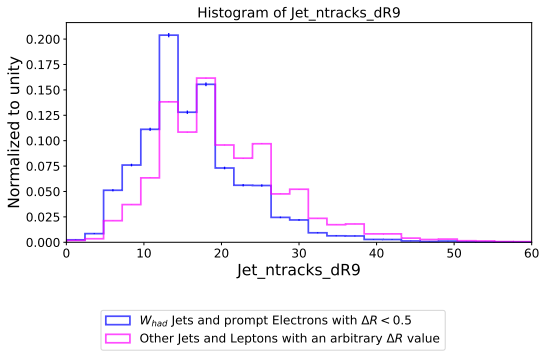


(c) Distribution of the jet ϕ

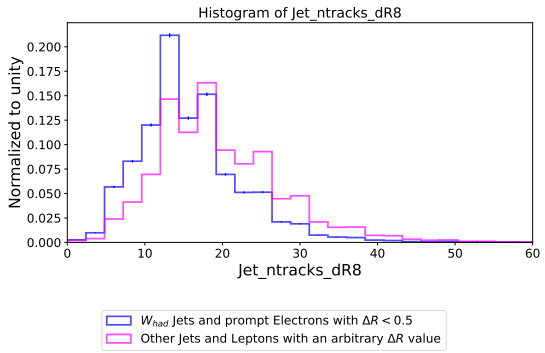
Figure A.1.: Distributions of various jet properties: transverse momentum (p_T), pseudorapidity (η) and azimuthal angle (ϕ).



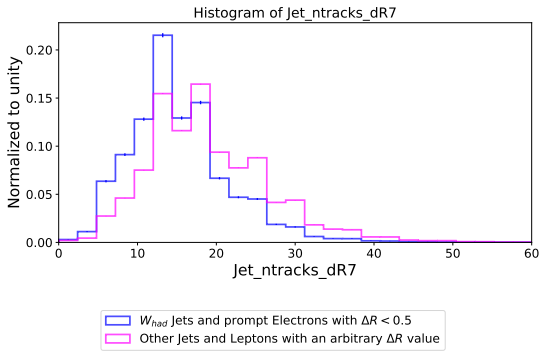
(a) Distribution of the number of tracks within $\Delta R = 1.0$



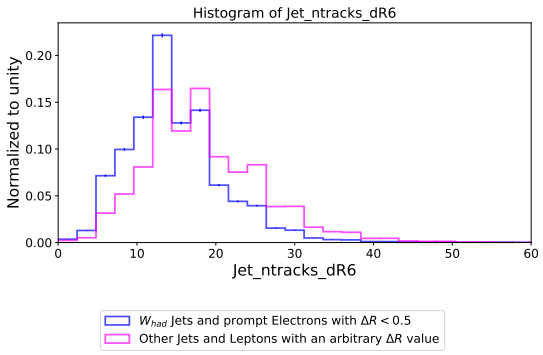
(b) Distribution of the number of tracks within $\Delta R = 0.9$



(c) Distribution of the number of tracks within $\Delta R = 0.8$

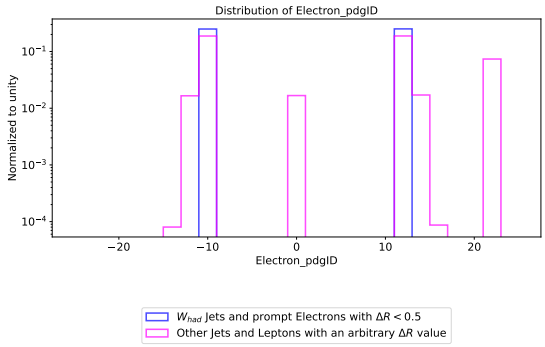


(d) Distribution of the number of tracks within $\Delta R = 0.7$

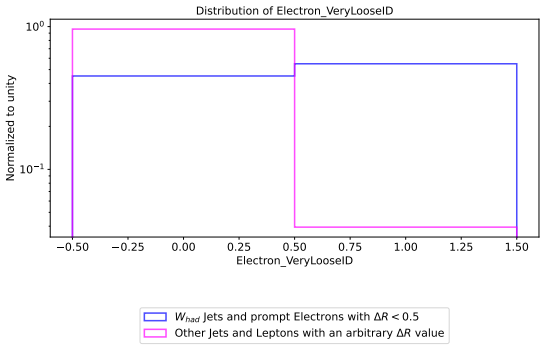


(e) Distribution of the number of tracks within $\Delta R = 0.6$

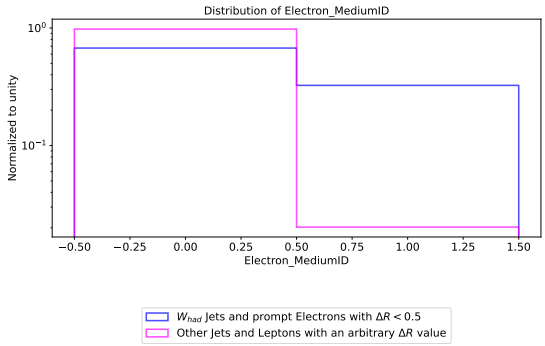
Figure A.2.: Distributions of the number of tracks within different ΔR values.



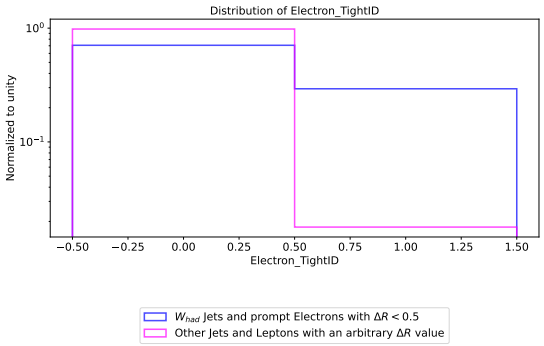
(a) Distribution of the electron PDG ID



(b) Distribution of the electron Very Loose ID



(c) Distribution of the electron Medium ID



(d) Distribution of the electron Tight ID

Figure A.3.: Distributions of various electron IDs.

B. Optimization Process

B.1. Starting Point

```
model = Sequential([
    Dense(64, activation='relu', kernel_regularizer=l2(0.01),
        input_shape=(X_train.shape[1],)),
    BatchNormalization(),
    Dropout(0.2),
    Dense(32, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.2),
    Dense(16, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.2),
    Dense(8, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.2),
    Dense(2, activation='softmax')
    )

    model.compile(optimizer=optimizer,
                loss='categorical_crossentropy',
                metrics=['accuracy', keras.metrics.AUC()])

    history = model.fit(X_train, y_train,
                        epochs=10,
                        batch_size=1028,
                        validation_split=0.2,
                        verbose=1)
```

B.2. Optimizer & Learning Rate

Optimizer	**Loss**	**Accuracy**	**AUC**
SGD	0.3537875711917877	0.8542509078979492	0.9309352040290833
RMSprop	0.6088815927505493	0.7887178659439087	0.8703969717025757
Adam	0.47098442912101746	0.820212185382843	0.8959004878997803
Adagrad	0.3350462019443512	0.8594033122062683	0.9356442093849182
Adadelta	0.31208229064941406	0.8694478869438171	0.942688524723053
Adamax	0.4103929400444031	0.8261270523071289	0.9049563407897949
Nadam	0.4756775498390198	0.797019362449646	0.8805756568908691
Nesterov	0.35934871435165405	0.8531018495559692	0.9284294843673706

Table B.1.: Performance of different optimizers for Learning Rate 0.1

Optimizer	**Loss**	**Accuracy**	**AUC**
SGD	0.3284856081008911	0.8618418574333191	0.9371823072433472
RMSprop	0.3984920084476471	0.8347687721252441	0.9137218594551086
Adam	0.37834569811820984	0.8419525623321533	0.9201933741569519
Adagrad	0.3122376799583435	0.8686411380767822	0.9427957534790039
Adadelta	0.42965081334114075	0.8451664447784424	0.921707034111023
Adamax	0.35366588830947876	0.8530381321907043	0.9301657676696777
Nadam	0.37248507142066956	0.8438640236854553	0.9220110774040222
Nesterov	0.3243614137172699	0.8642727732658386	0.9388145208358765

Table B.2.: Performance of different optimizers for Learning Rate 0.01

Optimizer	**Loss**	**Accuracy**	**AUC**
SGD	0.6797289252281189	0.834211528301239	0.9124084711074829
RMSprop	0.338821142911911	0.8583331108093262	0.935515284538269
Adam	0.3335524797439575	0.8615761995315552	0.9369778037071228
Adagrad	0.5255790948867798	0.8400086164474487	0.9173421859741211
Adadelta	1.1551693677902222	0.7992526888847351	0.8737013339996338
Adamax	0.3163285255432129	0.8690990805625916	0.9420225024223328
Nadam	0.3291243314743042	0.8627187609672546	0.9382164478302002
Nesterov	0.6818354725837708	0.8353551626205444	0.9141890406608582

Table B.3.: Performance of different optimizers for Learning Rate 0.001

Optimizer	**Loss**	**Accuracy**	**AUC**
SGD	1.385091781616211	0.73713219165802	0.810443103313446
RMSprop	0.31145766377449036	0.8702178597450256	0.9428486227989197
Adam	0.31227731704711914	0.8684834837913513	0.9423569440841675
Adagrad	1.3830777406692505	0.7553821802139282	0.818976640701294
Adadelta	1.6331415176391602	0.6121407151222229	0.6278716921806335
Adamax	0.33054694533348083	0.8618407845497131	0.9377246499061584
Nadam	0.3090340793132782	0.8705364465713501	0.9436644315719604
Nesterov	1.3619047403335571	0.7869164943695068	0.8592619895935059

Table B.4.: Performance of different optimizers for Learning Rate 0.0001

Optimizer	**Loss**	**Accuracy**	**AUC**
SGD	1.652526617050171	0.6499819159507751	0.6432902216911316
RMSprop	0.47673213481903076	0.8405119180679321	0.9178560376167297
Adam	0.4607590436935425	0.8443435430526733	0.9210081100463867
Adagrad	1.6715415716171265	0.5558832287788391	0.5733823776245117
Adadelta	1.7126731872558594	0.5226620435714722	0.5296611785888672
Adamax	0.778911828994751	0.8245341181755066	0.9017135500907898
Nadam	0.47193774580955505	0.8406955003738403	0.9176440834999084
Nesterov	1.6253985166549683	0.6686886548995972	0.6917518973350525

Table B.5.: Performance of different optimizers for Learning Rate 0.00001

Best Combinations

Number	**Optimizer**	**Learning Rate**
1	Adadelta	0.1
2	Adagrad	0.01
3	Adamax	0.001
4	RMSprop	0.0001
5	Adam	0.0001
6	Nadam	0.0001

Table B.6.: Best Performances of different optimizers and Learning Rates

B.3. Loss Functions

Loss Function	Loss	Accuracy	AUC
binary crossentropy	0.31204941868782043	0.8689262866973877	0.9425574541091919
hinge	0.6391475796699524	0.8655709028244019	0.8795554041862488
squared hinge	0.6955493688583374	0.8692415952682495	0.9176466464996338
binary crossentropy with from logits=True	0.3123341500759125	0.8682966828346252	0.9425604939460754
poisson	0.6574316024780273	0.8683236837387085	0.9417302012443542
categorical crossentropy	0.3116093575954437	0.8697038292884827	0.9432282447814941

Table B.7.: Performance of Adadelta and learning rate 0.1

Loss Function	Loss	Accuracy	AUC
binary crossentropy	0.3084441125392914	0.8707459568977356	0.9437572956085205
hinge	0.6391527056694031	0.8663700222969055	0.8849620819091797
squared hinge	0.6940346956253052	0.8693701028823853	0.9214828610420227
binary crossentropy with from logits=True	0.31017178297042847	0.87039715051651	0.9433274865150452
poisson	0.6556634902954102	0.8690580129623413	0.9430629014968872
categorical crossentropy	0.31239208579063416	0.8686994910240173	0.9422914385795593

Table B.8.: Performance of Adagrad and learning rate 0.01

Loss Function	Loss	Accuracy	AUC
binary crossentropy	0.31849533319473267	0.8659488558769226	0.9410554766654968
hinge	0.6471104621887207	0.8596096038818359	0.865862250328064
squared hinge	0.6990091800689697	0.8670385479927063	0.9144145250320435
binary crossentropy with from logits=True	0.3167499601840973	0.8673938512802124	0.9415303468704224
poisson	0.659796416759491	0.8669953346252441	0.9409213066101074
categorical crossentropy	0.3173893094062805	0.8666476011276245	0.94117271900177

Table B.9.: Performance of Adamax and learning rate 0.001

Loss Function	**Loss**	**Accuracy**	**AUC**
binary crossentropy	0.3125527799129486	0.8681336045265198	0.9424213767051697
hinge	0.640101969242096	0.8645978569984436	0.8766472935676575
squared hinge	0.694812536239624	0.8701509237289429	0.9219750165939331
binary crossentropy with from logits=True	0.3131623864173889	0.8685882687568665	0.9421939253807068
poisson	0.6565827131271362	0.8688798546791077	0.942297101020813
categorical crossentropy	0.3116426169872284	0.8686606287956238	0.9428696036338806

Table B.10.: Performance of RMSprop and learning rate 0.0001

Loss Function	**Loss**	**Accuracy**	**AUC**
binary crossentropy	0.31063875555992126	0.8703010082244873	0.9432990550994873
hinge	0.64079749584198	0.8641604781150818	0.8741347789764404
squared hinge	0.6937539577484131	0.8701163530349731	0.9174135327339172
binary crossentropy with from logits=True	0.3107384741306305	0.8691174387931824	0.9430868625640869
poisson	0.6562788486480713	0.8692545890808105	0.9427903890609741
categorical crossentropy	0.3106471300125122	0.8697502613067627	0.9432035088539124

Table B.11.: Performance of Adam and learning rate 0.0001

Loss Function	**Loss**	**Accuracy**	**AUC**
binary crossentropy	0.3119449019432068	0.8683549761772156	0.942639172077179
hinge	0.6412317156791687	0.8639444708824158	0.8742322325706482
squared hinge	0.6952899098396301	0.8684186935424805	0.9167739152908325
binary crossentropy with from logits=True	0.3111535608768463	0.8689057230949402	0.9427803158760071
poisson	0.6557345390319824	0.870040774345398	0.9431236386299133
categorical crossentropy	0.3122270703315735	0.8689597249031067	0.9424363970756531

Table B.12.: Performance of Nadam and learning rate 0.0001

Best Combinations:

Optimizer	Learning Rate	Loss Function
Adadelta	0.1	Categorical Crossentropy
Adam	0.0001	Categorical Crossentropy

Table B.13.: Best Performances of different optimizers, Learning Rates and Loss Functions

B.4. Activation Function

Activation Function	Loss	Accuracy	AUC
relu	0.31153181195259094	0.8691530823707581	0.9427415728569031
leaky relu	0.3263707160949707	0.8611021041870117	0.9363694190979004
elu	0.391489177942276	0.8291735649108887	0.9109557867050171
sigmoid	0.43084779381752014	0.8029482364654541	0.8846057057380676
tanh	0.38066011667251587	0.835237443447113	0.9164707660675049

Table B.14.: Best Performances of different Activation Functions for Adadelta and lr=0.1

Activation Function	Loss	Accuracy	AUC
relu	0.3109780251979828	0.8693625926971436	0.9430792331695557
leaky relu	0.3263750672340393	0.8612943291664124	0.9365575313568115
elu	0.3859194219112396	0.8323993682861328	0.9133470058441162
sigmoid	0.40272942185401917	0.8231118321418762	0.9035152792930603
tanh	0.3679303228855133	0.8405799269676208	0.9221805930137634

Table B.15.: Best Performances of different Activation Functions for Adam and lr=0.0001

Best: Adam and lr=0.0001 with ReLU

B.5. Dropout Rate

Dropout Rate	Loss	Accuracy	AUC
keine	0.30475589632987976	0.8762525916099548	0.9478942155838013
0.1	0.3026161193847656	0.8741175532341003	0.9461998343467712
0.2	0.31111517548561096	0.8698139786720276	0.9430739879608154
0.3	0.31955644488334656	0.8653818964958191	0.9396075010299683
0.4	0.32921102643013	0.8601430654525757	0.9359275102615356
0.5	0.3365066647529602	0.8544636964797974	0.9327797889709473

Table B.16.: Best Performances of different Dropout Rates (Adam and lr=0.0001 with ReLU)

Best: no Dropout or Dropout Rate=0.1

B.6. Regularizer

Regularizer	Loss	Accuracy	AUC
L1	0.33672958612442017	0.8638559579849243	0.9395376443862915
L2	0.30381160974502563	0.8770917057991028	0.9485225081443787
L1 and L2	0.3370113968849182	0.8639682531356812	0.939145028591156

Table B.17.: Best Performances of different Regularizer for Adam, lr=0.0001, ReLU, no Dropout

Regularizer	Loss	Accuracy	AUC
L1	0.34994253516197205	0.8564820885658264	0.9340189099311829
L2	0.3026522099971771	0.8743594884872437	0.9462410807609558
L1 and L2	0.3520844578742981	0.8546462059020996	0.9326677322387695

Table B.18.: Best Performances of different Regularizer for Adam, lr=0.0001, ReLU, Dropout 0.1

Best: L2 Regularizer

B.6.1. Regularizer Rate

Regularizer Rate	Loss	Accuracy	AUC
0.0	0.3388156592845917	0.8623602390289307	0.9377015233039856
0.1	0.3354109823703766	0.8644812107086182	0.9397829174995422
0.01	0.3398600220680237	0.8630232810974121	0.9387258887290955
0.001	0.3348984122276306	0.8655179738998413	0.9397636651992798
0.0001	0.33923402428627014	0.8600524067878723	0.9364966154098511
0.00001	0.33758437633514404	0.8653775453567505	0.9394603371620178

Table B.19.: Best Performances of different Regularizer Rates for Adam, lr=0.00, ReLU, L2, Dropout 0.0

Regularizer Value	Loss	Accuracy	AUC
0.0	0.34865057468414307	0.8569151163101196	0.9342195391654968
0.1	0.3523922860622406	0.8555857539176941	0.9330404996871948
0.01	0.35373079776763916	0.8531623482704163	0.9324083924293518
0.001	0.3475794196128845	0.8592132329940796	0.935444176197052
0.0001	0.3519290089607239	0.8556461930274963	0.9332339763641357
0.00001	0.3521043360233307	0.8552045226097107	0.9330962896347046

Table B.20.: Best Performances of different Regularizer Rates for Adam, lr=0.0001, ReLU, L2, Dropout 0.1

Best: no Dropout and L2 Regularizer with 0.001

B.7. Batch Size

Batch Size	Loss	Accuracy	AUC
16	0.31690093874931335	0.8699608445167542	0.9429407715797424
32	0.2982230484485626	0.8772925734519958	0.948190450668335
64	0.2965218424797058	0.8788087964057922	0.9494154453277588
128	0.29521670937538147	0.8787299990653992	0.9494580626487732
256	0.2985486686229706	0.8781111836433411	0.9494020938873291
512	0.310638159134735	0.8745473623275757	0.9468926191329956
1024	0.32532984018325806	0.8706488013267517	0.9445446133613586
2048	0.33819496631622314	0.8675417900085449	0.9421520233154297
4096	0.3541666865348816	0.8663009405136108	0.9403467774391174

Table B.21.: Performances of different Batch-Sizes

Best: 128

B.8. Number of Hidden Layers

Number	Loss	Accuracy	AUC
1	0.31661301851272583	0.8669348359107971	0.9407336711883545
2	0.28872454166412354	0.8824590444564819	0.9522656202316284
3	0.2791387140750885	0.8877345323562622	0.956551730632782
4	0.26821181178092957	0.8954151272773743	0.9612974524497986
5	0.26784688234329224	0.8968254923820496	0.9624056816101074
6	0.26487401127815247	0.8991906046867371	0.9642356038093567
7	0.2733747363090515	0.8980825543403625	0.9632548093795776
8	0.2747383713722229	0.8975501656532288	0.9635468125343323
9	0.2843092083930969	0.8932141661643982	0.9600677490234375
10	0.27641141414642334	0.89545077085495	0.9620593786239624

Table B.22.: Performances of different numbers of Hidden Layers

Best: 6 Hidden Layers

B.9. Neurons per Layer

Model	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
1	128	64	32	16	8	4
2	512	256	128	64	32	16
3	512	512	256	256	128	128
4	512	512	512	256	256	256

Table B.23.: Models with different numbers of neurons per layer

Model	Loss	Accuracy	AUC
1	0.27659407258033751	0.8895909786224365	0.9574908614158632
2	0.24666009843349457	0.9165020585060121	0.9728460311889648
3	0.24340121448040009	0.9264732003211975	0.9765763878822327
4	0.24374139308929443	0.9206706285476685	0.9747917652130127

Table B.24.: Performances of different models

Best: 512 → 512 → 256 → 256 → 128 → 128

B.10. Best Model

```
model = Sequential([
    Dense(512, activation='relu', kernel_regularizer=l2(0.001),
        input_shape=(X_train.shape[1],)),
    BatchNormalization(),
    Dense(512, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dense(256, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dense(256, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dense(2, activation='softmax')
```

```
])


loss = 'categorical_crossentropy'
#optimizer = tf.keras.optimizers.Adagrad(learning_rate=0.01)
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=optimizer,
              loss=loss,
              metrics=['accuracy', keras.metrics.AUC()])
```

B.11. More epochs with optimized model

```
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=4,
    verbose=1, min_lr=1e-6)

checkpoint = ModelCheckpoint(filepath='best_model.keras', monitor='val_loss',
    save_best_only=True, verbose=1)

early_stopping = EarlyStopping(monitor='val_loss', patience=8, verbose=1,
    restore_best_weights=True)

history = model.fit(X_train, y_train,
                    epochs=100,
                    batch_size=128,
                    validation_split=0.2,
                    callbacks=[checkpoint, early_stopping, lr_scheduler],
                    verbose=1)
```

→ 54 epochs

Performance:

- Test Loss: 0.1352573186159134
- Test Accuracy: 0.9718350768089294
- Test AUC: 0.9917726516723633

Acknowledgement

I would like to take this opportunity to thank everyone who made it possible for me to complete this Bachelor's thesis.
My special thanks go to Prof. Dr. Stan Lai. Thank you for giving me the opportunity to immerse myself in the fascinating world of particle physics and for giving me the chance to do my bachelor thesis in your research group.
An equally heartfelt thank you goes to Kira. I am very grateful for your constant support during the last weeks and for the patience with which you answered all my questions. Especially in the final stages I could always count on you, whether it was late at night or on the weekend.
I would also like to thank Prof. Steffen Schumann for agreeing to be the second referee for my thesis.
Last but not least, I would like to thank my parents. Without their support and belief in me, my studies would not have been possible. They have always stood behind me and encouraged me in every way, for which I am very grateful.

Erklärung nach §13(9) der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Universität Göttingen:
Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.
Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

Göttingen, den 21. Oktober 2024

(Josephine Katharina Stanitzok)