Machine Learning for Particle Identification

Candidate Number: 202817
Supervisor: Dr Jonas Lindert
Word Count: 9,100
Last Updated: May 26, 2021

University of Sussex

Abstract

This report outlines a study on the performance of machine learning algorithms in the classification of the $t\bar{t}H$ process where the Higgs decays into a $b\bar{b}$ pair and the top quarks decay semi-leptonically. The background for this analysis was W+jets, which has identical final-state particles, though there are much more dominant backgrounds for this signal. The identification of the $t\bar{t}H$ process is one of the primary goals of the Large Hadron Collider because it allows us to examine the coupling of the Higgs boson to the top quark via the Yukawa interaction, providing insights on electroweak symmetry breaking and physics beyond the standard model. Though this process is very complex, our final model managed to achieve an accuracy of $\sim$ 77%, showing promise for future applications of machine learning in its identification.

Contents

List of Tables

List of Figures

Preface

The first three chapters consist of background information on LHC Physics and Machine Learning. The content is based on my understanding of the topics, with strong influence from the cited literature and feedback from my supervisor. Some of the literature and discussion topics were suggested by my supervisor, while a subset was selected by myself, based on an independent literature review. The material in chapter 4 describes the analysis that was conducted to obtain results. The generation of the dataset as described in 4.2 was done entirely by my supervisor, whereas the rest of the chapter material was my own work, with assistance from clarification and feedback from my supervisor when my understanding was lacking or when I needed a second opinion. This includes the code on my GitHub repository[1], here most of it was written by me from scratch (often with the help of cited documentation and supervisor feedback) and a few short lines were written by my supervisor. Finally, the results in Chapter 5 are entirely original.

[1] https://github.com/johngeorgousis/jet-classifier

Chapter 1

Introduction

1.1 Introduction

The purpose of this report is to assess the ability of machine learning algorithms to distinguish the production of $t\bar{t}$ quarks along with a Higgs boson where the top quarks decay semi-leptonically and the Higgs decays into a $b\bar{b}$ pair (signal), from the frequent background of $W+$ jets (background). The $t\bar{t}H$ channel is of particular interest because it allows us to investigate electroweak symmetry breaking via the Yukawa coupling of the top quark to the Higgs boson (Dawson, 2001). However, this is a very complex process with a large background with identical final-state particles, making it a particularly challenging classification task. The application of machine learning in this test case is lacking, so there could be potentially promising results.
The remainder of this chapter will provide a brief overview of the report, while Chapter 2 and Chapter 3 will provide more detailed background information on the two main themes of this project: LHC Physics and Machine Learning.

1.2 LHC Physics

1.2.1 LHC

Primary Goal
The primary goal of particle physics is to help us improve our understanding of the most elementary physical phenomena. This is often done through the production of rare and insightful physical processes that we can observe and analyse. Analyses of these processes have the potential to lead to more accurate measurements and new discoveries on dark matter, the Higgs boson, electroweak symmetry breaking, physics Beyond the Standard Model (BSM) and more (Paganini, 2019). Particle colliders, such as the Large Hadron Collider (LHC), are essential instruments in creating the conditions required for these processes to take place.
Particle Colliders
The way they achieve this is by accelerating particles (often pairs of protons) to speeds very close to the speed of light and then have them collide with one another. This leads to an extremely high-energy environment that can give rise to a variety of processes that often involve heavy particles such as top quarks and the Higgs boson.
Insights from Processes
The study of these heavy particle interactions is essential for deepening our understanding of cutting-edge physics. For example, the production of the Higgs boson along with a top and an anti-top quark (referred to as $t\bar{t}H$) is important in understanding electroweak symmetry breaking (Santos et al., 2017), since it involves the Yukawa coupling of the top

quark and the Higgs boson.

Many processes in hadronic collisions manifest themselves as high-energy physical objects that we call *jets* (Paganini, 2019), which are measured by the collider's detectors. Thus, due to their ubiquity, jets will also be a main topic of discussion.

An initial problem with processes such as $t\bar{t}H$ is that they are rare, meaning that we would need a great number of collisions to get the chance to observe them. For example, at the LHC, approximately one in a trillion collisions will produce a Higgs boson (Fermilab, 2021). However, this problem is solved by having an equally high frequency of proton collisions; namely, up to about one billion collisions per second take place inside the LHC (CERN, 2021b).

1.2.2 Signals and Backgrounds

Problem

A more significant problem is the difficulty in correctly identifying important processes, i.e. actually observing them when they happen. For each proton collision, there is a multitude of different processes that can occur.

Signal-Background Classification

In experimental particle physics terminology, we refer to insightful processes that we wish to examine further as **signals**. Signals are processes that physicists consider important as they can deepen our understanding of certain physical theories and phenomena (such as physics Beyond the Standard Model). Processes that are not considered interesting for the purposes of an experiment are referred to as **backgrounds**. For example, in our case, $t\bar{t}H$(bb) with semi-leptonic top decays is our signal and W+jets constitute our background (as W+ jets are backgrounds to $t\bar{t}H$ as well as many searches in physics [46]).

Solution

A key goal of collider physics, and the goal of this project, is to find ways to effectively distinguish between signals and backgrounds (a task known as *signal-background classification*). The problem arises when a given signal process yields an identical final-state to the background process. Then, it can become extremely difficult to reliably extract the specific signal from a large background. What makes matters worse, is when the background occurs at a much higher frequency than the signal, demanding an even higher accuracy in correctly classifying the two. Two popular approaches in tackling this problem are the *Matrix Element Method* (MEM) and *Machine Learning* (ML).

1.2.3 Matrix Element Method

The Matrix Element Method (MEM) is a method used in multivariate analyses which makes use of experimental results as well as existing theoretical knowledge Wertz, 2016. MEM has been used specifically in the search for the $t\bar{t}H$ process either by itself or in combination with ML (Wertz, 2016).

Though MEM and ML have applications in many areas, in this report we will focus on one use case that is shared between the two: distinguishing between two hypotheses (e.g. an event originating from a signal or a background).

MEM makes use of the *likelihood ratio* along with the Neyman–Pearson lemma (Wertz, 2016). This requires the computation of conditional probability $P(\mathbf{x}|\alpha)$ for the observation of an event $\mathbf{x}$ under a theoretical hypothesis α, which further requires the evaluation of a non-trivial, multidimensional integral.

One core advantage of this method is that it is not limited by the amount of available events in a dataset, which can be an issue in searches for rare processes (Wertz, 2016). In addition, MEM is a more *analytical* approach, while ML is almost purely *computational*.

This makes MEM more transparent, allowing us to gain more intuition on the underlying physics, whereas an ML model is often viewed as a *black box*, meaning that useful information of its internal workings is practically impossible to obtain (this especially applies to Artificial Neural Networks). However, despite its downsides, machine learning tends to outperform other methods in signal-background classification and is what will be the focus of this report.

1.3 Machine Learning

1.3.1 Overview

In the 2000s, Machine Learning methods started becoming increasingly popular in particle identification due to a sudden rise in computing power and increased research in that area (Bourilkov, 2019). Indeed, in 2012, a machine learning algorithm called "Boosted Decision Trees" assisted in the discovery of the Higgs boson (Bourilkov, 2019).

Machine learning involves a variety of computer science, data analysis, and statistical techniques applied to large amounts of data. The goal is to generate an algorithm that models the data sufficiently well, such that it is able to make predictions on unseen data, without defining a specific algorithmic procedure as is traditionally the case in computer science.

There are different types of Machine Learning based on the structure of the problem in question, but the one that is of interest to us is *Supervised Learning* which is based on the idea of a computer programme that can learn from experience. What that actually means is that an algorithm is shown examples of the objects we want it to be able to make predictions on (e.g. particle collision data) and then progressively tunes itself to be able to make better and better predictions. A prerequisite for this process to be possible is the availability of training data: previous examples that have been labelled correctly so that the algorithm can see examples of correct predictions.

Dataset. In particle physics, the generation of datasets is essential (Paganini, 2019). This is done using randomised methods for Monte Carlo sampling, yielding large *Monte Carlo datasets*. Much effort is put into ensuring that these data accurately resemble real LHC collisions (Paganini, 2019). In our analysis, the dataset was generated using Sherpa, a popular, general-purpose event generator [41, 48]. In the case of signal-background classification, the training data consists of events that are labelled based on whether they are signal or background events. The goal of the model is, given a labelled dataset of events, to learn to predict the label of events it has not seen before.

1.4 Aim of the Project

The ultimate purpose of this project is to examine how various ML algorithms compare to one another and to see how well our best model performs in the signal-background classification of $t\bar{t}H$ and W+jets. In the next two chapters, we will go into more detail about the relevant physics and machine learning theory.

Chapter 2

LHC Physics and Jet identification

2.1 LHC physics

2.1.1 Why Particle Colliders?

As is the case with all theories in physics, theoretical predictions must be directly compared to experimental results to verify that there is agreement between theory and reality. After a point in the development of elementary particle physics and its dominating theory, the Standard Model, there grew a need for larger apparatus that could produce extremely high energies, replicating the conditions shortly after the Big Bang. Popular machines known as *particle colliders* had the potential to meet these requirements, which lead to their massive upscaling, with the most prominent example being the Large Hadron Collider (LHC) at CERN, the European Organisation for Nuclear Research. The work that is conducted at CERN is regarded as "The World's Largest Experiment" with over 10,000 scientists and hundreds of universities taking part (Paganini, 2019, Seiberg, 2016).

2.1.2 LHC Acceleration

The LHC is a 27-kilometre ring that uses magnetic fields to accelerate particles to ultra-relativistic speeds (CERN, 2021d). In each operation, trillions of particles circle the ring about 11, 245 times per second (CERN, 2021a). Two high-energy proton beams are boosted to speeds close to the speed of light, travelling in opposite directions in an ultrahigh vacuum. Their paths are precisely controlled by more than 50 types of superconducting electromagnets, exploiting the positive charge of the proton. In order for the magnets to maintain their superconducting properties offering virtually zero resistance and loss of energy, they are cooled down to temperatures close to absolute zero with the use of liquid helium (CERN, 2021d). This allows the electromagnets to produce a magnetic field of 8.33 tesla, more than 100,000 times more powerful than the Earth's (CERN, 2021a), which is sufficient for controlling the ultra-fast beams of protons. When these beams are made to collide, we observe energies at the TeV regime. Though these collisions are not energetic enough for evaluating theories such as the Grand Unified Theory (GUT), they are sufficient for the study of electroweak symmetry breaking (Paganini, 2019), leaving room for many exciting discoveries.

2.1.3 LHC Detectors

Along with a large cooling system and thousands of magnets to control the beams, more tools are needed to observe what happens after each pair of hadrons collide. An essential component of particle colliders are *detectors* which measure the outcome of the physical processes that occur in the volume of the collider, in order to later determine the type

of process itself. After a hadron collision, final-state particles interact with the walls of the tube, depositing their energy to the detectors that absorb them. *Event reconstruction* refers to the process of analysing the electronic signals of a detector to infer the type of particle or physical object that passed through the detector along with its properties (e.g. momentum, direction, position) (Paganini, 2019). This is possible because each particle interacts differently with the sub-components of detectors, leaving unique signatures. This is illustrated in Fig. 2.1. which shows the average signature that each particle type yields as it interacts with the different layers of a wedge of the ATLAS detector (Paganini, 2019).

Reconstruction Algorithms To perform event reconstruction, the LHC employs *reconstruction algorithms* that are made for each type of detector (Paganini, 2019). These algorithms simplify the raw signals output by the detectors, mapping them to a physical object with specific features (e.g. photons, electrons, **jets**), and finally turning it to tabular, particle-level data which can be directly analysed using statistical methods (Paganini, 2019). Detectors along with reconstruction algorithms are what turn an entirely physical process into the 90 petabytes of data that the LHC produces each year (CERN, 2021b). Possible reconstructed physical objects include jets, photons, electrons, muons, and taus. Sometimes, using conservation of energy and momentum, we observe that there is energy that has not been detected by the detector. This is referred to as *missing energy*, and is used to infer the existence of weakly-interacting particles such as neutrinos, and is also the signature of many theories Beyond the Standard Model (Aad et al., 2014). However, in our analysis, the focus will be on **jets**.

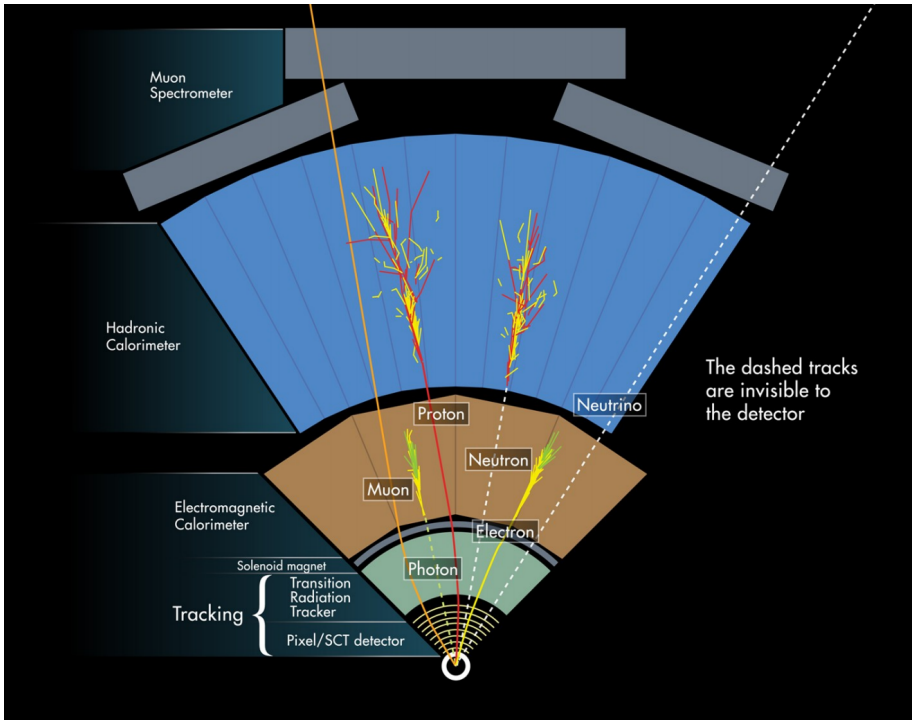

Figure 2.1: Diagram illustrating how elementary particles interact with the ATLAS detector at CERN. The nuances of each signature are used to identify the corresponding particle (Paganini, 2019).

2.2 Jets: definition, algorithms, classification

2.2.1 Definition

The specific physical phenomena that we will be analysing in our data are called *jets*. These are physical objects that appear very frequently in high-energy hadron collisions inside particle colliders (Paganini, 2019). Loosely defined, a jet is a narrow cone of final-state (post-collision) particles such as pions, kaons, and baryons. After the collision of protons in particle colliders, there is a production of their constituents, quarks and gluons, together referred to as *partons*. The resulting cascade of radiation is called a *parton shower*. These partons then hadronise before they decay to form baryons and mesons. Finally, these hadrons are boosted together to form narrow cones of particles, which are the so-called jets. An important subset of these jets that are studied in our analysis are referred to as *fat jets*, which occur when those decay products originate from heavy particles (such as the top quark) and are collimated into one large and massive jet (Marzani, Gregory Soyez, and Spannowsky, 2019). Thus, it is often these jets that event reconstruction algorithms yield. The goal then is to map these jets to the processes that caused them, since the features of each jet correlate with the features of the original process. However, rigorously defining a jet is not trivial.

2.2.2 Jet Clustering Algorithms

Strictly speaking, jets are not uniquely defined. Instead, there are different algorithms, known as jet-clustering algorithms, which are implemented to identify detector signals as jets with various properties. These algorithms vary in how they localise a given jet and how they use the energy depositions from final-state particles to determine the properties of the jet. For example, it is typically required to specify a radius parameter R, which defines the width of the jet, often referred to as the *jet radius*. Fat jets typically span a radius of $R = 1$ while smaller sub-jets are often defined with $R = 0.2$ (Paganini, 2019). For our $t\bar{t}H$ process, we have chosen a radius of $R = 1.5$.

In jet clustering algorithms, there are two broad categories for jet construction and representation: IRC-safe and Machine Learning algorithms. IRC-safe algorithms have two important properties: infrared (IR) and collinear (C) safety.

2.2.3 IRC-safe Clustering Algorithms

IR safety means that the deduced jet properties do not vary under the addition of infinitely soft radiation, while C safety guarantees invariance under the substitution of any parton with a series of collinear partons with identical total momentum (Paganini, 2019). Theoretically, IRC safety guarantees the calculation of important quantities (e.g. cross-section).

2.2.4 ML Clustering Algorithms

A popular alternative to IRC-safe algorithms are machine learning algorithms that learn an optimal representation of the boundaries of a jet, instead of analytically determining it (Paganini, 2019). The branch of machine learning that is used in this domain is called *Unsupervised Learning*. In unsupervised learning, similar to supervised learning, a general algorithm is chosen that will train on a certain task. This time, however, the algorithm does not require labelled data and is instead given an unlabelled dataset. In a process known as *clustering*, the algorithm performs mathematical operations to assign each input into a generic group that can be defined based on, say, the distance between data points in a given space. In jet clustering, these groups would define the boundaries of a jet based on the detector data and the parameters of the model.

2.2.5 Jet Classification Algorithms

Another set of algorithms that have seen a steep rise in use in the past two decades (Bourilkov, 2019) are *jet classification algorithms*. Jet classification algorithms are algorithms that label jets depending on the type of particle that they originated from. This process is known as *jet tagging* and is the ultimate machine learning task of this analysis. Popular areas of jet tagging include flavour tagging, boson tagging, and quark-gluon tagging. Boson tagging classifies jets initiated by the hadronic decay products of weak or Higgs bosons, while quark-gluon tagging attempts to discriminate the jets initiated by light quarks from those initiated by gluons (Paganini, 2019).

However, in this report, the focus is on flavour tagging, and specifically **top tagging**, which involves the labelling of jets that have originated from processes involving top quarks. These jets, as previously mentioned, are often fat jets, and are essential for BSM, Electroweak Interaction analyses and more(Paganini, 2019).

Chapter 4 will elaborate on how top tagging works in practice, while the following section will give more insight into the underlying physics of our classification process.

2.3 The Electroweak Interaction and Our Process

While jets are the high-level objects that we actually observe, what we are interested in are the underlying particle processes that produce these jets. As mentioned earlier in this report, one of the most important particles in these events is the Higgs boson due to its relevance in electroweak symmetry breaking (EWSB)Plehn, 2012.

2.3.1 The Electroweak Interaction and our Signal Process

In the 1970s, it was discovered that two fundamental forces, electromagnetism and the weak force, would be combined into one, namely, the electroweak force (CERN, 2021c). This is now described by the larger theoretical framework of the Standard Model. However, there was an issue with the initial formulation of this unified theory, since two force carriers that we know to have mass (namely the W and Z bosons), were predicted to be massless. To solve this issue, physicists Robert Brout, Francois Englert, and Peter Higgs proposed what is known as the Brout-Englert-Higgs mechanism. This mechanism gives mass to the W and Z bosons as they interact with an invisible field that pervades all of space, which we now call the *Higgs field*. In fact, the Higgs field is what gives mass to every particle in the universe, as long as it interacts with it, in a process governed directly by EWSB (Collaboration, 2019). And, as every field, it has an associated particle known as the *Higgs boson*. Thus, if we wish to study EWSB, it is essential that we analyse how the Higgs boson (something that we can indirectly observe) interacts with other elementary particles.

2.3.2 Higgs Production Modes

Production Modes

Given the significance of the Higgs boson in physics, it is important to know how and when it is produced. There are many production modes for the Higgs, each with differing cross sections, as shown in figure 2.2. The vertical axis shows the predicted cross section σ for the Higgs, which is a measure of the probability that a Higgs boson will be produced from a given process. The larger the value for σ, the more likely it is that a corresponding Higgs production process will occur. This is plotted versus the Higgs mass from 100GeV to 500GeV (Note: the measured Higgs mass is approximately $M_H = 125$GeV (CERN, 2019)).

While gluon-gluon fusion (shown in green) is the dominant process for Higgs production, the one that we aim to analyse is the production of $t\bar{t}H$ due to its insights on EWSB (Santos

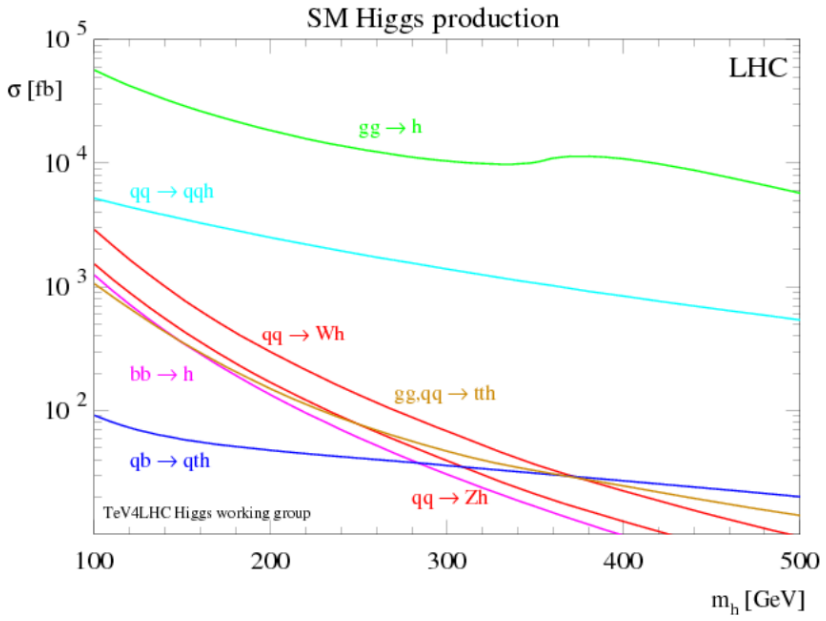

Figure 2.2: Production cross sections as a function of the Higgs mass (Ilisie, 2011).

et al., 2017).
As shown in Fig. 2.2, $t\bar{t}H$ production has one of the smallest production rates for $M_H = 125\text{GeV}$, making it more challenging to extract against a larger background. To better visualise our production mode of interest, an example Feynman diagram for $pp \to t\bar{t}H$ production is shown in figure 2.3.

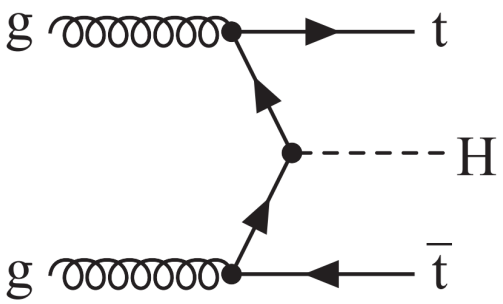

Figure 2.3: Example diagram for $t\bar{t}H$ production. Where g represents a gluon, t a top quark, and H a Higgs boson (Sirunyan et al., 2018).

Having discussed the Higgs production modes and $t\bar{t}H$, the next subsection will focus on the decay modes for the Higgs boson and the top quarks in our signal process before arriving at our signature.

2.3.3 Higgs and Top Decay Channels

Higgs Decay Channels

Like every unstable particle, the Higgs boson decays into more stable particles, which happens very shortly after its production. The different ways the Higgs can decay are called decay modes (or *decay channels*), and they're what ultimately results in the jet image that we observe. Thus, understanding them is essential to identifying the original process that took place. The probability of a particle to decay via a given channel is called a *branching ratio*. The branching ratios for a Higgs boson of given masses are shown in Fig. 2.4.

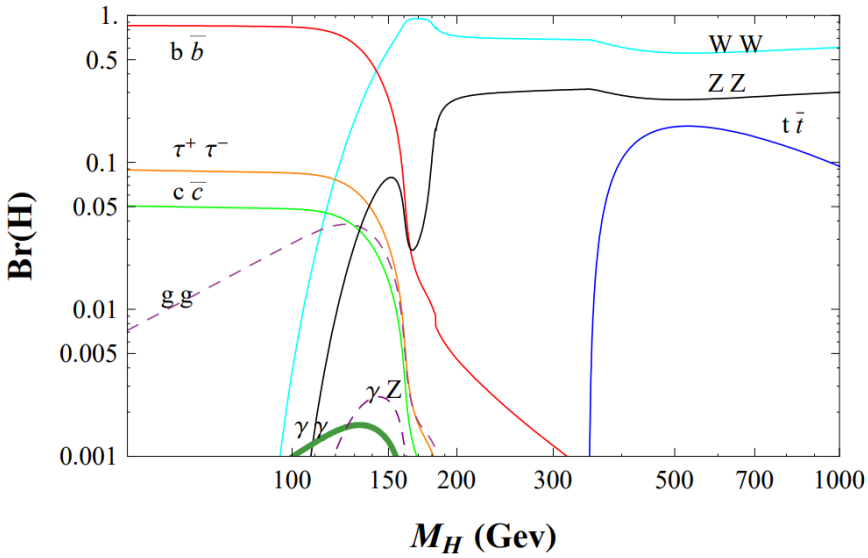

Figure 2.4: Branching ratios for all known Higgs decay channels as a function of the Higgs mass (Ilisie, 2011). Notice that for $M_H = 125$GeV, $b\bar{b}$ is the dominant decay mode with WW being the second most prominent.

While $\gamma\gamma$, ZZ, WW, $\tau\tau$, and $b\bar{b}$ decay modes for Higgs have been observed, the $t\bar{t}$ decay mode is kinematically impossible (Sirunyan et al., 2018). This is the main reason why the study of the $t\bar{t}H$ process is essential in examining the uniquely massive top quark under the Yukawa interaction, and thus, gaining further insight into the dynamics of EWSB. As we can see in the figure, the $b\bar{b}$ decay channel is the dominant channel ($Br \approx 58\%$) for a Higgs mass of $M_H = 125$GeV, making it the most promising channel for probing Higgs-top coupling. Thus, it is the chosen decay channel for our signal process.

Top Quark Decay Channel

Contrary to the Higgs boson, the top quark has only a single (known) decay channel[1]. It can decay through the weak interaction, producing a W-boson and a bottom quark. In the next subsection, we will discuss our specific signature that is obtained through the Higgs and top decays.

[1] http://www.scholarpedia.org/article/Properties_of_the_top_quark

2.3.4 Signature and Background

Signature

To finalise our signature and define our signal, we need to specify the decay modes of the W bosons resulting from the top decays. A W^+W^- boson pair has three decay modes:

1. Two leptons and their corresponding neutrinos $\ell\bar{\nu}\bar{\ell}\nu$ (the **leptonic channel**)
2. Two quarks and antiquarks $q\bar{q}q\bar{q}$ (the **hadronic channel**)
3. A lepton, a neutrino, a quark and an anti-quark $\ell\bar{\nu}q\bar{q}$ (the **semi-leptonic channel**)

For our signal process, we chose the semi-leptonic channel with a branching ratio of approximately 0.45 (Couchman, 2002). The Feynman diagram for our complete signal process along with its signature can be seen in Fig 2.5.

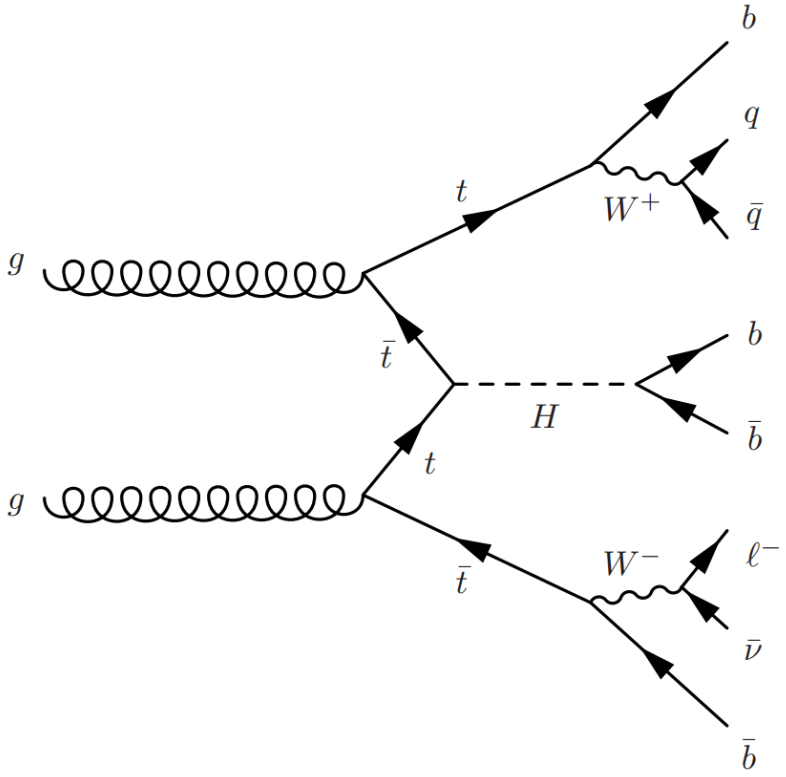

Figure 2.5: Feynman diagram of our signature. Note that the neutrinos cannot be detected by the detectors at LHC due to their elusive nature so they are recorded as missing transverse energy.

As we can see, our signal (Santos et al., 2017) process has a very complex final state with 4b + 2q + l.

Background process

The background process that the $t\bar{t}H$ top tagging will be classified against involves W+ jets, which is a background to many searches for new physics (Englert et al., 2011). As can be seen in Fig. 2.6, W+jets and our signal have identical final-state particles.

In addition, while W+jets is not the largest background for our signature (instead, $t\bar{t}$+b-jets is), it is still large enough to be considered (Guindon, 2017). Because of this, distinguishing $t\bar{t}H$ with $t\bar{t}$ decaying semi-leptonically from W+jets is particularly challenging, which makes it an important background to this particular signal.

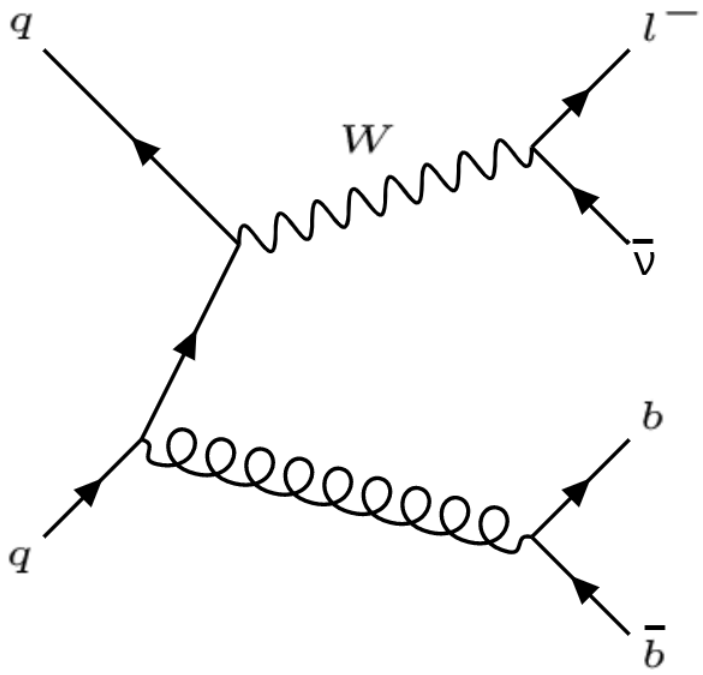

Figure 2.6: Example Feynman diagram for our W+jets background (Verkerke, 2017). Note that more jets than shown here could arise, which were included in our event samples.

In an attempt to correctly classify the two, we will use tools from the field of *machine learning*. Thus, having understood the background physics, we will move onto a comprehensive overview of ML in the following chapter.

Chapter 3

Machine Learning

3.1 Introduction

The term Machine Learning describes a class of algorithms and techniques that enable computers to learn how to perform specific tasks from experience, without being explicitly programmed. This is achieved by combining Computer Science techniques and statistical modelling methods to create generic optimisation algorithms. This property of generality allows these algorithms to be used in various contexts, unchanged. Therefore, although Machine Learning is generally thought of as a Computer Science field, it has applications in a huge variety of disciplines such as finance, healthcare, engineering, and physics (Marr, 2018). Generally speaking, most fields that produce large amounts of data and require inferences can benefit from Machine Learning.

3.2 Formal Definition

In machine learning, the ultimate goal is to create a computer programme (model) that makes accurate inferences in a given context (e.g. predicting the stock market). ML models use multi-purpose algorithms that are based on computational techniques from mathematics and statistics, but machine learning also draws on results from philosophy, neurobiology, artificial intelligence (AI), and other fields (Mitchell, 1997). In AI terminology, ML models are thought of as *intelligent agents* that are characterised by *inductive learning*. Inductive learning is the process of discovering general patterns by observing examples (Russell, 1995). As defined by Tom Mitchell in 1997, "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." (Mitchell, 1997). In essence, this means that an ML model uses data (experience) and a given performance metric to learn how to make accurate predictions on a given task, without being explicitly programmed to work on that task. The model also starts with zero knowledge of the patterns in the task, i.e. its initial performance is random. As an example, in our HEP case, the experience E is the data from particle collisions, and the task T is signal-background classification. The performance measure is key in optimising the model, and it involves the minimisation of an *error function* that is internal to the model.

3.3 Machine Learning Models

A Machine Learning model can be represented as a mathematical function that takes one or more input variables, applies a set of coefficients (weights) to each one, and performs a sum on the results to produce an output. These functions can become extremely complex,

with millions of parameters and can even consist of multiple layers of inner functions that transform the input before passing it to the next function in the chain. In practice, these functions are represented as algorithms that can be implemented by computer programmes. Their inputs represent the object we want to perform inference on (e.g. information on final-state particles) and the output represents the prediction (e.g. signal or background). The value of the prediction can take the form of a probability in classification tasks, or a continuous value in the case of regression tasks. This report focuses on classification, as this is the type of task that our use-case belongs to.

3.4 The Role of Data in Machine Learning

The prerequisite for any ML task is access to a relatively large dataset (training data) that constitutes the "experience" the ML model will learn from. In most ML use-cases, training data is represented in the form of a table with rows and columns. Each row of the dataset is called a training example (e.g. an event) and each column is called a feature or variable (e.g. transverse momentum of a particle). Training examples represent the object that the ML model is tasked with making predictions on. As an example, in the popular case of classification of images of hand-written digits[1], each training example is a set of pixel values that represent a single image of a digit that the model could be asked to classify. In this case, each individual pixel makes up a feature in the dataset.

The way the ML model makes a prediction on a training example is by mapping each feature of that example onto each of the model's input variables (which is why features need to take numerical values) and applying its weights to compute an output. This output represents the prediction that the model has made about this training example. In the case of classification of hand-written digits, each prediction could be a number from 0 to 9, representing the digit the model believes the image represents. In reality, the model would produce 10 separate outputs in the form of probabilities that represent the likelihood that the given training example belongs to each of the categories (classes) we care about, which in this case are the digits 0-9.

It is a general rule in ML that the more data that is available to the model during training, the more accurate predictions the model will make. However, there are many properties of a dataset besides its size that constitute its quality and applicability to a specific ML use case. Data often needs to be transformed before being useable by a model. This is known as preprocessing and it's usually the first step in the ML process. Subsequent chapters explain how the data for this project needed to be transformed and what steps were taken to prepare it for training.

3.5 Training

Training a Machine Learning model means adjusting its weights to make progressively better predictions on the training data. While each ML algorithm is unique, there is a general procedure used to perform this optimisation process.

Initially, the weights of the model are randomised or set to a default value, such as 0.0, at which point the model is considered untrained. The training, then, takes the form of an iterative process, which consists of a series of computational steps. First, the model takes as input a single training example and outputs a prediction and passes it through an error function that calculates the error of that prediction (e.g. the distance from the true value, which is often known). This process is repeated for every training example in the dataset and the resulting errors are used to compute the *gradient* of the error function, which

[1] https://www.marktechpost.com/2019/10/16/classify-handwritten-digits-with-tensorflow/

is then used to adjust the weights of the model. This method of optimisation is called *gradient descent*, where the calculated gradient points in the direction that minimises the error function, and the weights of the model are adjusted such that the error function moves towards that direction.

The optimisation process ends when the gradient descent algorithm has converged on an error that is sufficiently small.

3.6 Inference

After the training process has completed, the model is considered to be trained and is ready to make predictions on unseen data. The inference process is identical to the first step of the training process: a single input, which has the same structure as a training example, but was not part of the training data, is passed to the model which produces an output. This output represents the prediction of the model for that input and is usually encoded in a format that is mathematically convenient but not understandable by humans without context. Therefore, the output is often put through a series of transformations known as postprocessing which turn the numerical output into a meaningful interpretation of the prediction. In the aforementioned case of hand-written digit classification, the output is a set of 10 floating-point numbers in the range [0.0, 1.0]. During postprocessing, the largest of these probabilities is selected and mapped onto the digit, 0-9, that it corresponds to. The chosen digit is then selected as the final output of the algorithm.

3.7 Model Evaluation and Avoiding Overfitting

When training an ML model, it is important to consider how the performance of the model is going to be assessed after the model has been trained. There exist a variety of statistical techniques used for model evaluation, and all of them rely on the basic premise that the predictive accuracy of a model cannot be properly evaluated on data that the model has already been trained on. This is because the model's weights have been adjusted to fit those exact data, so it's not an unbiased sample. Therefore, the initial dataset is often split into two randomised subsets: a training set on which the model is trained and a test set on which it's evaluated.

This technique works well enough in cases where the model needs to be trained only once before the final evaluation. In most cases, however, there are a lot of decisions that need to be made when selecting a model and a lot of minor adjustments we can make to improve it, even after it has been trained. As a result, intermediate steps are introduced between training and testing, to make small adjustments to our chosen model. When evaluating the effectiveness of those adjustments it is important to not use the same data that the model was trained on, because that can lead to fine-tuning the model to fit those exact data, making it harder to generalise on new, unseen data. This problem is known as *overfitting* and it's a very common issue in machine learning. In order to avoid overfitting, we need to use a dataset that the model has never seen before, but we cannot use the test set because we need it to remain an unbiased sample for the final model evaluation. Therefore, prior to any model training, the initial dataset can be split further into subsets called validation sets. The model can then be trained on the training set, go through a series of modifications which are evaluated using the validation set(s) to avoid overfitting, before finally being evaluated on the test set. In later chapters, we show and elaborate on the implementation of this process for our project's specific use case.

In the following chapter, we will put the theory of Chapters 2 and 3 into practice and discuss our dataset, its preprocessing into jet images, and finally the application of machine learning techniques.

Chapter 4

Machine Learning for Particle Identification

4.1 Data Preparation

Preprocessing: what and why? In practice, a core part of machine learning involves *data preprocessing*. Preprocessing describes a number of operations performed on a dataset in order to optimise the performance of an ML algorithm. Its benefits are improved model performance and a lower model training time.

Preprocessing varies with each dataset and often with the algorithm that we wish to implement. In our case, five main preprocessing steps were used to turn collision data into 2D images which are referred to as *jet images*. But before we discuss those, we will give a description of the dataset that was used in this analysis.

4.2 Dataset

The event samples for our signal and background in the dataset were generated with Sherpa, a Monte Carlo event generator for the simulation of particle collisions. The parameters that were chosen are a centre of mass energy of $\sqrt{s} = 13$ TeV, parton shower effects included, and hadronisation effects omitted. For our background W+jets process, up to three jets are merged, and for both W+jets and our signal $pp \to t\bar{t}H$ (with semi-leptonic decays), there is exactly one far jet in the final state. Our signal is also on-shell, meaning that it satisfies classical equations of motion. For the clustering of the jets, the Cambridge-Aachen jet clustering algorithm was used with a radius parameter $R = 1.5$.

4.2.1 Dataset Features

Each row of the dataset represents an event containing a jet. The columns consist of the pseudorapidity η, the azimuthal angle ϕ, and the transverse momentum p_T of each constituent of the jet. Intuitively, the quantities η and ϕ should be thought of as the spatial coordinates of a constituent, and p_T as the constituent's energy.

Pseudorapidity

The pseudorapidity is a physical quantity used in experimental particle physics which relates to the angle of a particle relative to the beach axis θ in the accelerator, and is given by

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right]. \tag{4.1}$$

Fig. 4.1 shows how η is related to the polar angle θ.

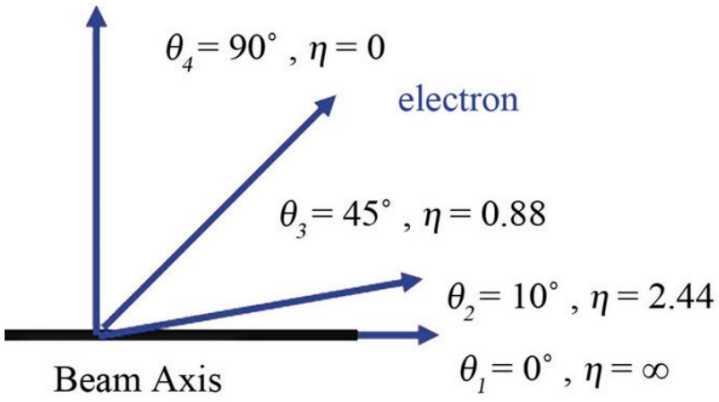

Figure 4.1: Diagram demonstrating η - θ relationship. (Mansour and Bakhet, 2013)

Pseudorapidity is derived from another quantity (namely, rapidity y) in the relativistic limit where the mass of the particle becomes negligible and its only energy is its momentum-energy. It is a useful quantity because differences in pseudorapidity $\Delta\eta$ are Lorentz invariant under boosts along the beam axis (Sahoo, 2016).

Azimuthal Angle

Our second spatial coordinate is the azimuthal angle ϕ, which is simply the angle that wraps around the beam axis with $\phi \in [-\pi, +\pi]$. The azimuthal angle is useful because, similar to η, differences in ϕ are Lorentz invariant under boosts along the longitudinal axis.

Transverse Momentum

Finally, the last important feature in the dataset is the transverse momentum p_T of each constituent in the jet, which represents the momentum of a particle transverse to the hadron beam. Typically, each cell in the calorimeter (detector) measures the energy of an incident constituent. However, a more useful quantity in the transformation of jet images (which we will discuss shortly) is p_T which is given by

$$p_{T,\,\text{cell}} = E_{\text{cell}} \,/\, \cosh\left(\eta_{\text{cell}}\right), \tag{4.2}$$

where E_{cell} is the energy deposited in the cell and η_{cell} is the cell's pseudorapidity (Paganini, 2019). Like η and ϕ, the transverse momentum is constructed so that it is invariant under longitudinal Lorentz boosts, which is why it is preferred over the energy.

In the following section, we will discuss jet images and their preprocessing.

4.3 Jet Images

Jet images are frequently used in particle identification (Santos et al., 2017, Plehn, 2012, Paganini, 2019, Cogan et al., 2015) because, with minimal information loss, the data are turned into a form that can benefit from the rapidly growing field of *computer vision* (Cogan et al., 2015), and popular algorithms for image classification such as Convolutional Neural Networks. Jet images also provide intuition and improve interpretability (Paganini, 2019).

Dimensions. In jet images, the pixels represent the sum of the energies that are deposited in a given region of the detector. In our case, a pixel is a bin of transverse momenta

of constituents in that region. The x-y plane of the image is the η-ϕ plane, with the pseudorapidity and azimuthal angle corresponding to each pixel. An alternative, physics-driven image format uses what is called the Lund plane (Dreyer, Salam, and Grégory Soyez, 2018), but this was not used in this analysis.

Jet Radius. Jet images depict the specific region where a jet is located, and do not include the whole event that took place. This is why an important parameter for these images is the radius R of the jet. In our case, we have a jet radius $R = 1.5$of $R = 1.5$.

Jet images always undergo a series of preprocessing steps. The preprocessing of these images exploits location invariance and thus makes the image much more meaningful without spoiling the physics. In this analysis, the image preprocessing was done in Python with the help of popular libraries, namely, pandas (team, 2020), NumPy (Harris et al., 2020), Matplolib (Hunter, 2007), and Seaborn (Waskom, 2021). The following section will go through the core image preprocessing steps.

4.3.1 Image Preprocessing

To understand the preprocessing more intuitively, figures will be shown of a "before and after" of the images. They display 10000 superimposed images, often referred to as an *average image*, to make the transformations of preprocessing clearer. As a reminder, the code that was written for this report is available on GitHub[1]. The preprocessing steps that were implemented are the following:

Step 1: Extract Maxima

The first preprocessing step involves the implementation of functions that extract the constituents with the three highest transverse momenta for each collision. These three maxima are used in the next three steps which exploit symmetries in the η-ϕ space and minimise useless information in the jet image (Cogan et al., 2015).

Step 2: Shift

In this step the aim is to shift the constituent with the 1st highest momentum to the centre of the image (Fig 4.2). This is done by transforming η and ϕ so that the highest constituent's coordinates are $(\phi', \eta') = (0, 0)$. Hence, for all training data, the brightest pixel (highest momentum) will always be at the centre of the image. In physics terms, this corresponds to rotating and boosting along the beam direction to centre the jet.

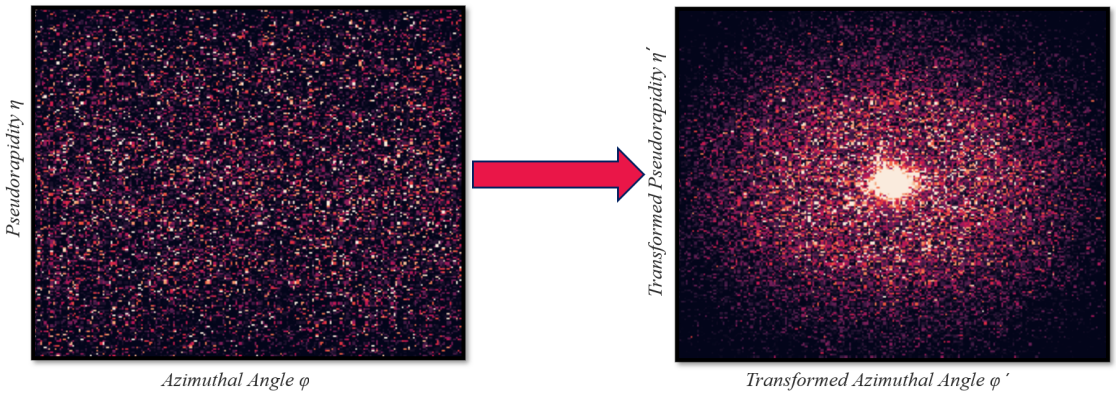

Figure 4.2: Average images demonstrating the centring of the constituents. Notice how brightness (high transverse momentum) accumulates at the centre.

[1] https://github.com/johngeorgousis/jet-classifier

Step 3: Rotate

Next, the image is rotated such that the 2nd highest momentum is in the 6 o'clock position (Fig 4.3). By this, for all images, the second brightest pixel is always in the bottom half and the middle of the image. Physically, this removes the stochastic nature of the decay angle relative to the η-ϕ coordinate system (Cogan et al., 2015).

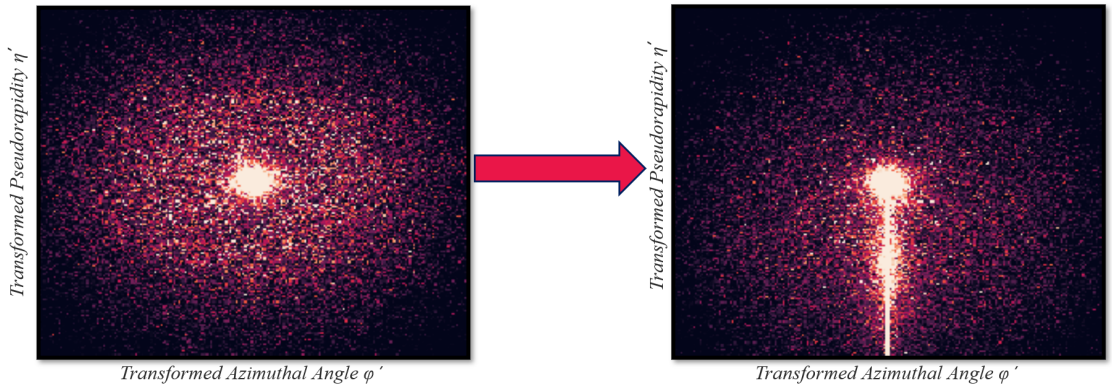

Figure 4.3: Average images demonstrating the rotation around the centre, leading to a vertical high energy line.

Step 4: Flip

Then, the image is reflected over its vertical axis such that the 3rd highest transverse momentum is on the right-half plane (Fig 4.4). By this, the third brightest pixel is always on the right-hand side of the image. This helps the model in classification since it guarantees that powerful radiation will be in similar locations in every image.

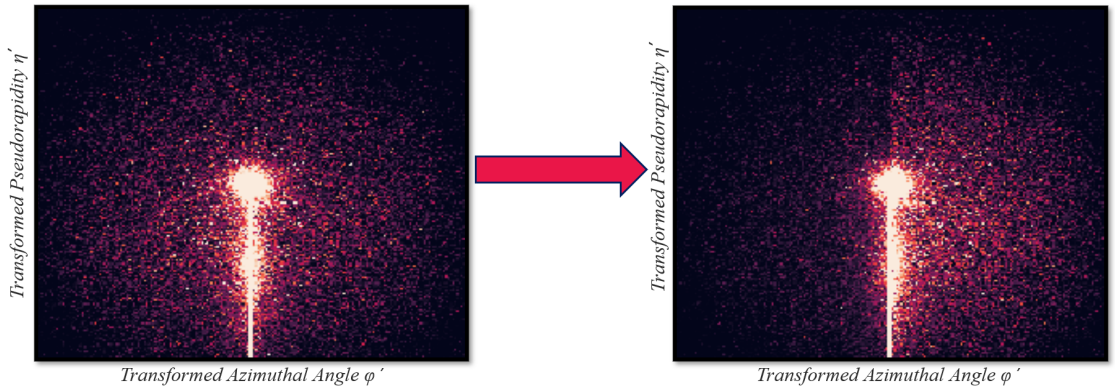

Figure 4.4: Average images demonstrating the flipping of the image, leading to a more energetic right half.

Step 5: Crop

Finally, the image is cropped to 40×40 pixels (Fig 4.5). This number is chosen so that there are enough pixels to decipher differences between images, but not so many that would make the preprocessing and model training times too time-consuming.

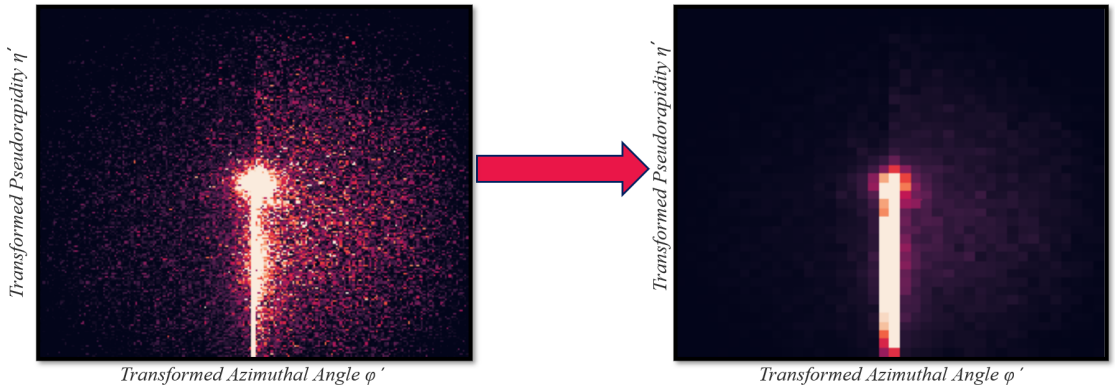

Figure 4.5: Average images before and after cropping to 40×40 pixels.

However, in the figures shown, there are 10000 superimposed images. What image does the model actually see when it has to make a prediction? Fig. 4.6 displays a sample of six single jet images as input to the ML model for testing, along with the predictions that our ConvNet model was able to make.

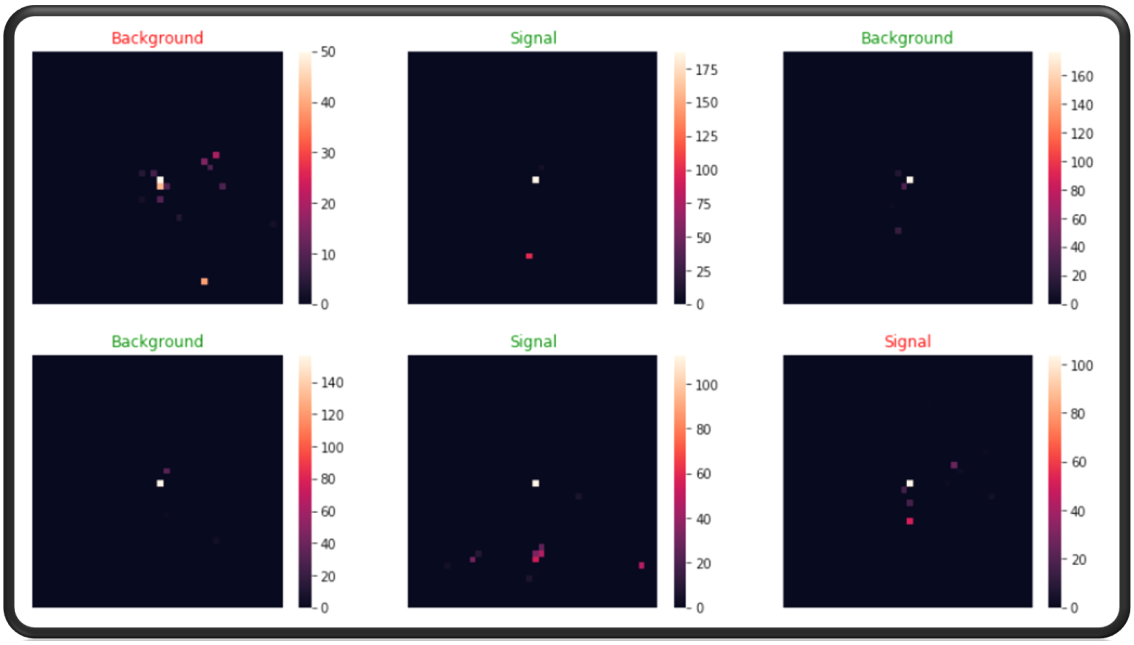

Figure 4.6: This figure displays six randomly chosen images from the test set, along with our model's predictions in the title. The title text is what the model predicted the image to be, while the colour shows whether the model's prediction was right (green) or wrong (red).

4.4 Machine Learning

After the preprocessing is done, the dataset is ready for the machine learning part of the analysis. In the following subsections, we will discuss some important data science tools that were used to optimise our ML model (see GitHub[2] for the code).

4.4.1 Preprocessing

First, the dataset went through a series of further preprocessing steps in order to be prepared for a given algorithm. Part of this preprocessing involved the splitting of the data into subsets for training ($\sim 70\%$ of events), validation ($\sim 15\%$), and testing ($\sim 15\%$) of the models. Two popular ML libraries were used (scikit-learn (Pedregosa et al., 2011) and TensorFlow (Martın Abadi et al., 2015)), the algorithms of which required different preprocessing of the data.

4.4.2 Learning Curve

An important tool in data science is the *learning curve*. The learning curve is a way to visualise how much data are needed in order to achieve the optimum model. If the dataset is too small, then the model will not have enough labelled training examples to be fully optimised, which negatively impact performance. This is known as *underfitting*. If the dataset is too large, then the model can take a substantial amount of time and resources to train, making it unfeasible to optimise or even build one. The learning curve trains and tests the model on datasets of increasing size, allowing us to see where a larger dataset no longer improves the performance of the model significantly. An example learning curve that we used for our Support Vector Machine (SVM) model is shown in Fig. 4.7. In this analysis, tests were run to find an optimal size for the dataset given our computing resources. An optimal size appeared to be 500,000 signal and 500,000 background events,

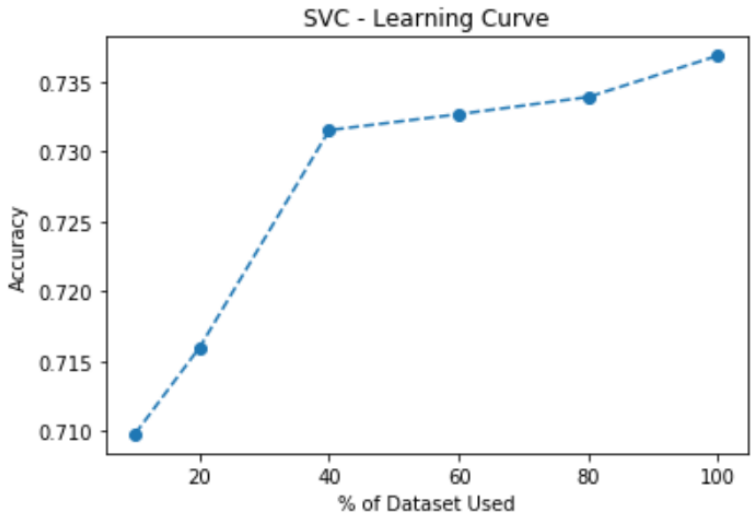

Figure 4.7: Learning curve for an SVM model with default parameters and a dataset of 100k events. As we can see, the model's accuracy on the test set increases significantly with the proportion of the dataset used when it is small, and then we observe a slower but still consistent increase beyond 90k events. This was true for almost all of our best-performing models up to about 1 million events.

which is what was used for our final models.

[2] https://github.com/johngeorgousis/jet-classifier

4.4.3 Model Complexity Graph

The model complexity graph was used for optimising one of the best-performing algorithms, the Convolutional Neural Network (ConvNet)[3]. This graph is useful in determining how long the ConvNet should be trained for. As mentioned previously, if there is insufficient training, this leads to underfitting. However, when the model trains for too long on the same training data it can lead to overfitting, failing to generalise to unseen examples. Moreover, it is useful to know how long the model should train for when time and resources are limited. Fig 4.8 shows the model complexity graph for our final ConvNet model. It shows how its training and validation accuracy change with epochs (i.e. with the times it was trained on the entire dataset). As we can see, our model did not end up overfitting but required no more than five epochs to reach optimal validation accuracy. Thus, it was chosen to stop training the model in the interest of time.

Figure 4.8: Model complexity graph for our final ConvNet model. The y-axis represents the model accuracy and is plotted versus the number of epochs in training.

4.4.4 Hyperparameter Tuning

In machine learning, each algorithm is different, and for each algorithm, there are parameters that change how the algorithm processes data. These parameters are called *hyperparameters*. So the question is: for each algorithm, what combination of hyperparameters will lead to the best-performing model? The process of determining an answer to this question is called *hyperparameter tuning*, and it is widely used in applied machine learning. Hyperparameter tuning is heavily computational as it involves trying out a large number of hyperparameter combinations to figure out which one leads to optimal performance. Since there usually is an incredibly large amount of combinations, this process requires a lot of time as well as computational power. In this analysis, hyperparameter tuning was used to

[3] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

some extent for the best-performing models, but more optimal parameters can be found through more exhaustive tuning, given the time and resources.

4.4.5 Training Times

An important aspect of ML algorithms is the amount of time they take to process input in the training (and often testing) process. If an algorithm takes too long to train, then this may require a sacrifice in performance if there is insufficient computing power (e.g. an inadequate CPU/GPU). This is because the implementer may have to use less data for training and testing to save up time, and perform less comprehensive hyperparameter tuning, risking missing out on the ideal set of parameters for a given algorithm. In the classification of jet images, training times can be particularly high since each training example contains an appreciable amount of information (e.g. 1600 pixels where many can have non-zero values), making the model training times an extra point of interest.

To achieve optimal performance, organisations use supercomputers as well as complex techniques (Zhang, 2021) to ensure that training times are not a major hindrance. In our case, an average home computer was used for the training of the models (CPU: AMD Ryzen 7 3700X), though it is likely that performance would increase if cloud computing was used to borrow computing power from a provider.

In our analysis, a function was implemented that returned values for the training time of an algorithm on increasing proportions of the dataset. By this, we not only see how long an algorithm takes to train on a dataset of a given size, but also how the training time changes with the size of the dataset (e.g. linearly vs exponentially). The results of this analysis will be reported in the next chapter after we discuss the evaluation of the ML algorithms that were tested.

4.4.6 Algorithms

In this project, twelve machine learning algorithms were tested, some of which are popular in particle identification, and others that have not seen much use. Due to lack of time and computational resources, not all models could be tuned and tested on the whole dataset. Thus, three elimination phases of ascending rigour were implemented to converge to the best performing model. In each phase, the worst-performing models were eliminated and disregarded for the rest of the analysis, while the best-performing ones moved onto the next phase. Note: Our ConvNet model made it straight to the final phase because of its exceptional performance relative to the other models.

1st Elimination Phase

Dataset size: 100,000 backgrounds 100,000 signals.
Hyperparameter tuning: No.
Metric: Accuracy
Initial models: 11
Final models: 5

2nd Elimination Phase

Dataset size: 50,000 backgrounds 50,000 signals.
Hyperparameter tuning: Yes.
Metric: Accuracy
Initial models: 5
Final models: 1

3rd Elimination Phrase

Dataset size: 100,000 backgrounds 100,000 signals.
Hyperparameter tuning: No.
Metric: Signal accuracy & Background accuracy
Initial models: 2 (1 + ConvNet model)
Final models: 1

4.4.7 Evaluation Metrics

Evaluation metrics are different ways of measuring the performance of a machine learning model. There's a wide variety of metrics in data science, but we will focus on the ones that provide the most intuition in our use case.

Accuracy

Classification accuracy is one of the simplest performance metrics in data science. It is simply the ratio of the number of correct predictions to the total number of predictions, i.e. the proportion of predictions that the model got right. As an evaluation metric, accuracy can be biased if the data classes are imbalanced (e.g. 20k signal events and 80k background events), so, to avoid this, we chose an equal amount of signals and backgrounds in our dataset.

Signal and Background Accuracy

Two more important values to consider are the specific accuracies in classifying signals and backgrounds. Signal accuracy P(signal|signal) is the proportion of signal images that were classified as signal, while background accuracy P(background|background) describes the proportion of background images that were classified correctly.

Confusion Matrix

A useful tool that includes the aforementioned metrics is called a *confusion matrix*, an example of which is shown in Fig. 4.9. The confusion matrix is an $n \times n$ matrix where n is the number of classes (in our case, n = 2) that is constructed so that the number of accurate predictions of the model fall into the main diagonal, and all false predictions are counted outside that diagonal. The blocks of the matrix are coloured based on the number of examples that are in each, providing a way to visualise how successful our model is. If the main diagonal is way "brighter" than the rest of the matrix, then the model makes accurate classifications for all classes. The confusion matrix takes into account false positives/negatives as well as true positives/negatives of the model, making it a powerful tool in our analysis.

4.4.8 Algorithms

As mentioned earlier, a multitude of ML algorithms were tested for our specific signal-background classification task, all of which are listed in Table 4.1 along with their common names. While the most frequently used algorithms in particle physics are BDTs and various types of Neural Networks (NN), it was decided to use a more comprehensive subset for the sake of completeness and because results in this specific classification task are not abundant.

In Chapter 5 we will discuss the results of our training times and model performance comparisons before discussing the model that yielded the best classification results.

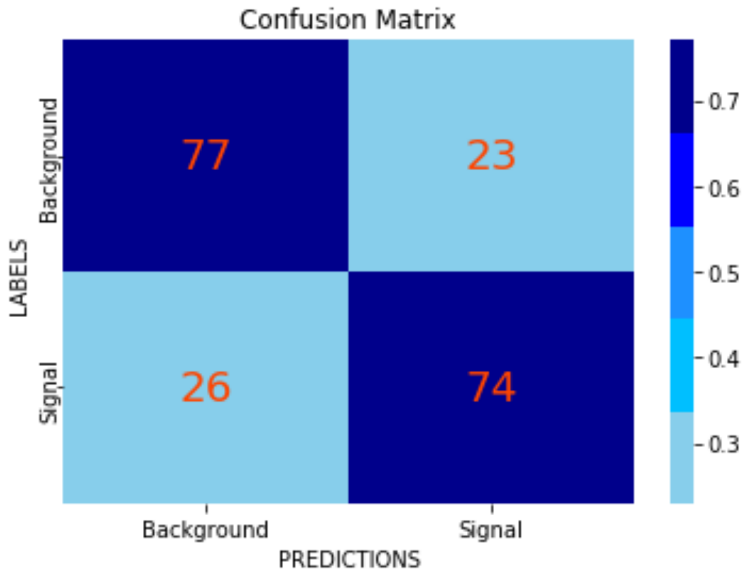

Figure 4.9: Confusion matrix for one of our ConvNet models. The top left block is the value for P(background|background) and the bottom right block is the value for P(signal|signal) while the proportions of false positives and false negatives are shown in the top right and bottom left corners respectively.

Algorithm	Common Name
MultinomialNB	Naive Bayes
GaussianNB	Naive Bayes (Variation)
RandomForestClassifier	Random Forests
KNeighborsClassifier	K-Nearest Neighbours (KNN)
SGDClassifier	Stochastic Gradient Descent (SGD)
LinearDiscriminantAnalysis	LDA
AdaBoostClassifier	AdaBoost
ConvNet	Convolutional Neural Networks
DecisionTreeClassifier	Decision Trees
BaggingClassifier	Bagging
GradientBoostingClassifier	Boosted Decision Trees (BDTs)
SVC	Support Vector Machine (SVM)

Table 4.1: List of ML algorithms that were tested in this analysis.

Chapter 5

Results and Model Comparison

5.1 Model Training Times

As discussed in the previous section, when choosing an ML algorithm, an important aspect to consider is the amount of time it takes to process input and produce an output (summarised as a model's *training time*). If an algorithm takes too long to train, then sacrifices might need to be made in the amount of training data that will be used, and our learning curves showed that, for most algorithms, there was an increase in performance even after one million training examples. Thus, it is recommended that more training examples by other users in this field, given the time and resources.

Training Times

When comparing the training times for each ML algorithm in this analysis, there were two main points of interest:

1. How long an algorithm takes to train.
2. How the training times change with respect to the size of the dataset (i.e. the number of generated events used for training and testing).

To extract these, algorithms were set to their default hyperparameters and a dataset of 35,000 signals and 35,000 backgrounds was used. Note well that training times can vary significantly for different values of hyperparameters, so these results are not fully rigorous for this use case. The results for each classifier are shown in table 5.1.

The ConvNet model could not be directly compared to the rest, so a time estimate was made using a standard Conv2D model in TensorFlow[1] with the epochs required to reach maximum accuracy on validation data. In addition, the ConvNet model was trained using a CPU instead of a GPU, so its training time here is much larger than it would be in practice.

As we can see, there can be significant differences between model training times, upwards of 5 orders of magnitudes for 70,000 events. This gap can widen rapidly as the number of training examples increase, seeing as some models approximate a constant rate of change, while others display an exponential one. What this means in practice is that a Support Vector Machine model, for example, would require either costly resources or to potentially compromise classification performance for the sake of a lower training time. In contrast, a Naive Bayes or Boosting model could afford more comprehensive testing and a significantly larger dataset to train on. Moving on from training times, we will next compare the performance of these algorithms when it comes to classifying unseen events as either background or signal.

[1] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

Model	Training Times (minutes)	Rate of Change
MultinomialNB	10^{-3}	Linear
GaussianNB	10^{-2}	Linear
RandomForestClassifier	10^{-1}	Exponential
KNeighborsClassifier	10^{-1}	Exponential
SGDClassifier	10^{-1}	Exponential
LinearDiscriminantAnalysis	10^{-1}	Linear
AdaBoostClassifier	10^{0}	Linear
ConvNet*	10^{0}	Exponential
DecisionTreeClassifier	10^{0}	Exponential
BaggingClassifier	10^{0}	Exponential
GradientBoostingClassifier	10^{1}	Linear
SVC	10^{2}	Exponential

Table 5.1: Training time comparison for each algorithm. The second column displays the training times to the nearest order of magnitude, whereas the third column shows the rate of change of the training time with respect to the number of training examples.

5.2 Model Performance Comparison

5.2.1 1st Elimination

In the first elimination phase, we started with 11 algorithms and eliminated 6, based on their classification accuracy on a balanced sample of 50k signals and 50k backgrounds.

Table 5.2 shows the model performances in descending order, where the last 6 algorithms were not used further in this analysis. As we can see, the SGD and Gaussian Naive Bayes models defaulted to an accuracy of 50%, which is just as well as a model would do if it classified images randomly. This often happens when models predict all images as either only backgrounds or only signals, which is what happened with the SGD model. The Decision Trees, LDA's, Adaboost, and BDT models achieved a higher accuracy, but still not high enough to move onto the next elimination phase. As an important reminder, while we will disregard these algorithms due to their performance on this specific test, it is recommended that more effort be put into testing these models given sufficient time, as we reached this conclusion using their default parameters which are almost certainly suboptimal.

Model	Accuracy (%)
SVC	74.2
RandomForestClassifier	73.7
MultinomialNB	71.3
KNeighborsClassifier	70.7
BaggingClassifier	70.6
GradientBoostingClassifier	70.1
AdaBoostClassifier	69.9
LinearDiscriminantAnalysis	69.5
DecisionTreeClassifier	66.2
SGDClassifier	50.3
GaussianNB	49.7

Table 5.2: Table showing the phase 1 classification accuracy on unseen test data. Note: Individual signal and background accuracies were also investigated and were almost identical in all but the last two models.

5.2.2 2nd Elimination

In the second elimination phase, the top 5 remaining contenders were trained and tested on a smaller dataset of 50k balanced events, but with the addition of hyperparameter tuning. We observe that while the addition of hyperparameter tuning improved the performance of some algorithms, this was not to a very significant margin, with all accuracy measures remaining in the 70s range. Three out of five algorithms were eliminated due to their inferior performance in this test and were not investigated further in this analysis. In the final elimination phase, we will see how Random Forests and Support Vector Machines compare in classification accuracy versus our most popular, Convolutional Neural Network model.

Model	Accuracy (%)
SVC	74.2
RandomForestClassifier	73.6
KNeighborsClassifier	71.7
BaggingClassifier	71.6
MultinomialNB	71.3

Table 5.3: Table displaying the phase 2 classification accuracy on unseen test data. Note: Individual signal and background accuracies were also investigated and were almost identical in all models.

5.2.3 3rd Elimination

In the third and final elimination phase, our ConvNet model along with the best-performing model of phase two were trained, tested, and compared on 500k events to choose the best model for this classification task. The results of this comparison are shown in Table 5.4. As we can see, the ConvNet model outperformed the SVM in both signal and background classification accuracy, while requiring less than 18 times the training time that SVM required.

Model	Accuracy (%)	Background Acc. (%)	Signal Acc. (%)	Training Time (Mins)
ConvNet	76.8	71.9	81.7	37
SVC	74.4	72.8	76.2	683

Table 5.4: Table showing the phase 3 classification accuracy on unseen test data. The individual Signal and Background accuracies are also shown along with the model training time.

5.3 Best-Performing Model

As demonstrated in our performance tests, the Convolutional Neural Networks model is by far superior to the other eleven models in this image classification task. In addition, the ConvNets model required less training time than most of our best-performing models to reach a higher accuracy. Due to this, it was chosen as our best-performing model, and consequently trained on another 1,000,000 events to improve its performance even further. The final classification accuracy that we could achieve with our best model is shown in Table 5.5 along with its specifications in Figure 5.1.

Model	Accuracy (%)	Background Acc. (%)	Signal Acc. (%)
ConvNet	76.8	74.7	78.6

Table 5.5: Classification accuracy on unseen test data.

Discussion

Comparing our best model for this classification task to other efforts to distinguish our signature process from a background (Santos et al., 2017) or other particle identification

tasks altogether (Cogan et al., 2015, Kasieczka et al., 2017, Paganini, 2019), our classifier still managed to perform reasonably well, given our time and resources. In practice, since our background is not overwhelmingly large the model could make meaningful signal classifications against W+jets. The final chapter will provide an outlook on our study and some points of self-reflection.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 38, 38, 16)        160
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 19, 19, 16)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 17, 17, 32)        4640
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 64)          18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               73856
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 258
=================================================================
Total params: 97,410
Trainable params: 97,410
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.1: ConvNet model's specifications after hyperparameter tuning. Hyperparameters: kernel_size=3, padding='valid', activation='relu', optimizer='adam', loss=SparseCategoricalCrossentropy()

Chapter 6

Conclusion

From the results in Chapter 4, machine learning seems like a potentially promising tool for $t\bar{t}H$(bb) classification and shows that it can perform well even in complex, multi-jet environments. However, there is room for improvement, and many more important backgrounds should be considered for our signature, such as the dominant $t\bar{t}$+b-jets background. Given the importance in particle physics of correctly identifying this signal, we hope that this study will pave the way for new semi-leptonic $t\bar{t}H$ analyses.

In terms of future improvements to this study, there should be a larger variety of model performance metrics (e.g. using production rates for the signal and background), more comprehensive and rigorous model building and training, and more plots providing intuition on the relationship between the performance of the models and the physics.

Acknowledgements

I am indebted to my supervisor, Dr Jonas Lindert, for providing the idea for this project, and for organising numerous meetings spent discussing the physics involved and debugging assorted codes.

Bibliography

Aad, G. et al. (May 2014). "Search for direct production of charginos, neutralinos and sleptons in final states with two leptons and missing transverse momentum in pp collisions at $\sqrt{s} = 8$TeV with the ATLAS detector". In: *Journal of High Energy Physics* 2014.5. ISSN: 1029-8479. DOI: 10.1007/jhep05(2014)071. URL: http://dx.doi.org/10.1007/JHEP05(2014)071.

Bourilkov, Dimitri (Dec. 2019). "Machine and deep learning applications in particle physics". In: *International Journal of Modern Physics A* 34.35, p. 1930019. ISSN: 1793-656X. DOI: 10.1142/s0217751x19300199. URL: http://dx.doi.org/10.1142/S0217751X19300199.

CERN (2019). *CMS measures Higgs boson's mass with unprecedented precision*. URL: https://home.cern/news/news/physics/cms-measures-higgs-bosons-mass-unprecedented-precision (visited on 03/15/2021).

— (2021a). *Pulling together: Superconducting electromagnets*. URL: https://home.cern/science/engineering/pulling-together-superconducting-electromagnets (visited on 03/15/2021).

— (2021b). *Storage*. URL: https://home.cern/science/computing/storage (visited on 03/15/2021).

— (2021c). *The Higgs boson*. URL: https://home.cern/science/physics/higgs-boson (visited on 03/15/2021).

— (2021d). *The Large Hadron Collider*. URL: https://home.cern/science/accelerators/large-hadron-collider (visited on 03/15/2021).

Cogan, Josh et al. (Feb. 2015). "Jet-images: computer vision inspired techniques for jet tagging". In: *Journal of High Energy Physics* 2015.2. ISSN: 1029-8479. DOI: 10.1007/jhep02(2015)118. URL: http://dx.doi.org/10.1007/JHEP02(2015)118.

Collaboration, ATLAS (2019). *New milestone reached in the study of electroweak symmetry breaking*. URL: https://atlas.cern/updates/briefing/milestone-electroweak-symmetry-breaking (visited on 03/15/2021).

Couchman, Jonathan (2002). *W Boson Decays*. URL: http://www.hep.ucl.ac.uk/~jpc/all/ulthesis/node45.html (visited on 03/15/2021).

Dawson, S. (2001). "Top quark Yukawa couplings and new physics". In: *AIP Conference Proceedings*. ISSN: 0094-243X. DOI: 10.1063/1.1394318. URL: http://dx.doi.org/10.1063/1.1394318.

Dreyer, Frédéric A., Gavin P. Salam, and Grégory Soyez (Dec. 2018). "The Lund jet plane". In: *Journal of High Energy Physics* 2018.12. ISSN: 1029-8479. DOI: 10.1007/jhep12(2018)064. URL: http://dx.doi.org/10.1007/JHEP12(2018)064.

Englert, Christoph et al. (2011). *W+jets, Z+jets, multijets and new physics searches*. arXiv: 1110.1043 [hep-ph].

Fermilab (2021). *Frequently Asked Questions About the Higgs Boson*. URL: https://news.fnal.gov/wp-content/uploads/Higgs_Boson_FAQ_July2012.pdf (visited on 03/15/2021).

Guindon, Stefan (2017). *Search for ttH at ATLAS*. URL: https://www2.physics.ox.ac.uk/sites/default/files/2012-03-27/tth_guindon_oxford_pdf_20865.pdf (visited on 03/15/2021).

Harris, Charles R. et al. (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Ilisie, Victor (2011). *S.M. Higgs Decay and Production Channels*. URL: http://ific.uv.es/lhcpheno/PhDthesis/master_vilisie.pdf.

Kasieczka, Gregor et al. (May 2017). "Deep-learning top taggers or the end of QCD?" In: *Journal of High Energy Physics* 2017.5. ISSN: 1029-8479. DOI: 10.1007/jhep05(2017)006. URL: http://dx.doi.org/10.1007/JHEP05(2017)006.

Mansour, H. M. M. and Nady Bakhet (2013). "Search for High Energy Electrons from New Neutral Massive Gauge Boson Decay in the CMS Detector at the LHC Using Monte Carlo Simulation". In: *Open Journal of Microphysics* 03.02, pp. 34–42. DOI: 10.4236/ojm.2013.32007. URL: https://doi.org/10.4236/ojm.2013.32007.

Marr, Bernard (2018). *27 Incredible Examples Of AI And Machine Learning In Practice*. URL: https://www.forbes.com/sites/bernardmarr/2018/04/30/27-incredible-examples-of-ai-and-machine-learning-in-practice (visited on 03/15/2021).

Martın Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Marzani, Simone, Gregory Soyez, and Michael Spannowsky (2019). *Looking Inside Jets*. Springer International Publishing. DOI: 10.1007/978-3-030-15709-8. URL: https://doi.org/10.1007/978-3-030-15709-8.

Mitchell, Tom (1997). *Machine Learning*. New York: McGraw-Hill. ISBN: 0070428077.

Paganini, Michela (2019). *Machine Learning Solutions for High Energy Physics: Applications to Electromagnetic Shower Generation, Flavor Tagging, and the Search for di-Higgs Production*. arXiv: 1903.05082 [hep-ex].

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Plehn, Tilman (2012). "Lectures on LHC Physics". In: *Lecture Notes in Physics*. ISSN: 1616-6361. DOI: 10.1007/978-3-642-24040-9. URL: http://dx.doi.org/10.1007/978-3-642-24040-9.

Russell, Stuart (1995). *Artificial intelligence : a modern approach*. Englewood Cliffs, N.J: Prentice Hall. ISBN: 0-13-103805-2.

Sahoo, Raghunath (2016). *Relativistic Kinematics*. arXiv: 1604.02651 [nucl-ex].

Santos, R. et al. (Apr. 2017). "Machine learning techniques in searches fortthin theh→bb decay channel". In: *Journal of Instrumentation* 12.04, P04014–P04014. ISSN: 1748-0221. DOI: 10.1088/1748-0221/12/04/p04014. URL: http://dx.doi.org/10.1088/1748-0221/12/04/P04014.

Seiberg, Nathan (2016). *The World's Largest Experiment*. URL: https://www.ias.edu/ideas/2007/seiberg-lhc (visited on 03/15/2021).

Sirunyan, A.M. et al. (June 2018). "Observation of tt¯H Production". In: *Physical Review Letters* 120.23. ISSN: 1079-7114. DOI: 10.1103/physrevlett.120.231801. URL: http://dx.doi.org/10.1103/PhysRevLett.120.231801.

team, The pandas development (Feb. 2020). *pandas-dev/pandas: Pandas*. Version 1.1.4. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

Verkerke, W. (2017). *Measurement of W+jets and top quark pair production cross-sections in ATLAS*. URL: https://indico.cern.ch/event/121499/attachments/67567/96963/cernseminar.pdf (visited on 03/15/2021).

Waskom, Michael L. (2021). "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60, p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

Wertz, Sébastien (Oct. 2016). "The Matrix Element Method at the LHC: status and prospects for Run II". In: *Journal of Physics: Conference Series* 762, p. 012053. DOI: 10.1088/1742-6596/762/1/012053. URL: https://doi.org/10.1088/1742-6596/762/1/012053.

Zhang, Zhao (2021). *Enabling Scalable and Efficient Deep Learning on Supercomputers*. URL: https://indico.cern.ch/event/779557/contributions/3244287/attachments/1768183/2871837/DeepLearningSystems.pdf (visited on 03/15/2021).