EXPLORATION OF BOTTOM QUARK ORIGINATED JET TAGGING WITH MACHINE LEARNING TECHNIQUES

AYŞE ASU GÜVENLİ

ÖZYEĞİN UNIVERSITY

JANUARY, 2025

EXPLORATION OF BOTTOM QUARK ORIGINATED JET TAGGING WITH MACHINE LEARNING TECHNIQUES

by
Ayşe Asu Güvenli

Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in
Physics

Advisor: Prof. Dr. Taylan Akdoğan
Co-advisor: Prof. Dr. Bora Işıldak

Graduate School of Science and Engineering
Özyeğin University
İstanbul

January, 2025

EXPLORATION OF BOTTOM QUARK ORIGINATED JET TAGGING WITH MACHINE LEARNING TECHNIQUES

Approved by:

Prof. Dr. Taylan Akdoğan, Advisor
Department of Natural Sciences
Özyeğin University

Asst. Prof. Parviz Elahi
Department of Natural Sciences
Özyeğin University

Prof. Dr. Bora Işıldak, Co-advisor
Department of Physics
Yıldız Technical University

Doç. Dr. Kaan Güven
Department of Natural Sciences
Özyeğin University

Prof. Dr. Burçin Ünlü
Department of Natural Sciences
Özyeğin University

Approval Date: December 27, 2024

DECLARATION OF ORIGINALITY

I hereby declare that I am the sole author of this thesis and that this is the true copy of my thesis, including the final revisions, approved by my thesis committee. All data and information have been obtained, produced, and presented in accordance with the rules of research ethics and principles of academic honesty. As required by these rules, to the best of my knowledge I have acknowledged ideas, thoughts, and any copyrighted material in accordance with the standard referencing rules. I certify that any part of this thesis has not been submitted for a degree or diploma in another educational institution.

Ayşe Asu Güvenli

ABSTRACT

In this study, Machine Learning methods are applied to the problem of classifying b quark-originated jets in particle physics. For this purpose, a custom dataset was specifically prepared, and Multi-Layer Perceptron (MLP), XGBoost, and Retentive Networks models were tested. Among these, the Retentive Network-based architecture, JetRetNet, demonstrated superior performance. Performance evaluations were conducted by comparing JetRetNet with state-of-the-art models such as DeepJet and Particle Transformer. While JetRetNet does not outperform these models, it shows significant potential for future applications in b-jet tagging with further data and optimization.

Keywords: Particle Physics, Machine Learning, Heavy Flavor Tagging

ÖZET

Bu çalışmada, parçacık fiziğinde b kuark kaynaklı jetlerin sınıflandırılması problemine Makine Öğrenimi yöntemleri uygulanmıştır. Bu amaçla, özel olarak hazırlanmış bir veri seti üzerinde Çok Katmanlı Algılayıcı (MLP), XGBoost ve Retentive Networks modelleri test edilmiştir. Retentive Network tabanlı JetRetNet mimarisi, üç model arasında üstün performans göstermiştir. Performans değerlendirmeleri, DeepJet ve Particle Transformer gibi son teknoloji modellerle karşılaştırılarak gerçekleştirilmiştir. JetRetNet, bu modelleri aşamamakla birlikte, daha fazla veri ve optimizasyon ile gelecekte b jet etiketleme problemlerinde önemli bir potansiyele sahip olduğunu göstermektedir.

Anahtar Kelimeler: Makine Öğrenimi, Parçacık Fiziği, Ağır Tat Etiketleme

ACKNOWLEDGEMENTS

We sincerely thank the CERN CMS Open Data Portal for providing the simulated dataset used in this work. This resource's accessibility and quality have been invaluable for conducting this research.

I must thank Prof. Dr. Bora Işıldak for his trust in me. Without his presence, I would have been lost in my academic pursuits. Although I have a lot of passion and dedication, it means nothing if I don't know where to put that into. Thanks to him, I now have a direction to look at and great confidence that I can achieve whatever I want if I put enough work into it. As I declare him my life-long advisor, my questions will continue after this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ATLAS	A Toroidal LHC Apparatus
AUC	Area Under Curve
BSM	Beyond Standard Model
BR	Branching Ratio
CERN	The European Organization for Nuclear Research
CMS	Compact Muon Solenoid
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DL	Deep Learning
DNN	Deep Neural Networks
ECAL	Electromagnetic Calorimeter
FFN	Feed Forward Network
FPR	False Positive Rate
FSR	Final State Radiation
GeLU	Gaussian Error Linear Unit
GPU	Graphics processing unit
HCAL	Hadronic Calorimeter
HEP	High-Energy Physics
IP	Interaction Point
IRC	Infrared and Collinear

ISR	Initial State Radiation
LHC	Large Hadron Collider
LSTM	Long Short Term Memory
MC	Monte Carlo
MHA	Multi-Head Attention
ML	Machine Learning
MLP	Multi Layer Perceptron
MPI	Multi-Parton Interaction
MSR	Multi-Scale Retention
MSE	Mean Squared Error
ParT	Particle Transformer
PID	Particle Identification
QCD	Quantum Chromodynamics
ReLU	Rectified Linear Unit
RetNet	Retentive Networks
RGB	Red Green Blue
RNN	Recurrent Neural Networks
ROC	Receiver-operating characteristic
SM	Standard Model
SV	Secondary Vertex
TPR	True Positive Rate
WP	Working Point

XGB	Extreme Gradient Boosting

1. INTRODUCTION

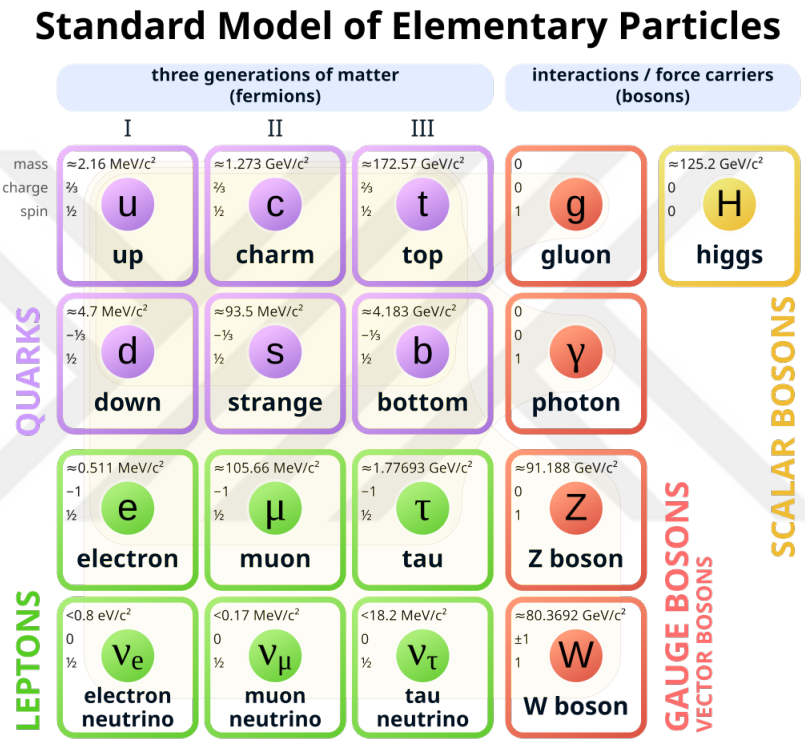
In this chapter, the necessary information that leads to jet formation and detection, especially for the b quark-originated jets, will be given. In Section 1.1, The Standard Model of particle physics will be introduced, and then in Section 1.2, the particle accelerators and their connection to High-Energy Physics (HEP) will be explained. This is followed by the working principles of particle detectors in Section 1.3. In Section 1.4, the physics that leads to jet formation and the definition of a jet will be explained. Finally, in Section 1.5, the significance of the detection of b quark-originated jets will be given, which is the main topic of this thesis.

1.1 Standard Model of Particle Physics

The Standard Model (SM) represents our current understanding of the behavior of particles and their interactions at the smallest scale. Throughout history, scientists have believed that they have discovered the most fundamental and unbreakable particles. The Greek philosopher Democritus first proposed this idea, known as Atomism, which comes from the Greek word Atomos, meaning "uncuttable." However, we later discovered that atoms are composed of protons, neutrons, and electrons, which were assumed to be the fundamental particles. Each time we made such an assumption, we eventually realized that even more fundamental particles make up the ones we previously thought were fundamental. Protons and neutrons are no exception to this observation. With the advancement of theory and experimental capabilities, we have found that quarks and their interactions form a proton or a neutron. Currently, physicists consider electrons (leptons) and quarks to be the fundamental particles. However, if these are not the true fundamental particles, we are currently unable to determine this since the Large Hadron Collider (LHC) or any other collider does not possess the necessary energy to split these particles.

Various sub-categories of the SM can be explored. While we will cover most of them, let's begin with the most basic sub-division, which consists of two categories: Fermions and Bosons.

Figure 1.1: *Standard Model of Elementary Particles and Interactions. The model can be divided into bosons and fermions, and the latter can be further divided into quarks and leptons.[1]*



The fermions are relatively easier to understand because they align with our historical and intuitive understanding of matter. The fermions consist of quarks, which form protons and neutrons, as well as electrons (leptons) and their neutrinos. We will explain neutrinos later. The brick-and-mortar analogy is a commonly used analogy to differentiate between fermions and bosons. In this analogy, fermions are represented as bricks that make up matter, while bosons are depicted as the mortar that holds these matter particles together. It is important to note that this analogy is not comprehensive but rather provides

a general understanding of the subject. As the mortar analogy suggests, bosons are referred to as force-carrying particles. Therefore, contrary to our previous knowledge of physics, particle interactions in the SM are represented by the exchange of force-carrying particles.

In the universe, we know there are (at least) four types of forces, each associated with at least one boson, except for one force. The forces and their corresponding bosons are as follows: the electromagnetic force is related to the photon (γ), the weak force is associated with the $W^{\pm}$ and Z bosons, and the strong force is associated with eight types of gluons.

As one may have noticed, the first force discovered, the gravitational force, is missing. Theoretical predictions suggest the existence of a particle called the graviton, which is associated with the gravitational force. However, experimental validation is lacking. This may be because the gravitational force is much weaker than the other forces, or it could indicate that it has a different nature. According to Einstein's General Relativity, the gravitational force arises from the curvature of space and time.

It is important to note that the SM is a Quantum Field Theory that incorporates special relativity [2]. A theory that combines Quantum Field Theory and General Relativity is currently a significant goal in physics.

The Higgs boson, which was experimentally validated in 2012, is also involved. The Higgs mechanism imparts mass to particles. Although it may appear to be related to gravity, this mechanism explains why and how each particle has a specific mass but not the gravitational interaction between the massive particles.

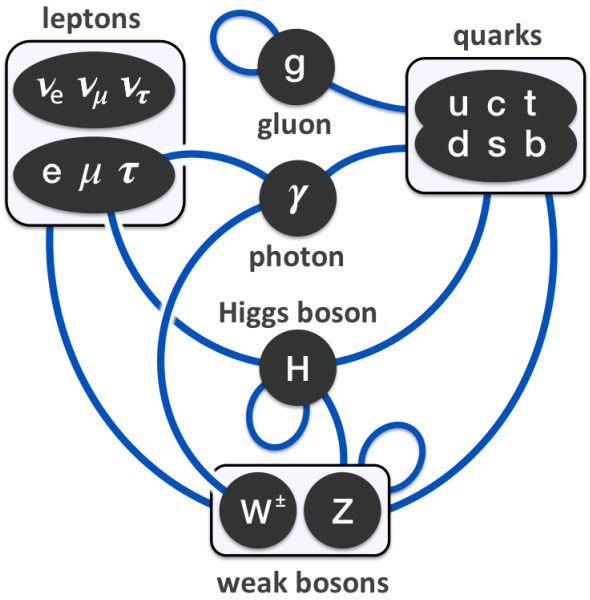
This encompasses our first subdivision, fermions and bosons. There are no further meaningful subcategories of bosons, but there are for fermions. Fermions are divided into quarks and leptons. There are six types of quarks: up, down, charm, strange, top, and bottom quarks, with the latter being the focus of this thesis. Similarly, there are also six

types of leptons: electron, muon, tau, and their corresponding neutrinos.

Fermions can also be classified into generations. These generations correspond to the three columns of the fermion side in Figure 1.1. As we move from left to right in Fig. 1, the generation and mass increase while the lifetime of the particle decreases. We are most familiar with the first generation, which is stable, as the higher generations tend to decay into the lower generations.

Having briefly discussed each type of particle, it is now time to discuss the interactions between matter particles and the force carriers. Because of the nature of each force and particle, we will see that some particles are ”blind” to others. What is meant by ”blind” is that there is no direct interaction between certain particles, but they can still interact via a mediator particle. It is best to examine this visually in Figure 1.2. Let's explore each force and understand the reasons behind these connections.

Figure 1.2: *The interaction between the matter particles and bosons. Where each type of particle interacts with one another through specific rules.*



Let's begin with the most familiar force, the electromagnetic force. It is known

that the electromagnetic force only acts on charged particles. Therefore, a photon can interact with the quarks, all fermions except neutrinos, and the $W^{\pm}$ bosons. Since a photon is a neutral particle, it does not interact with itself directly. Additionally, a photon does not interact with the Higgs Field due to its lack of mass.

Moving on to the next force, the strong force. To comprehend why certain particles are affected by the strong force, it is necessary to provide some information about their nature.

Similar to how some particles possess an electric charge and are influenced by electromagnetism, certain particles have a color charge that is subject to the strong force, explained by Quantum Chromodynamics (QCD). These color charges are not related to actual colors, but their algebraic properties align with the Red Green Blue (RGB) (Red-Blue-Green) group algebra. Just like electromagnetism has a positive charge and its corresponding negative charge, color charges also have anti-colors. There are six color charges: red, blue, green, anti-red, anti-blue, and anti-green.

In nature, only quarks and gluons possess color charge, thus making them the only particles affected by the strong force. This explains why the strong force only manifests within the size range of a nucleus.

An important phenomenon that governs strong force is known as color confinement. This phenomenon states that no color charge can be observed in a bound state, which means that quarks and gluons must form color-neutral groups to be in a bounded state. The formation of protons, or more generally hadrons, directly results from this phenomenon [3]. We will discuss the strong force further when we mention jet formation in Section 1.4.

Next, we have the Weak Interaction, which is the mechanism responsible for nuclear decay. Similar to the strong force, it acts at a distance smaller than the diameter of a proton. It can convert a proton into a neutron or vice versa.

An essential characteristic of the weak force is that it is the only interaction that does not conserve the flavor of the particles involved. Recall that each quark has a flavor, and the six quarks in Figure 1.1 have different flavors. This means that if we have a top quark, it can only change generation through weak interaction. Another consequence is that if we produce a b quark, its jet in the final state should also contain a b quark if the process does not involve weak interaction. Depending on the charges of the initial and final particles, the weak interaction is mediated by $W^{\pm}$ or Z bosons, also known as charged and neutral currents.

Lastly, the Higgs boson only interacts with massive particles. In other words, particles that interact with the Higgs field acquire mass. Higgs mechanism explains how particles acquire their masses.

One might notice that no fermion can directly interact with another fermion. They require an interaction particle, or boson, to be able to interact. That is why the explanation in Figure 1.2 was presented through the lens of the forces/bosons.

Even though the SM has extreme precision and explains most of the things we observe in nature, it has its limitations. From cosmological observations and theories (ΛCDM), we know that ordinary matter (SM particles) make up only 5% of the mass-energy content of the universe. The rest is composed of 26.8% Dark Matter [4] and 68.2% Dark Energy. The currently SM does not have any explanation for these observations. Hence, the physicists constantly search for particles Beyond Standard Model (BSM), or look for places where the experiment does not match SM predictions.

1.2 Particle Accelerators and High-Energy Physics

In the most fundamental sense, HEP examines the collisions of particles that are very fast and small and, therefore, possess high energy. There are two primary reasons for the necessity of high energy.

The first reason is to generate particles with a significant mass greater than that of an electron. This can be understood by considering Einstein's energy-momentum relation given in Equation 1.1.

$$E^2 = p_{\mathrm{i}}^2 c^2 + m_{\mathrm{i}}^2 c^4 = m_{\mathrm{f}}^2 c^4 \tag{1.1}$$

The subscripts i and f refer to the initial and final states. By assuming the final state momentum is zero to obtain the minimum energy, it becomes clear that in order to produce massive final particles, the initial particles must possess high momentum and, consequently, high energy. It is important to note that this calculation relies on the conservation of energy, a crucial aspect of most particle physics calculations.

The second reason for the high energy requirement is its ability to investigate small scales. This concept is not immediately obvious and can be explained as follows: In collider experiments, we primarily collide protons or electrons, both of which are subject to quantum mechanics and possess a de Broglie Wavelength, given in Equation 1.2.

$$\lambda = \frac{h}{p} \tag{1.2}$$

Where h represents Planck's constant, and p denotes the particle's momentum. As the momentum and energy increase, the wavelength decreases. To illustrate this, consider the analogy of two distant stars that appear close to each other. If we send a wave, as depicted in the upper image of Figure 1.3, we are unable to distinguish the two stars and perceive them as a single entity. However, if we send a wave, as shown in the lower image of Figure 1.3, we can discern the two stars as separate entities.

This same logic applies to collider experiments: with smaller wavelengths, we are able to investigate smaller phenomena. Figure 1.4 depicts energy scales and their corresponding research areas. Understood from here, as we reach the Tera electron Volt (TeV) scales, we are able to probe the quarks and the bosons.

Figure 1.3: *Observing two distant stars with different wavelengths. Long wavelengths on the upper part and short wavelengths on the lower part show the relationship between separability and wavelength.*

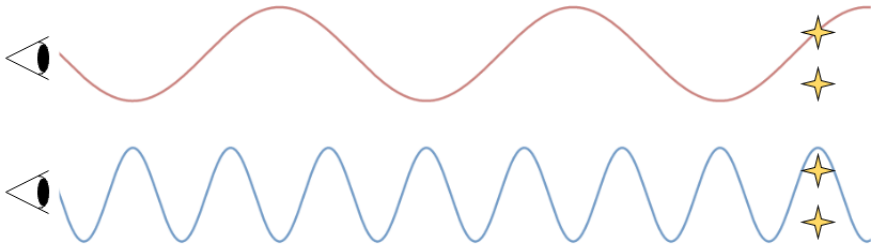
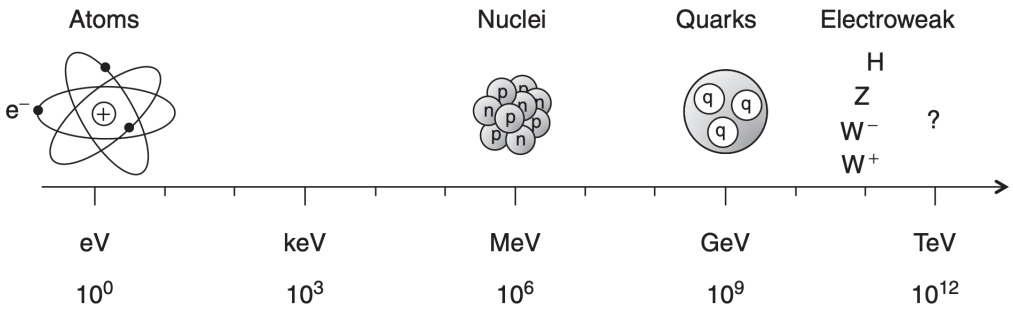


Figure 1.4: *The nature of observation of particles at different energy scales.[5]. At the eV range, we can observe atoms; at the MeV range, we can study the nuclei and quarks at the GeV scale, and finally, the electroweak sector, the Higgs boson, and new physics.*



Since we have gained an understanding of HEP, it is now time to discuss particle colliders. In order to comprehend the composition of something, it is possible to smash those objects together and examine the remnants of the collision. For instance, if we were to crash two cars, we would obtain the vehicle's various parts as the outcome. If we were to crash them at a low speed, we would get nearly two complete cars, which is not very informative. However, if we were to crash them at a very high speed, we may even break the engine and examine its components. This is certainly not the most effective method for understanding the constituents of a car, but it is an excellent method for studying small

particles.

This is essentially what physicists do with particle colliders; they accelerate the particles to nearly the speed of light and collide them head-on. One significant difference from car collisions, although not the only one, is that we can obtain new particles that were not present in the initial state. This is a direct consequence of Equation 1.1.

There are various types of particle accelerators. The first decision to make is which particles to collide with. The two most common options are proton-proton (pp) and electron-positron (e^+e^-) pairs. Both types have their advantages and disadvantages. Since electrons/positrons do not possess an internal structure, they collide with the precise energy they are tuned to. This makes them highly suitable for precision measurements. Additionally, due to the absence of an internal structure, the output of the collisions is less noisy compared to pp collisions. However, a disadvantage of e^+e^- collisions is the limitation on the energy that the particles can carry, resulting from the low mass of electrons/positrons.

Since the internal structure of protons consists of quarks and gluons, the energy of the colliding particles is not deterministic. These internal particles are referred to as partons, and various parton distribution functions are used to estimate the interaction energy and probability of collision for the individual particles. This internal structure results in only a fraction of the energy being utilized in the collision. However, due to protons having a much higher mass than electrons, they are less susceptible to synchrotron radiation. As a result, significantly higher energies can be achieved compared to e^+e^- accelerators.

The second factor to consider when designing a particle accelerator is whether to utilize two colliding beams or a fixed target with a single beam. One of the most crucial quantities in a collider experiment is the center of mass energy, denoted as $\sqrt{s}$. The Lorentz invariant quantity, s, is expressed in natural units (c=1) as:

$$s = (\sum_{\mathrm{i}=1}^{2} E_{\mathrm{i}})^2 - (\sum_{\mathrm{i}=1}^{2} \vec{p}_{\mathrm{i}})^2 \tag{1.3}$$

The summation indices represent the two colliding particles, and $\vec{p}_i$ denotes the three-dimensional momentum vector. It is known that both energy and momentum are conserved in any collision. Let's first consider a fixed-target experiment involving two protons.

$$s = (E + m_{\mathrm{p}})^2 - p^2 = E^2 - p^2 + 2Em_{\mathrm{p}} + m_{\mathrm{p}}^2 = 2m_{\mathrm{p}}^2 + 2Em_{\mathrm{p}} \approx 2Em_{\mathrm{p}} \tag{1.4}$$

If we take $E = 7$ TeV, the calculation in Equation 1.4 yields $\sqrt{s} = 115$ GeV, indicating a significant energy loss. Now, let's consider the case of two colliding beams. In this scenario, the momentum vectors have equal length but opposite directions, resulting in their cancellation. Consequently, what we are left with is given in Equation 1.5.

$$s = (E + E)^2 = 4E^2 \tag{1.5}$$

Again, if we take $E = 7$ TeV, then $\sqrt{s} = 14$ TeV. It is evident that the latter configuration yields a much higher center of mass energy. Therefore, we will employ the two colliding beams setup in our simulation, specifically the LHC at The European Organization for Nuclear Research (CERN) Geneva. LHC is designed to work with a collision energy of 14 TeV. In our simulated dataset, we will be using a collision energy of 13 TeV.

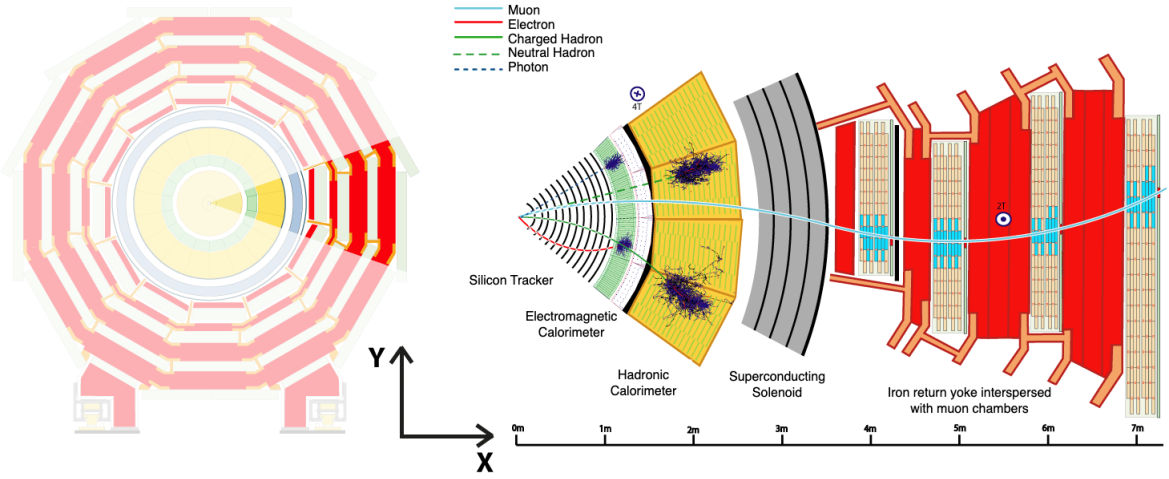
1.3 Particle Detectors

Inspection of the output from collisions is not straightforward. Particle detectors collect the data from collisions in the center of the detector. Many types of particle detectors are used worldwide to detect many different particles; discussing one general-purpose

detector, the CMS detector in LHC, is best.

A particle detector is similar to a cylindrical onion, with each layer, or sub-detector, detecting different particles or characteristics. The particles passing through these sub-detectors create digital (RAW) signals. We can obtain a detailed representation of an event by reconstructing these layers of various signals with the use of the Particle Flow (PF) algorithm. Figure 1.5 shows a transverse slice of CMS and how the various particles are reconstructed.

Figure 1.5: *A transverse slice of the CMS detector and its subdetectors. Reconstruction of the particle flow (PF) objects is represented on the right side.*



The sub-detectors of the CMS are as follows from the interaction point to outwards: tracker, Electromagnetic Calorimeter (ECAL), Hadronic Calorimeter (HCAL), and muon detectors. We will now go over each sub-detector.

When a charged particle passes through a medium, it ionizes the atoms and releases electrons along its path. A silicon tracker detects this ionization using layers of silicon micro-strip sensors, enabling us to reconstruct the trajectory of any charged particle with great precision.

Calorimeters are designed to halt and absorb the energy of passing particles. Typically, two types of calorimeters are used: Electromagnetic and Hadronic calorimeters.

ECAL primarily detects electrons and photons and consists of a grid of scintillators. When an electron passes through a scintillator, it slows down and emits a bremsstrahlung photon. This photon then generates an electron-positron pair. This process continues until the particle comes to a stop, resulting in an electromagnetic shower. This shower interacts with the scintillator, exciting its molecules. The decay of these excited states emits ultraviolet light, which is amplified and collected by photon detectors to reconstruct the particle's energy.

Charged hadrons, such as protons and charged pions, also deposit energy through ionization. However, regardless of whether they are charged or neutral, all hadrons interact with the nucleus of the detector material through strong interaction. An HCAL has alternating layers of an absorber and an active material, such as a scintillator. The hadronic shower occurs within the thick and dense absorber, followed by a thin layer of active material (scintillator) where the energy losses of charged particles are measured similarly to ECAL.

This inner detector (Tracker, ECAL, HCAL) is surrounded by a large superconducting solenoid that generates a uniform magnetic field perpendicular to the transverse plane. (This fact is valid for the barrel part of CMS, but we will be working only with the barrel.) This magnetic field exerts a transverse force on the charged particles, causing curvature in their trajectories. The formula for the magnetic force is given in Equation 1.6.

$$\vec{F}_B = q(\vec{v} \times \vec{B}) \tag{1.6}$$

Where q represents the charge of the particle, $\vec{B}$ is the applied magnetic field, $\vec{v}$ is for the velocity of the particle, and θ represents the angle between the velocity and

magnetic field vectors. The direction of the curvature can indicate whether the particle is positively or negatively charged. By examining the magnitude of the curvature, which is determined by $v \sin \theta$ (the projection of the particle's velocity onto the transverse plane), we can determine the particle's transverse momentum (p_{T}). Conversely, neutral particles do not have curvature or trajectories because they do not interact with the magnetic field or the tracker.

Outside the solenoid, there are muon detectors and iron return yokes. These return yokes serve two purposes. Firstly, they complete the uniform magnetic field lines inside the solenoid by forming a loop. This is their primary function, providing a uniform and desired path for the magnetic field lines. As a result, they generate a reverse magnetic field, causing the curvature to be reversed, as shown in muon paths in Figure 1.5. The return yoke's second purpose is to absorb muons. There are various types of muon detectors, but they all rely on charged particles ionizing the path they traverse. We will not delve into the details, as the principle is similar to that of the Tracker.

Each type of detector is strategically placed to "capture" the desired particle. The positions of these detectors are determined by considering the particle's lifetime. One might wonder why muons are not detected in the ECAL, as they are also charged. The muons produced in particle accelerators are highly energetic, resulting in time dilation. Consequently, their lifetime in the laboratory frame is extended. This is why muon detectors are positioned as the outermost layer.

1.4 Jet Physics: Formation and Characteristics

A jet is a collimated spray of stable particles commonly observed when quarks or gluons are produced in high-energy particle collider experiments. Jets are not distinct entities but depend on the parameters and reconstruction algorithms used.

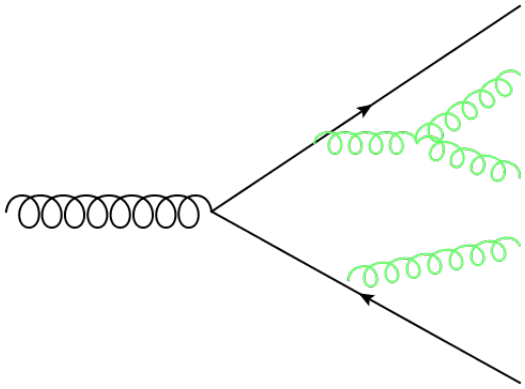
As mentioned in Section 1.1, color confinement requires that all particles with

color charge form color-neutral groups known as hadrons. Quarks and gluons, which are essential components of the SM, are frequently produced in hadron colliders. Due to the high collision energies, these produced quarks and gluons also possess high energies.

Two significant phenomena of the strong force play crucial roles in jet formation. The first is asymptotic freedom, which states that the strong force weakens as the energy increases or the scale decreases [3, 6]. The second is color confinement, where the strong force grows stronger as the distance between particles increases, unlike electromagnetism or gravity. This behavior can be compared to an elastic band: as the distance between quarks increases, the energy of the strong force also increases.

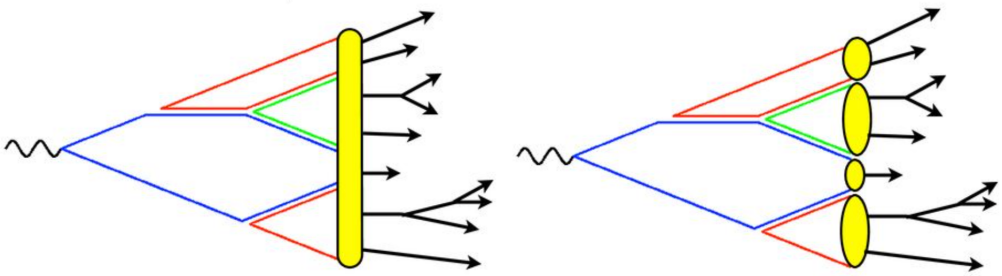
Jet formation occurs in two stages: the parton shower stage and the hadronization stage. In the parton shower stage, a high-energy collision produces partons (quarks or gluons) that move apart due to momentum conservation. As the distance between partons increases, the strong force becomes so intense that it is energetically more favorable to produce additional quark-antiquark pairs or gluons than to maintain the force between the separating partons. The process happens through the emitting of gluons, shown in Figure 1.6. The quarks decay into another quark and a gluon to preserve the color charge of the event; the gluons can also split into other gluons, etc.

Figure 1.6: *Feynman diagram of a parton shower, a gluon splitting into two quarks and then other gluons splittings create the particle shower.*



This cascading process continues until the energy of the emitted particles and their relative angles become small enough that further splitting is unlikely. At this point, non-perturbative QCD effects like hadronization dominate. The partons form bound states (hadrons) as a result of confinement, producing a collimated spray of stable hadrons—a jet. Determining the flavor of the originating parton is challenging due to the decay and subsequent formation of particle groups. The Color preserving nature of the parton shower and two types of clustering algorithms are shown in Figure 1.7.

Figure 1.7: *String and clustering hadronization methods. As single-color charged quarks decay through gluons, they lose energy and hadronize. Gluons are represented with double lines.[7]*

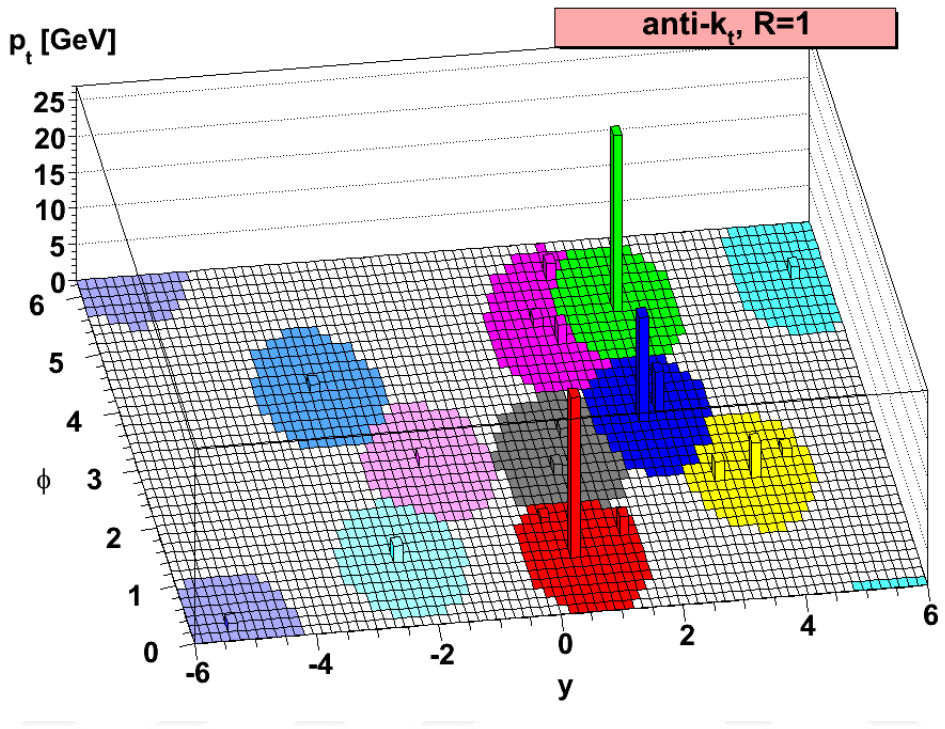


As mentioned, the definition of jet in each event depends on the reconstruction algorithm. One of HEP's most commonly used jet reconstruction algorithms is the anti-k_T algorithm [8] given in Figure 1.8.

Antik_T is an Infrared and Collinear (IRC) safe algorithm. These two concepts are the common requirement of a good jet reconstruction algorithm [9]. The definition is best given without alteration: "An observable is IRC safe if it is insensitive to infinitesimally soft or exactly collinear emissions" [10]. This means the low-energy (infra-red) emissions will not change the jet reconstruction nor the splitting of a particle into two collinear particles.

Although we have talked about jet formation, the definition of a jet in data is still ambiguous. This issue will be addressed in Chapter 2 when the data preparation will be

Figure 1.8: *A sample parton level event clustered with anti-k_T algorithm. Here the colored circles represent the 2D image of the reconstructed jet given with ΔR parameter.[8]*



presented.

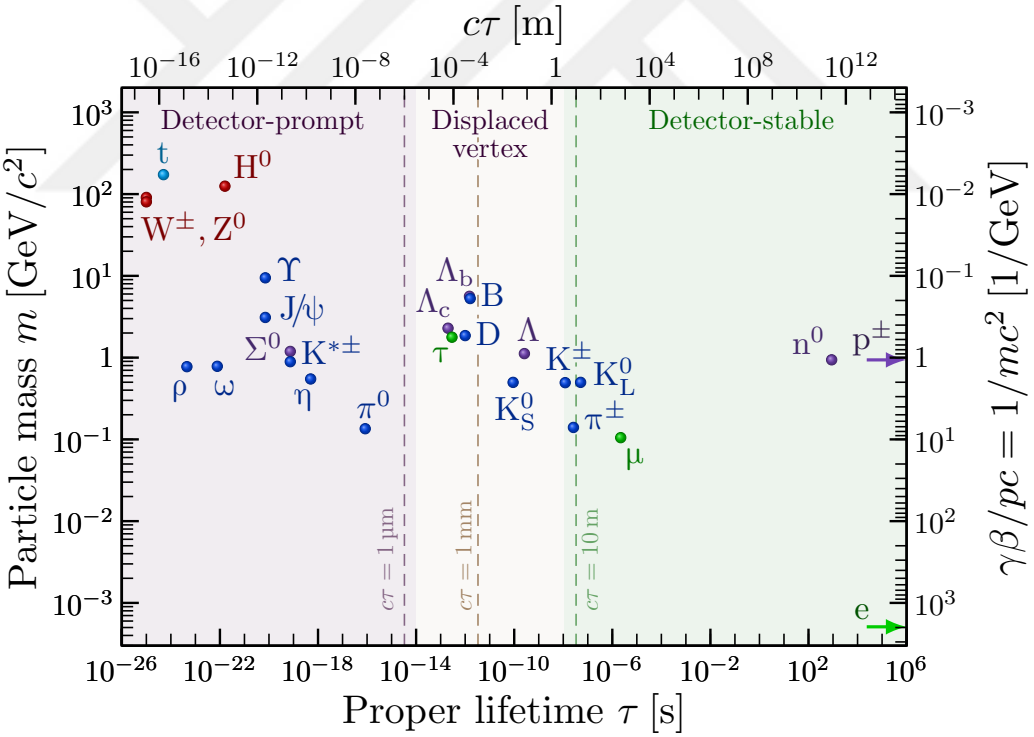
1.5 Significance of b Quark Originated Jet Tagging

Since the operation of the LHC and other particle colliders began, physicists have been able to study the first two generations of particles extensively. As technology has advanced, the maximum center of mass energy achieved by these colliders has gradually increased, allowing for the production of heavier particles such as the top quark, Higgs boson, W boson, and Z boson. However, our knowledge about these heavy particles is limited due to their relatively recent discovery. Therefore, further investigation and production of these particles may lead to expanding the SM or discovering new physics.

One challenge in detecting these particles is their short lifetime due to their high

mass. The lifetime of a particle is not directly related to its mass but rather to the forces responsible for its decay. Figure 1.9 shows a correlation between a particle's mass, its proper lifetime (τ), and the mean distance it travels before decaying ($c\tau$). In this context, τ represents the mean lifetime of the particle in its rest frame. For example, the top quark has a lifetime of approximately 5×10^{-25} seconds, while the Higgs boson has a mean lifetime of 1.6×10^{-22} seconds. But B-hadrons are in the range to create a displaced vertex in the detector. It is essential to mention that all the lifetimes mentioned are mean, as particle decays are statistical.

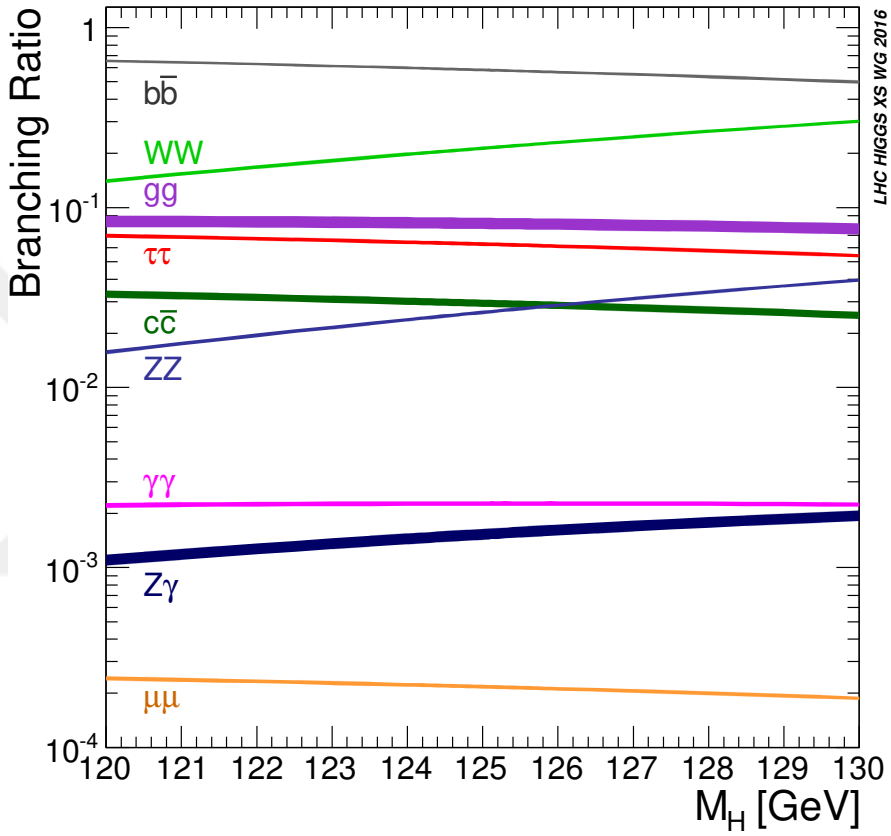
Figure 1.9: *Particle Mass vs. proper lifetime and mean distance traveled graph for some of the elementary particles and some hadrons. Showing also the interaction with the detector.[11]*



One can examine the Branching Ratio (BR) of these particles, for example, the top quark and the Higgs boson. BR represents the probability of a specific decay mode

occurring among all possible decay modes. The top quark almost always ($\approx 100\%$) decays into a W boson and a bottom quark [12]. Furthermore, the Higgs boson also has a high probability to decay into a $b\bar{b}$ pair, as shown in Figure 1.10.

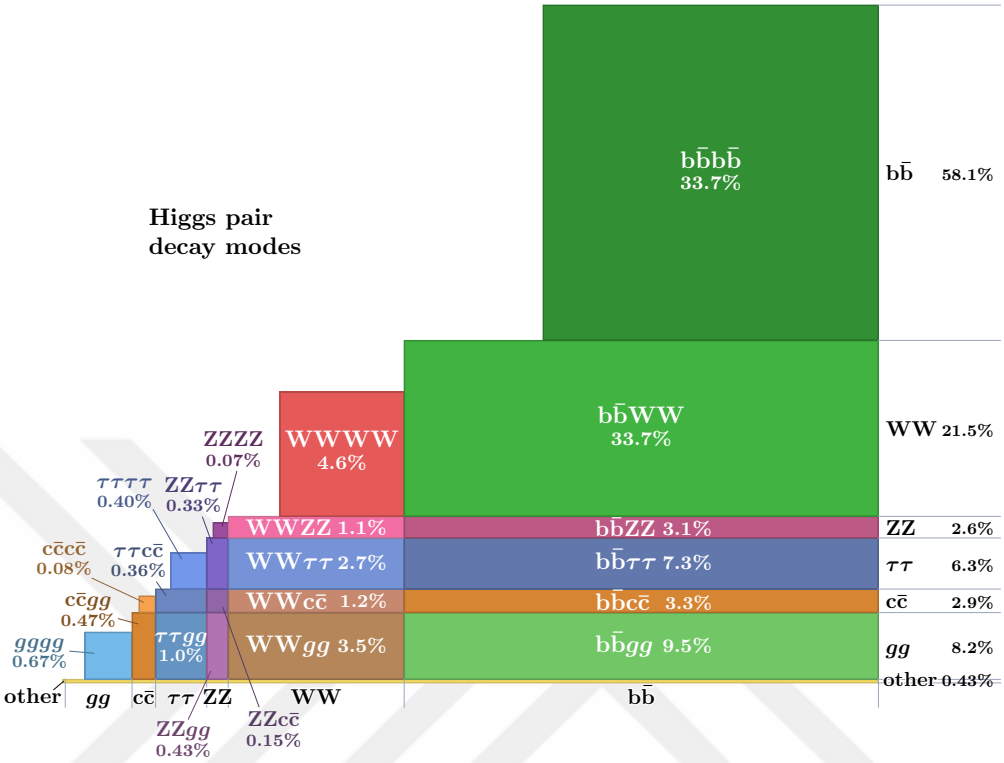
Figure 1.10: *Branching Ratios of the Higgs Boson for various masses. $b\bar{b}$ is seen to be the dominant channel.[12]*



We can also examine the decay products of a Higgs boson pair, where the decay modes are visualized in Figure 1.11. With more than 90% probability, a $b\bar{b}$ pair will be generated.

Understanding the properties of these two particles is essential to explore the mysteries of the particle physics. The Higgs boson is speculated to be a portal to the dark sector, meaning since the Dark Matter particles are massive, they should interact with the Higgs boson [14]. Hence, if we are to observe a Dark Matter candidate, it can decay to

Figure 1.11: *Stack visualization of the decay modes of a Higgs boson pair[13].*



SM particles through the Higgs mechanism. And since Higgs boson is too short-lived for it to interact with the detector, we can only detect it via its decay products.

The lifetimes of different quarks are an important discriminator for flavor tagging. Up and down quarks are stable particles that don't decay into lighter particles. Strange quarks live long enough to travel in the order of meters. Top quarks live so short that they can only travel in the order of $10^{-}16$ meters, which is around 1/10.000th the diameter of the proton. But, bottom and charm quarks travel in the order of millimeters, the charm being slightly shorter. This distance approximately corresponds to the tracker detectors in most particle accelerator detectors, like A Toroidal LHC Apparatus (ATLAS) and CMS in LHC. Thus, we get detailed information on tracks and secondary vertices of jets originating from bottom quarks and charm quarks to some extent.

To summarize, the ”interesting” particles that we want to inspect cannot be detected directly, but they decay into b quarks most of the time. B hadrons decay either very close to or within the detector, making them detectable. Tagging b-jets might benefit both the detection side and the analysis side. Using trigger algorithms that use b-tagging models would be highly beneficial in collecting all the Higgs and top quark events. On the analysis side, selecting b-jets in this way would also drastically reduce the background for certain analyses, making it possible to make more precise measurements.

2. DATASET PREPARATION FOR B-JET TAGGING

The selection of the dataset proved to be one of the most crucial elements of the model's performance during this thesis's work. We tried many different configurations, parameters, and events to achieve a pure and realistic dataset.

In this chapter, the details of the dataset preparation process will be discussed. Section 2.1 will address the event selection and generation. Then, feature extraction will be discussed in Section 2.2. Section 2.3 gives the details of the pre-processing step, which prepares the dataset for the model training. Finally, Section 2.4 will elaborate on the selection of the parameters.

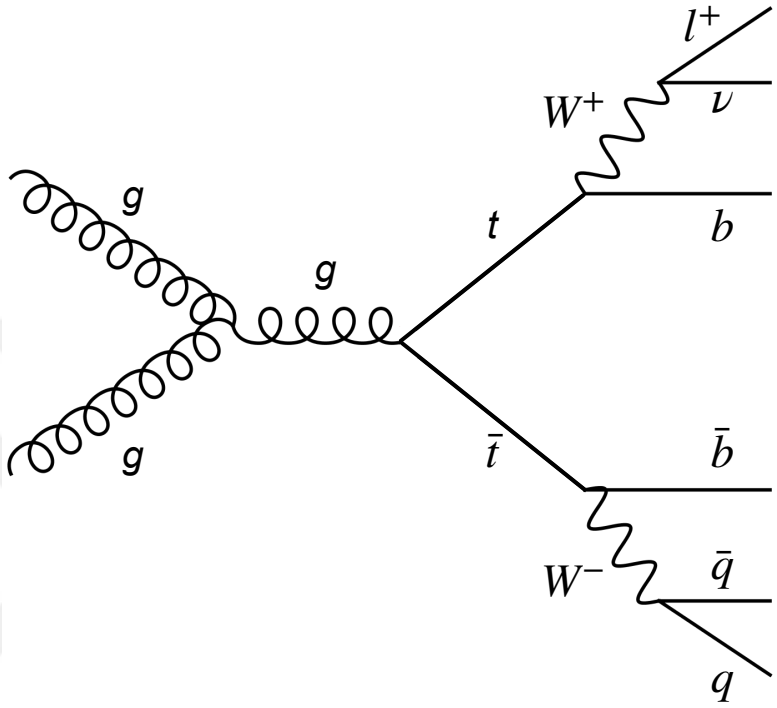
2.1 Data generation

The true flavor of the originating parton in a jet in an actual collision event is unknown. Hence, Monte Carlo event generators are used to produce a labeled dataset that is pre-processed and trained with supervised deep-learning models.

Semi-leptonic pp $\rightarrow$ t$\bar{t}$ decay event is used at 13 Tev, given in Figure 2.1. In this diagram, the right-hand side of the red dot is arbitrary and can be an interaction of any parton in the colliding protons. One of the reasons behind this selection is that the LHC, with its unprecedented collision energy and high luminosity, is often referred to as a 'top quark factory.' It produces an abundant number of top-antitop ($t\bar{t}$) quark pairs through proton-proton collisions, primarily via gluon-gluon fusion [15, 16].

The semi-leptonic channel includes a charged lepton (e or μ) and missing transverse energy from the neutrino are distinguishable from pure QCD backgrounds compared to fully hadronic decays. This channel introduces diversity in the background while still providing sufficient statistics, as it avoids the complete reliance on neutrino reconstruction like the di-leptonic channel [18]. A representative image is given in Figure 2.2 for

Figure 2.1: *Feynman diagram of a semi-leptonic $t\bar{t}$ decay. The upper top quark decays into a b quark and a W boson, which further decays leptonically. Lower anti-top decays into anti-b and a W boson, which decays into two quarks hadronically. The left part of this event before the top production is irrelevant for us.[17]*

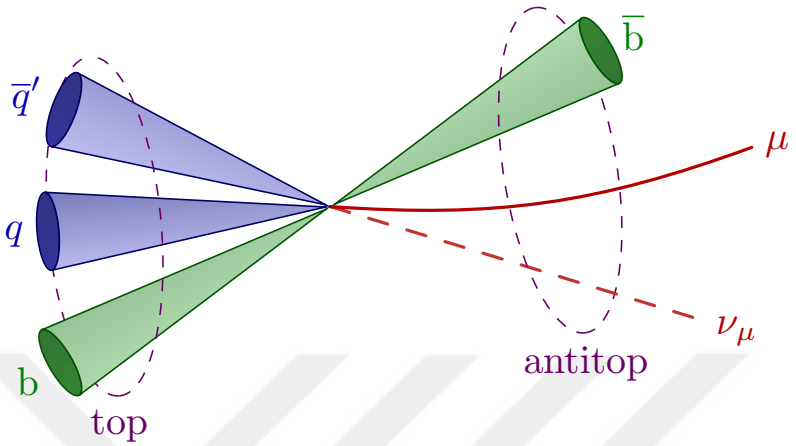


this event.

Decay modes of the $t\bar{t}$ are also given as a pie chart in Figure 2.3. Around 45% of the events are semi-leptonic, which we will be working with.

We have first generated a dataset at the parton level using MADGRAPH5[21]. Then, the parton showering and hadronization is simulated using PYTHIA8[22] with Initial State Radiation (ISR), Final State Radiation (FSR), and Multi-Parton Interaction (MPI) all allowed. Finally, DELPHES[23] is utilized to simulate the effects of the CMS detector and also reconstruct the jet objects. Default parameters are used except in Delphes. The TrackSmearing module is added to the default CMS card to be able to extract $d_0, d_z, \sigma_{d_0}, \sigma_{z_0}$ parameters of each track. For the jet clustering AntikT algorithm [8] with $\Delta R = 0.4$ is preferred, where ΔR is defined in Equation 2.1

Figure 2.2: *Visualization of the semi-leptonic $t\bar{t}$ decay. The top quark on the right decays to a b quark, and two additional quarks from the W boson form jets. And on the right, anti-top decays into a b quark producing jets, and a muon and its neutrino. [19]*

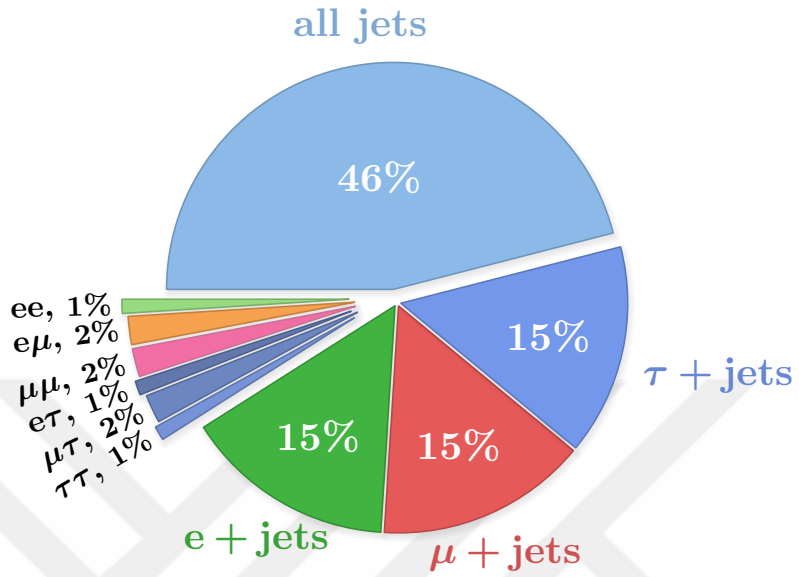


$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} \tag{2.1}$$

We first trained our models with this dataset. while it provided a good baseline to train our models, we recognized that using DELPHES for detector simulation has inherent limitations, particularly in modeling track parameters such as d_0, d_z, σ_{d_0}, and σ_{z_0}. After taking some expert opinions, we decided to use a dataset that has been simulated using GEANT4[24], which is a full detector simulation, whereas DELPHES is a fast detector simulation. The difference mainly arises from the track-smearing procedure. The TRACKSMEARING module in DELPHES offers an approximation of detector effects but does not fully capture the track-smearing significance as accurately as GEANT4.

We turned to the CERN CMS Open Data portal to address these limitations. We selected a dataset derived from a full simulation of $t\bar{t}$ events produced at $\sqrt{s} = 13$ TeV, which includes both realistic detector effects and track information. Specifically, we used the dataset RunII Monte Carlo (MC): TTJets_TuneCUETP8M1_13TeV-powheg-pythia8 [25]. This dataset was generated using POWHEG for parton-level event generation,

Figure 2.3: *Pie chart to show the decay modes of the $t\bar{t}$ decay. We see 46% of the time they decay fully hadronic and around 45% of the time they decay semi-leptonically.[20]*



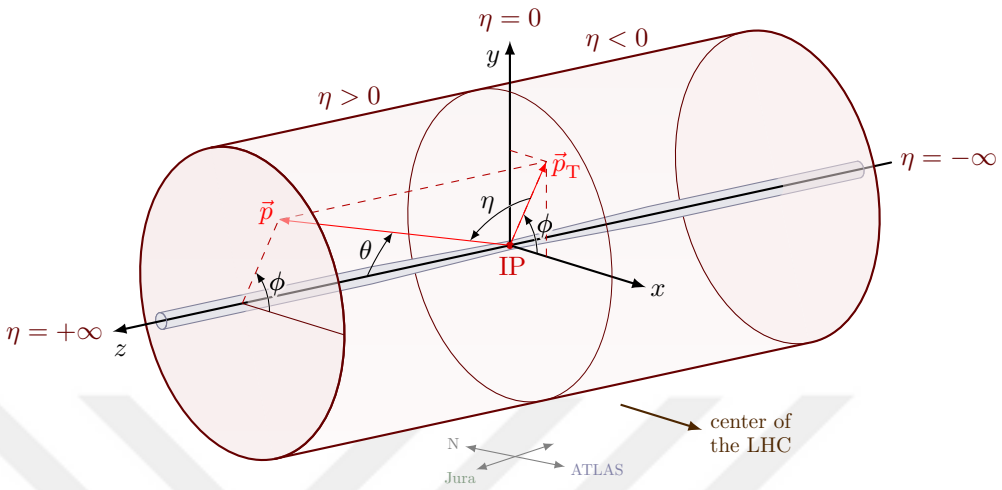
PYTHIA8 for parton showering and hadronization, and the CMS software framework with GEANT4 for a detailed detector simulation.

2.2 Feature Extraction

This study aims to construct a jet-based tagger rather than an event-based tagger, so we extract only the jets from each event. Jets are complex objects: They both have jet-level information (Jet energy, Jet p_{T}...) and information about the jet's constituents. These constituents are composed of tracks, secondary vertices, calorimeter towers, etc., but this thesis only utilizes track and secondary vertex features as well as a few global jet parameters. The coordinate system that will be used when defining the parameters is given in Figure 2.4.

During the extraction of jet parameters from the events, we apply a few cuts to ensure the jets we use are in the efficient region of the CMS detector. We apply a cut

Figure 2.4: *The coordinate system of the CMS detector, where IP is the interaction point. The important parameters are the rapidity η and the angle ϕ.[26]*



to jet p_{T} of $30GeV$ for each jet, meaning we discard the jets from our dataset if its jet $p_{\mathrm{T}} < 30GeV$. We apply another cut to jet η as well by removing the jets that have $\eta > 2.5$.

After applying the cuts on the jet level, we require other conditions from each track. We take the jet this time, but only the tracks with track $p_{\mathrm{T}} > 1GeV$ and a normalized $\chi^2 < 5$ are, assigned to the jets. This is done to speed up the pre-processing step that will follow after this step and remove the uninformative parameters from our dataset. This step ensures that we extract the highest quality tracks from the jet.

We do not apply cuts to the secondary vertices as there are already fewer SVs, and we don't have the same quality problem as in the track parameters. Although the tracks in an event are already associated with each jet, the same procedure for vertices has to be done manually. We do this by identifying the primary vertex of the jet; then, we assign each secondary vertex that is within $\Delta R < 0.4$ of the primary vertex. This choice makes sense as the jets are already clustered using $antik_T$ algorithm with $\Delta R = 0.4$, so if a secondary vertex is within this cone, it is highly logical to assume that particular

secondary vertex emerges as a result of the jet formation. Of course, there are exceptions to this role, but within the context of this thesis, the assumption is valid enough.

After extracting these parameters from the output of GEANT4 simulation as a root file, we save the parameters to a ROOT TTREE and save them to another root file. Pre-processing of the parameters follows these operations.

2.3 Pre-processing

Since not all jets go through the same processes during jet formation, the number of tracks and the number of Secondary Vertex (SV) in each jet is a variable number. This causes some problems with model training for most models, even making it impossible to train without pre-processing the data. The reason is that most Machine Learning (ML) architectures are suitable for only fixed-length input sizes. Some models can handle variable input sizes, but to be able to try out more models with the same dataset, we prefer to fix the input size.

To solve this obstacle, we first sort the tracks and SVs in each jet in descending p_{T} order. The descending p_{T} ordering is chosen with the following reasoning: Unless there are new collisions, the decay products of the particles will have less momentum than the decaying particle. With this logic, the tracks with very low momentum will either appear at the end of the jet formation process or are a result of ISR and FSR processes, thus containing less information on the originating particle. Since the aim of a jet-tagging task is to find the originating parton, it is acceptable to remove these low-information-containing parameters from our dataset for the sake of inference speed.

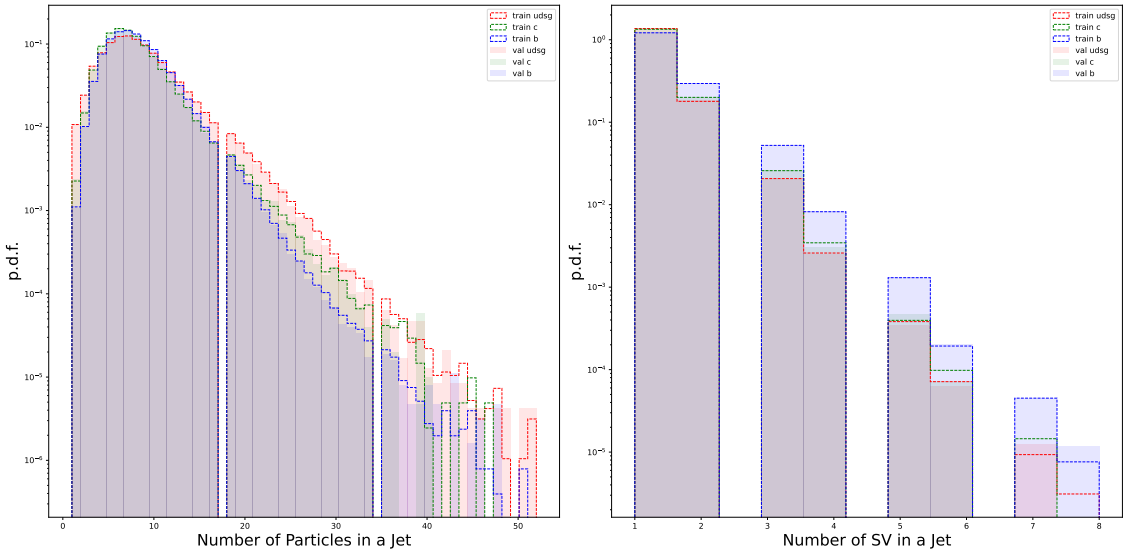
We select a limit to the number of particles in a jet to 16 and the number of secondary vertices to 5. The density plots of the number of tracks and SVs in the jets are given in Figure 2.5.

Then, if the number of tracks in a jet is more than 16, the tracks after 16^{th} are

discarded. If the number of tracks is less than 16, then 0 values are inserted (padded) into each track parameter until there are 16 tracks in the jet. These procedures are called cutting and zero-padding, respectively.

The same pre-processing procedure is also applied to SVs, but the maximum number of SVs is selected as 5, then sorting by decreasing SV p_{T}, followed by cutting/padding the parameters. These procedures are called cutting and zero-padding, respectively. This information on the number of padded tracks or SVs is also saved for later use in model training and scaling as a padding mask.

Figure 2.5: *Histograms of the number of tracks (left) and number of SV (right) in each jet. The cuts are applied at 16 tracks and 5 SV.*



5 million jets are then split into training and validation datasets with 0.8 and 0.2 ratios, respectively. Then, each parameter's mean, μ, and standard deviation, σ, values are extracted from the training dataset.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \qquad z = \frac{x-\mu}{\sigma} \tag{2.2}$$

Note that the zero-padded values are not used in the calculation of μ and σ. With

these, each parameter in both datasets is standardized, as shown in the left-most part of Equation 2.2. The normalized datasets are now ready for the training with ML models.

2.4 Parameter Selection

The number of features is kept at a minimum for two main reasons: Firstly, to create simpler models that run with only trivial information easily accessible by the detector. Due to computational limitations, human-made parameters such as jet probability are not used for this purpose. Complex models such as Transformers require vast computational resources, and the inference time becomes unrealistic as the number of parameters increases.

As the earlier sections stated, a jet has jet-level, track-level, and SV-level features. In this work, all three types of information are used. Hence, we define a jet object as having 16 arrays for each track feature, five arrays for each SV feature, and jet features of only single values.

Before delving into the selected parameters, it is best to discuss the flavor of the originating parton first, as this will be the label for our dataset. Hence, the correct assignment is of paramount importance. There are two types of jet flavor in the dataset taken from the Open Data Portal: hadron flavor and Parton flavor. The first one comes from the hadron ghost clustering [27]. This procedure is employed by the CMS collaboration and selected as the true flavor of the jets [28].

We first extracted this parameter hadron flavor from the dataset for each jet. The flavor of the originating parton can be up, down, charm, strange, bottom quarks and their anti-quarks, or it can be from a gluon. Since our task is b-tagging, we only care about whether the jet originated from a bottom quark or not. But to check the models' discrimination efficiencies against the charm quarks, we separate this flavor parameter into three classes: b-jet, c-jet, and light-jet, which contains u,d,s quark jets and gluon jets. We

do this because c-jets are also similar in nature to the b-jets as they have slightly longer lifetimes than the light-jets and produce similar secondary vertices, making it challenging to discriminate between b-jets and c-jets. We note the c-jet flavor, but we don't give this information to the model as our task is a binary classification task. So we map u,d,s,c quark and gluon jets to 0 and map b-jets to 1 to get our final labels for the dataset.

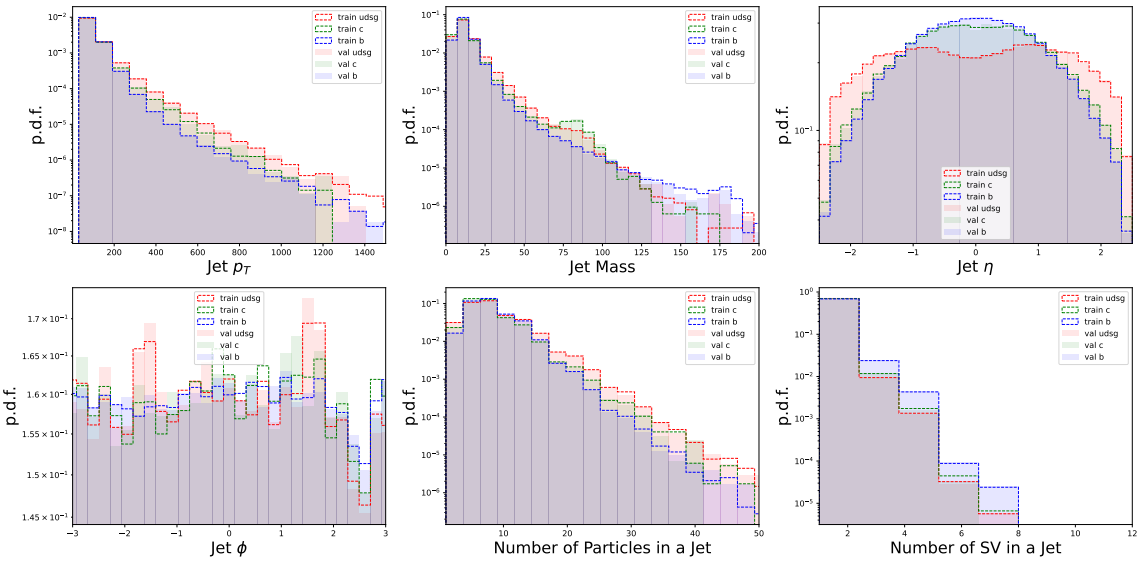
For the jet features, we have selected the following parameters; also, the density histograms for the distributions of jet features for the three flavors are given in Figure 2.6:

1. Jet p_{T}
2. Jet Mass
3. Jet η
4. Jet ϕ
5. Number of tracks/particles in the Jet
6. Number of Secondary Vertices in the Jet

For the track or, in other words, particle parameters, we selected the parameters listed below, also shown in Figure 2.8.

1. Track Energy
2. Track p_{T}
3. Track $\Delta\eta$
4. Track $\Delta\phi$
5. Track Particle Identification (PID) (Particle Identification)
6. Track Charge

Figure 2.6: *Density histograms of the jet parameters extracted from the dataset for three classes. Both training and validation sets are provided.*



7. Track d_0

8. Track σ_{d_0}

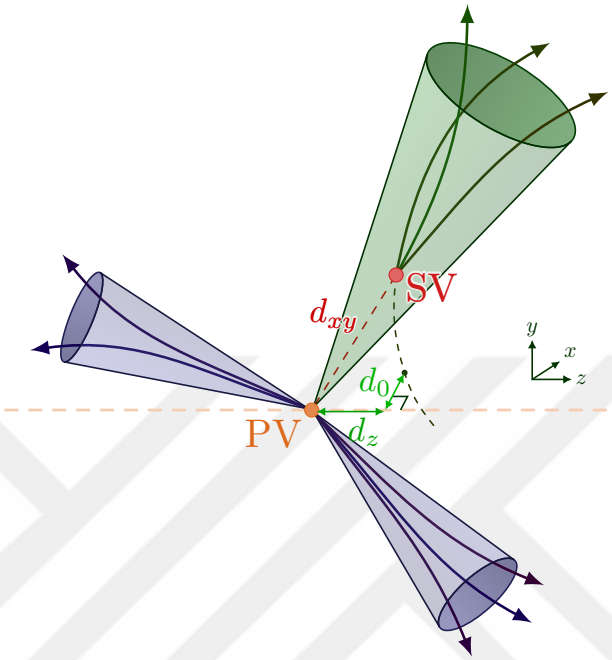
9. Track σ_{d_z}

Here $\Delta\eta$ ($\Delta\phi$) is defined as the difference of the track η (ϕ) and the jet η (ϕ). Track PID is the particle identification number of each particle [12]. The transverse and longitudinal impact parameters are d_0 and d_z, respectively. These two parameters are also visualized in Figure 2.7 and also defined in Equation 2.3.

$$d_0 = -\frac{x \cdot p_\mathrm{y} - y \cdot p_\mathrm{x}}{\sqrt{p_\mathrm{x}^2 + p_\mathrm{y}^2}}, \quad d_\mathrm{z} = z - \frac{(x \cdot p_\mathrm{x} + y \cdot p_\mathrm{y}) \cdot p_\mathrm{z}}{p_\mathrm{x}^2 + p_\mathrm{y}^2} \tag{2.3}$$

The significance of the impact parameters is also included; this parameter, in a way, measures the effectiveness of the impact parameters measurements and is defined as the distance between the closest approach of a track's trajectory to the primary vertex, as shown in Figure 2.7. These parameters are especially important in b-jet tagging since

Figure 2.7: *Transverse (d_0) and longitudinal (d_z)impact parameters of tracks are shown in figure. The SV's transverse decay length (d_{xy}) is also shown. PV is the primary vertex where the original event occurs.[29]*



b-quarks/B-hadrons have a long lifetime and travel relatively large distances ($\sim 5mm$). Therefore, they decay into particles with large impact parameters. More formally defined also in Equation 2.4.

$$\text{Significance}_{d_i} = \frac{d_i}{\sigma_{d_i}}, \quad i \in \{z, 0\} \tag{2.4}$$

From now on, we will refer to significance with the symbol σ_{d_i} for the simplification.

Finally, we have selected the SV parameters as follows:

1. SV p_T
2. Number of tracks in the SV
3. SV Mass

4. SV χ^2

5. Number of degrees of freedom in the SV

6. SV d_{xy}

7. SV d_{len}

8. SV $\sigma_{d_{\mathrm{xy}}}$

9. SV $\sigma_{d_{\mathrm{len}}}$

Where d_{xy} is the transverse decay length, and d_{len} is the 3D decay length.

This sums up all the used parameters. As already mentioned in Section 2.3, we have a variable number of each track and SV parameter depending on the jet constituents. We then sort and cut (or zero-pad) these arrays. Hence, we will provide information on the first track (SV), second track (SV), etc. Looking at these plots shown in Figures 2.6, 2.8, and 2.9 are informative, but these are not the actual distributions we provide to the model. The parameters for all tracks (SVs) are combined in these plots. We provide the same plots but for the specific tracks (SVs) to give a sense of what the model actually receives. Density plots for the track features of the first three tracks in the dataset are given in Figures 2.10, 2.11, and 2.12. Similarly, the density plots for each SV feature for the first three SV are given in Figures 2.13, 2.14, and 2.15.

Figure 2.8: *Density histograms of the track (particle) parameters extracted from the dataset for three classes. Both training and validation sets are provided.*

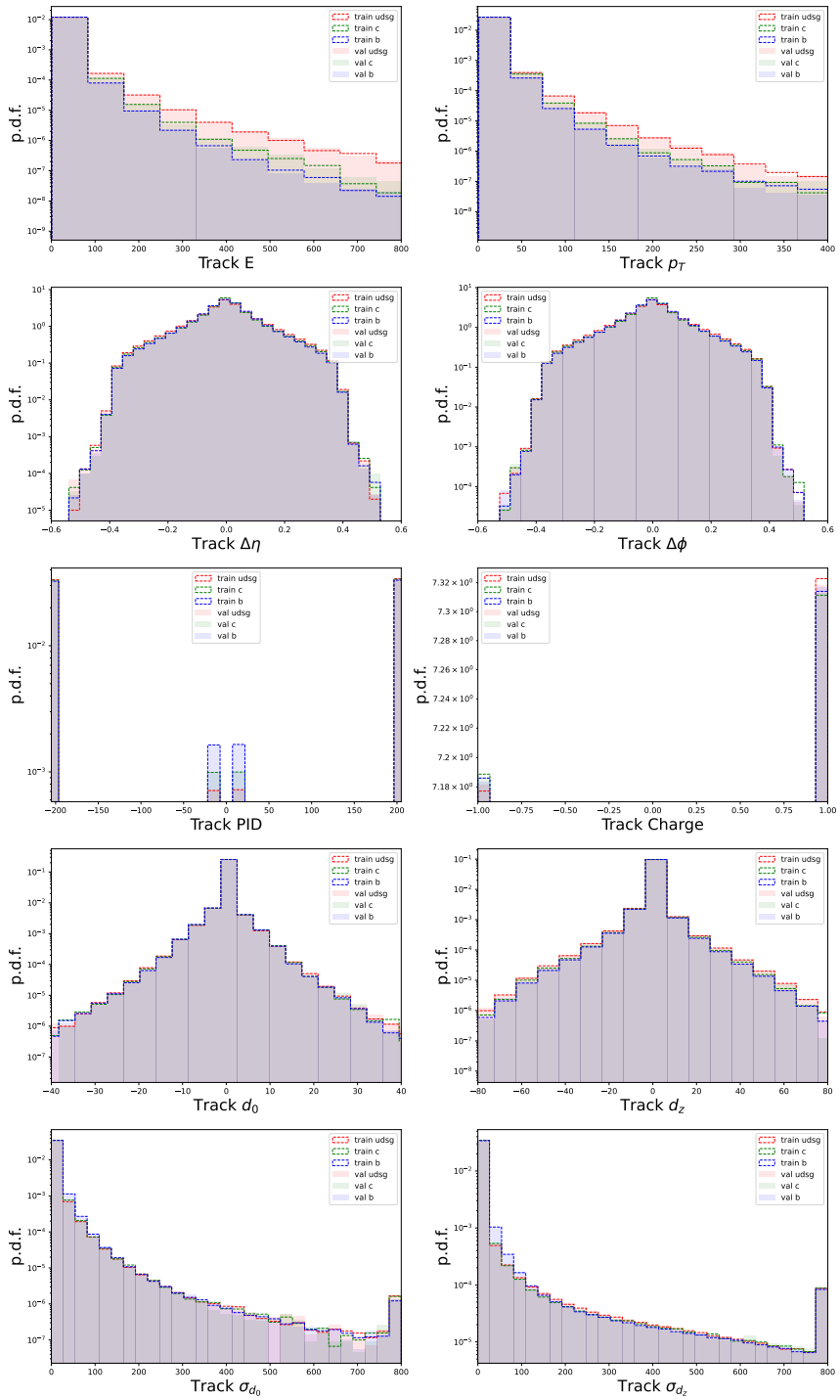


Figure 2.9: *Density histograms of the secondary Vertex parameters extracted from the dataset for three classes. Both training and validation sets are provided.*

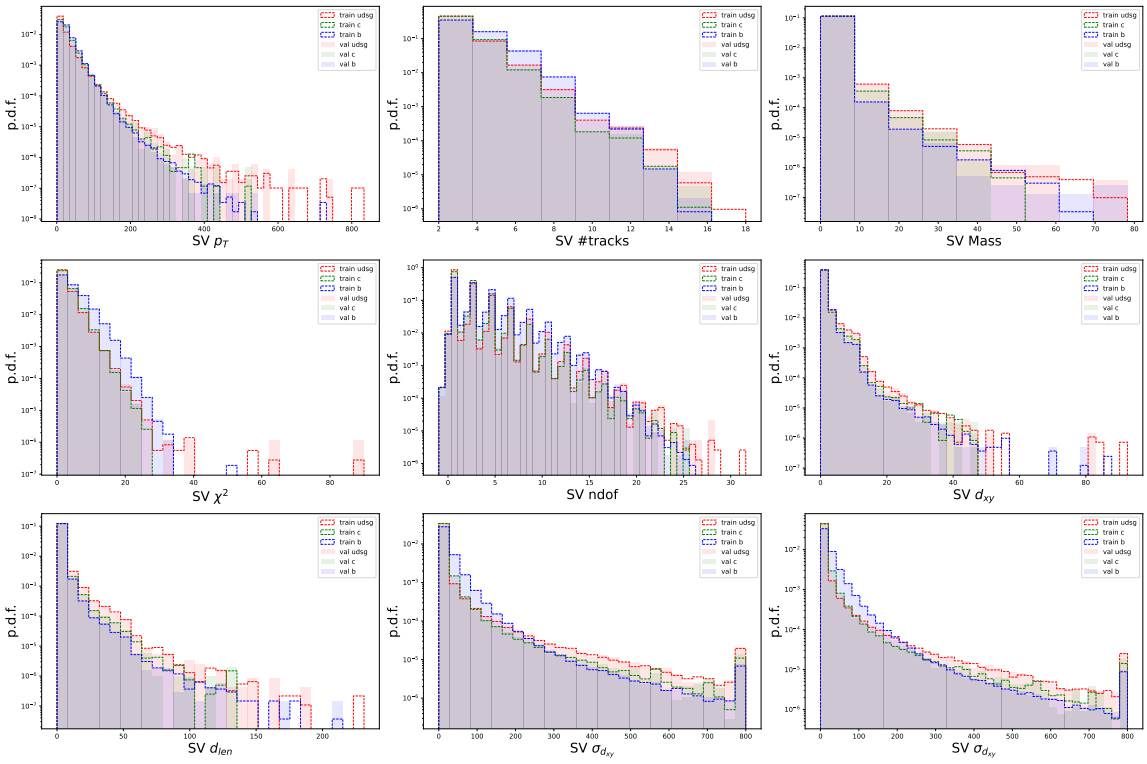


Figure 2.10: *Density histograms of the Track parameters of the first Track extracted from the datasets for three classes. Both training and validation sets are provided.*

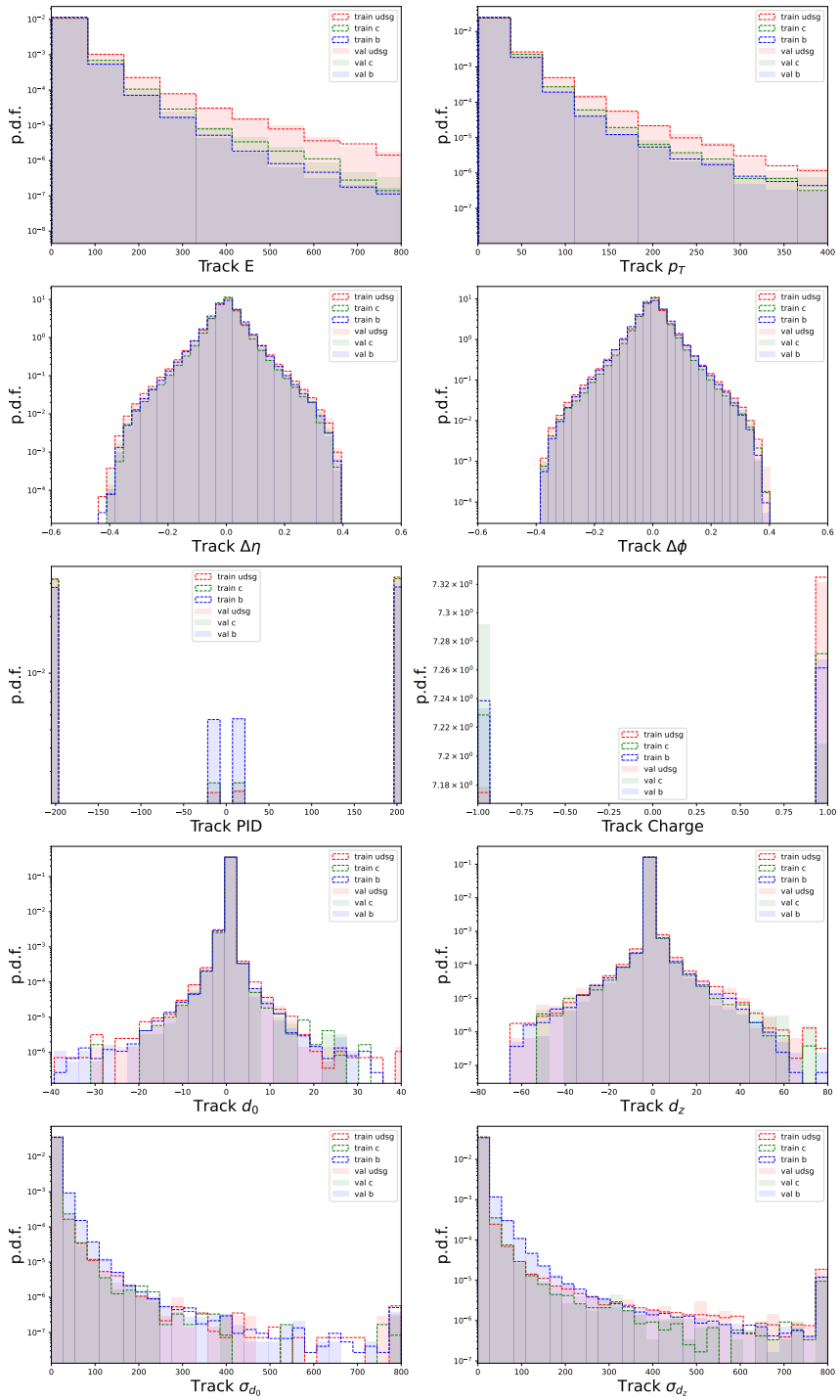


Figure 2.11: *Density histograms of the Track parameters of the second Track extracted from the datasets for three classes. Both training and validation sets are provided.*

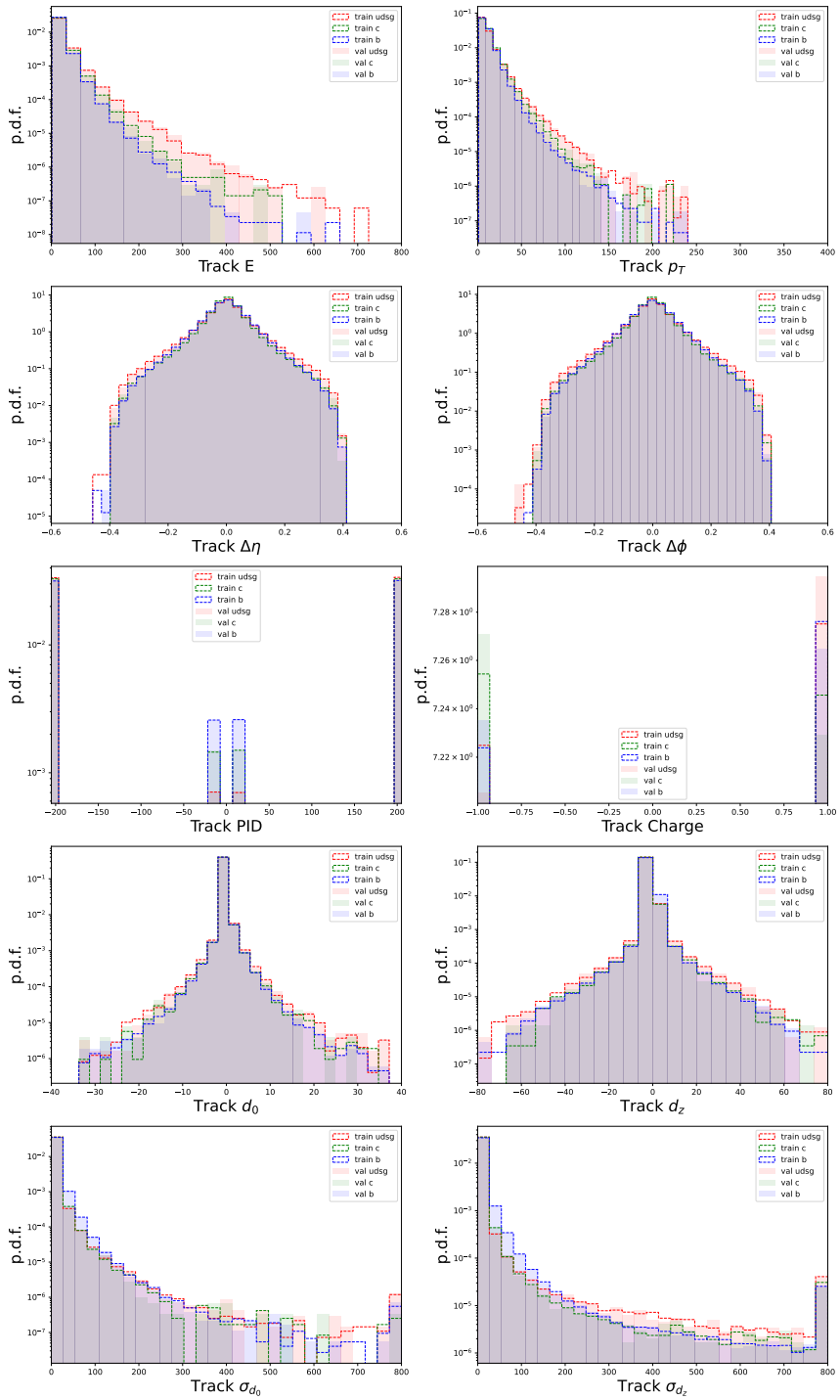


Figure 2.12: *Density histograms of the Track parameters of the third Track extracted from the datasets for three classes. Both training and validation sets are provided.*

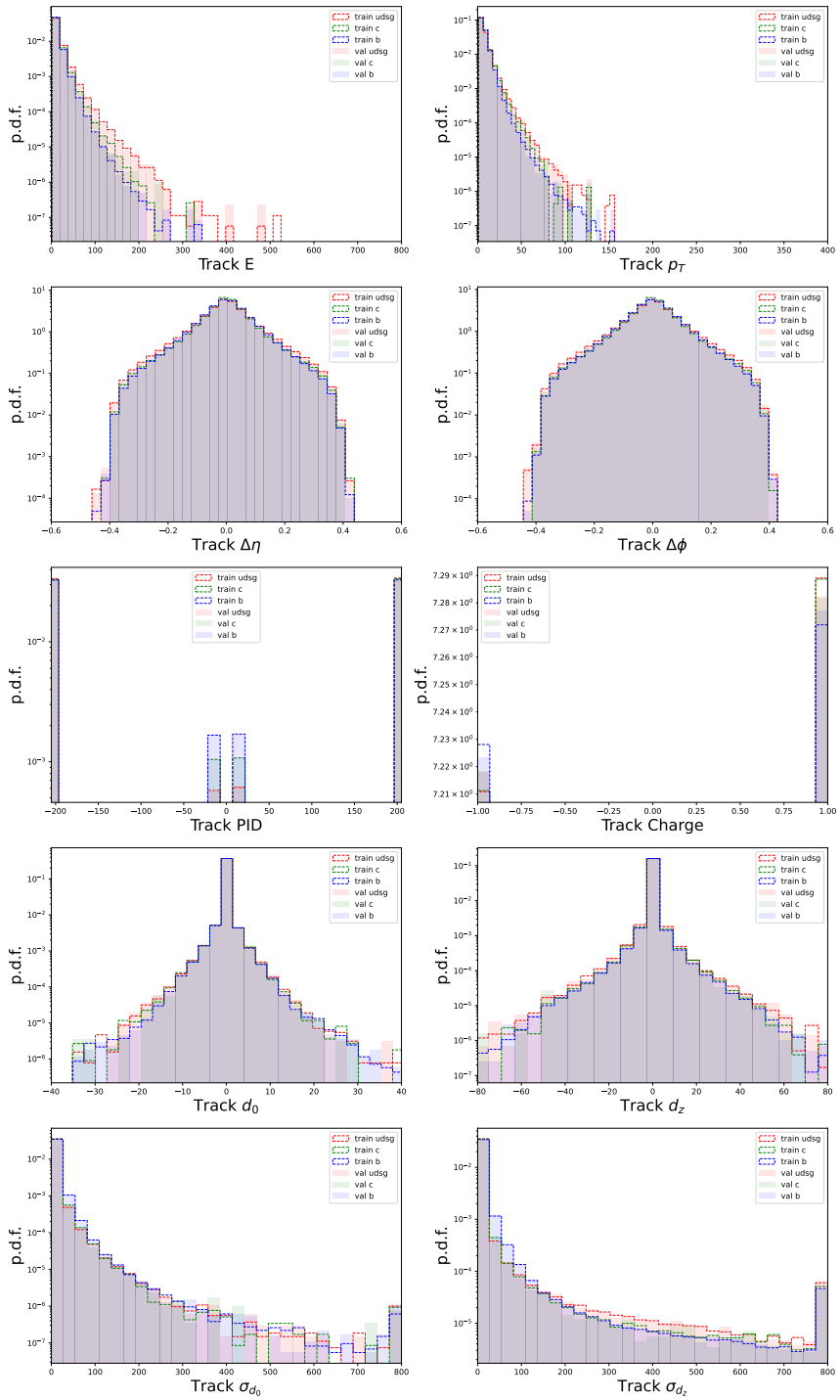


Figure 2.13: *Density histograms of the SV parameters of the first SV extracted from the datasets for three classes. Both training and validation sets are provided.*

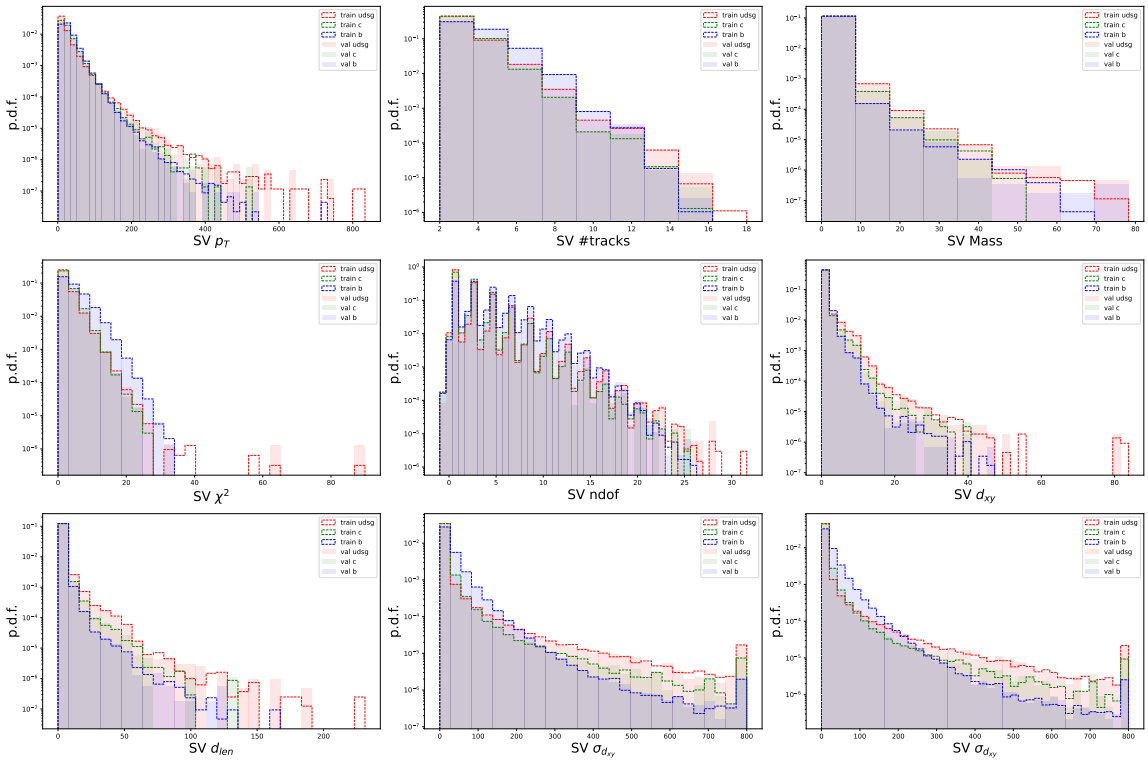


Figure 2.14: *Density histograms of the SV parameters of the second SV extracted from the datasets for three classes. Both training and validation sets are provided.*

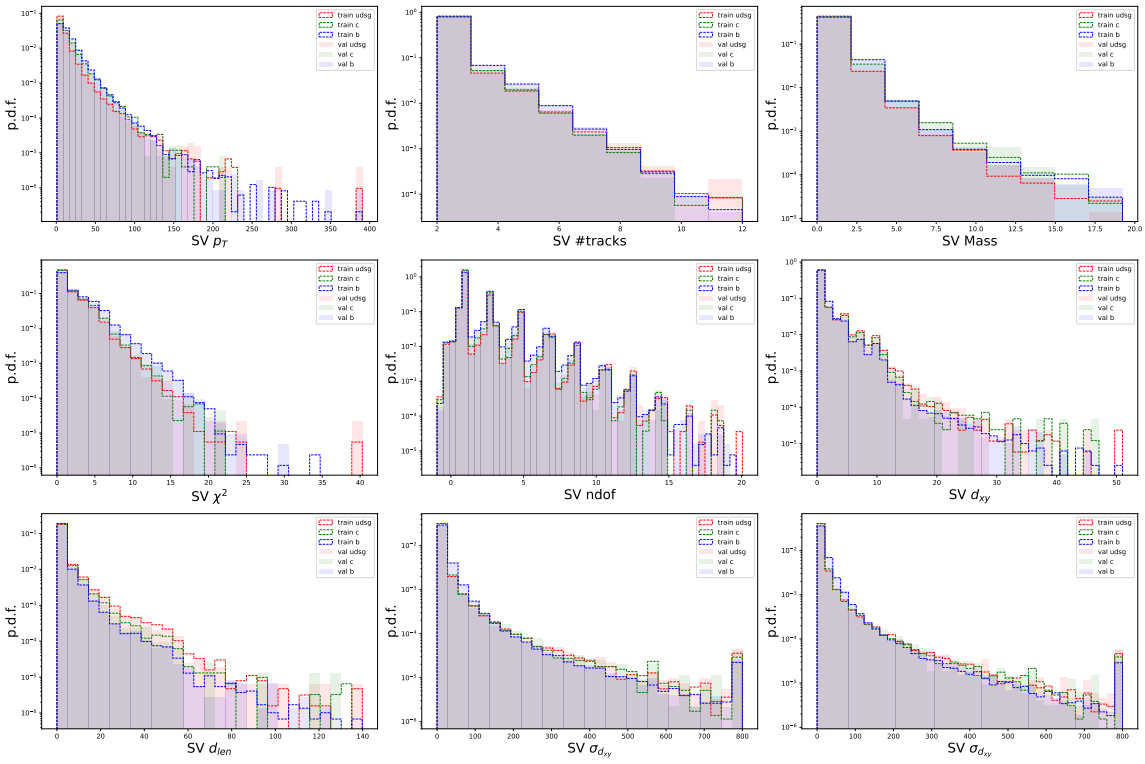
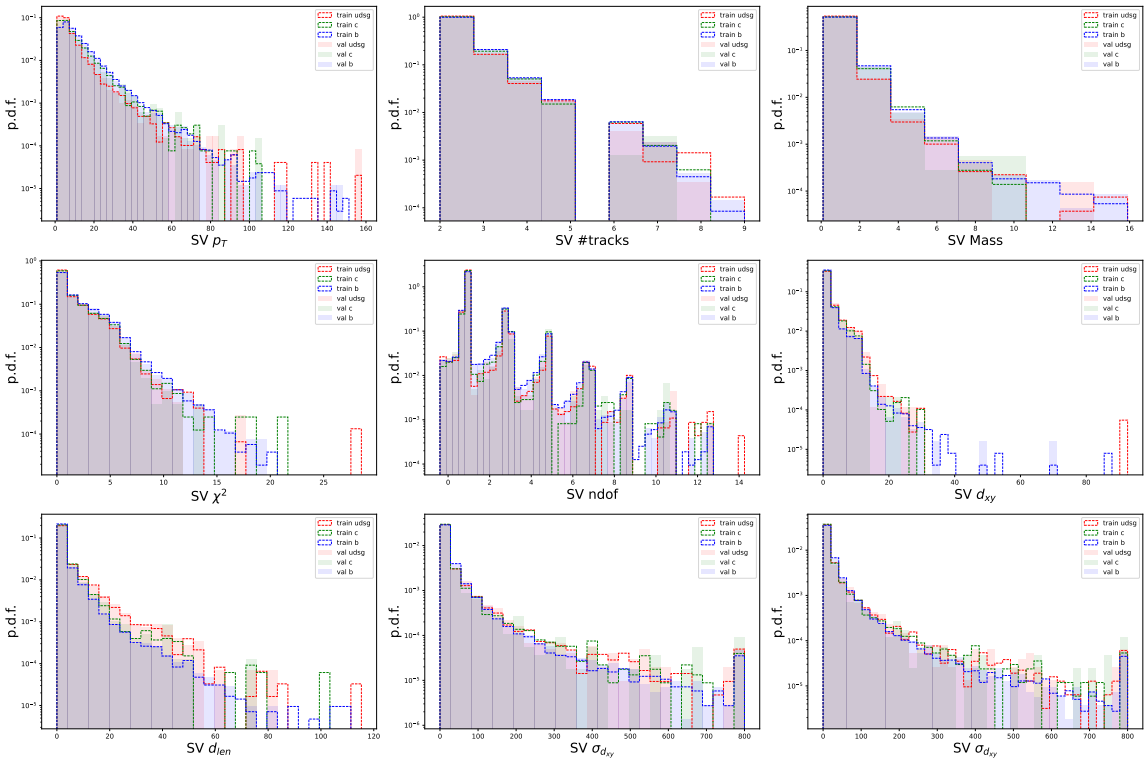


Figure 2.15: *Density histograms of the SV parameters of the third SV extracted from the datasets for three classes. Both training and validation sets are provided.*



3. MACHINE LEARNING

This chapter will briefly introduce ML, which is essential to appreciate the work of this thesis, and also will give the details of the models used. In Section 3.1, a brief introduction to the essential concepts of ML will be given as to why this kind of algorithm leads to the learning of the models. In the following Section 3.2, the concept of Deep Neural Networks (DNN) will be introduced; this part is also essential to understand the more complex DNN models that follow. In Section 3.3 the Extreme Gradient Boosting algorithm will be introduced. Then, in Section 3.4, the attention mechanism and the Transformer architecture will be explained. This section will prepare us for the following Section 3.5 on the retention mechanism, which is heavily inspired by the attention mechanism; in this section, we will also introduce the JetRetNet architecture specifically designed for this task. Finally, in 3.6, the state-of-the-art models for b-jet tagging will be explained.

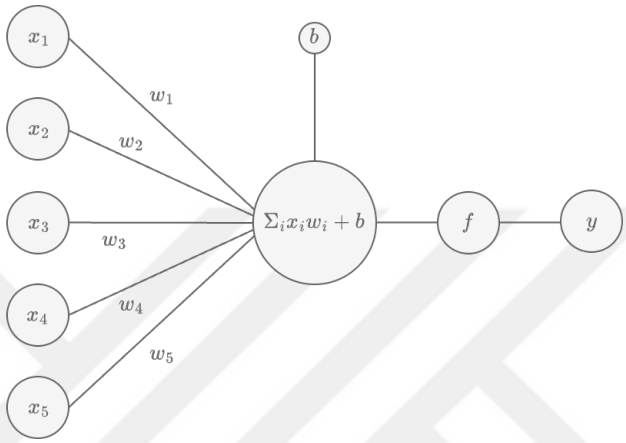
3.1 Introduction to Machine Learning

ML is a subset of Artificial Intelligence that relies on the idea that the model can update its parameters solely using data without any outside interference or set of defined rules. There are three learning types: supervised, semi-supervised, and unsupervised. The latter two will not be mentioned as they are not used in this study.

In supervised learning, we have a set of features, x, and labels, y, for each data. The features are given to the model with randomly initialized parameters, $\mathbf{w}$. This model predicts an output, $\hat{y}$, in the following way for a single node: Each input feature has a corresponding weight. The weighted sum of features is calculated, and then the bias term is added. This sum is given to an activation function, such as a sigmoid function, rectified linear unit (Rectified Linear Unit (ReLU)), gaussian error linear unit (Gaussian

Error Linear Unit (GeLU))... The purpose of these activation functions is to introduce some non-linearity to the output of a node. This process is represented in Figure 3.1 for a single node.

Figure 3.1: *Schematics of a single node, single layer network, where x_is are the inputs, w_is are the learnable weights, b is the bias, and f is the activation function.*



This output is then compared with the label using an error function, $E(y, \hat{y}|\mathbf{w})$, such as Mean Squared Error (MSE) shown below in Equation 3.1.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.1}$$

As the next step, the model takes this calculated error and updates its parameters with a predetermined step size called learning rate, η. The update of parameters is done using a back-propagating error correction algorithm, shortly called back-propagation, and gradient descent, an optimization algorithm. This is the step where the learning happens and also the step that makes ML such a powerful tool. Let's further understand both algorithms. Gradient descent is a procedure to minimize the error with a given set of parameters. $E(\mathbf{w})$ is a differentiable function, so it has a gradient vector:

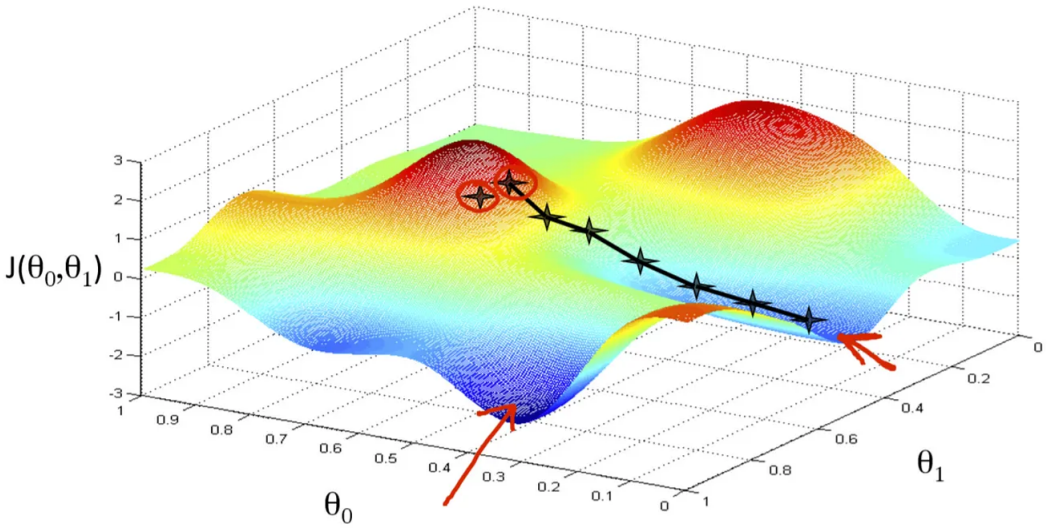
$$\nabla_{\mathbf{w}} E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_{\mathrm{d}}}\right]^T \tag{3.2}$$

Although this is the simplest case, the actual calculations of this gradient vector are done using a back-propagation algorithm. Which relies heavily on the usage of the chain rule across different layers. After this calculation, the weights are updated in the following way in Equation 3.3.

$$\Delta w_{\mathrm{i}} = -\eta \frac{\partial E}{\partial w_{\mathrm{i}}}, \qquad w_{\mathrm{i}} = w_{\mathrm{i}} + \Delta w_{\mathrm{i}}, \quad \forall \mathrm{i} \tag{3.3}$$

Consider an Error curve on the hyper-dimensional parameter space. We take the partial derivative of this Error function along each parameter, giving us the slope in that direction. Because of the minus sign in Equation 3.2, the weights are updated to a place where there is a lower error. As shown in Figure 3.2, it possibly ends up at a local minimum or at the global minimum.

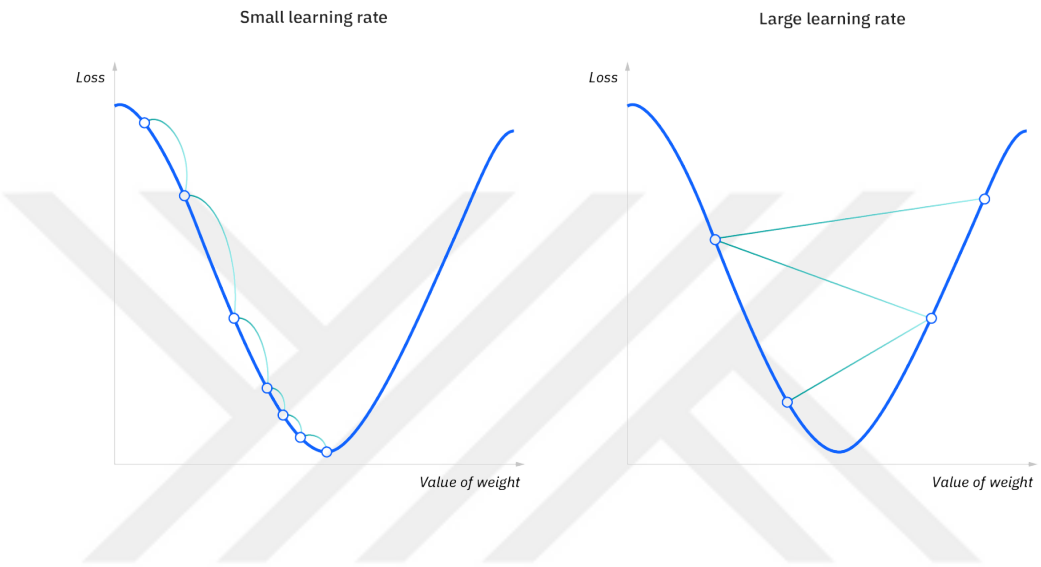
Figure 3.2: *Gradient Descent algorithm shown on the error space. The starting point is randomly initialized, and through training, the model converges to a local minima.[30]*



The effect of the learning rate is also shown in Figure 3.3. With a small step size model, finding a minima takes a very long time. Meanwhile, with a large step size, the

model's predictions "jump over" the minima, never reaching it. The success of the model depends on the proper adjustment of this hyper-parameter.

Figure 3.3: *Effect of learning rate, η, is shown in gradient descent algorithm. On the right side, the learning rate is tuned, and the model can indeed reach the minima. On the left plot, the learning rate is too high, and the model will skip the local minima.[30]*



Initially, the predictions are random, but after many iterations, the model begins to "learn" the relationship between the features and the labels. The model can now guess the labels of unseen data. This is why initially splitting the data into training and validation sets is important. The validation set helps us see the model's true performance. It is especially helpful in identifying whether the model is really learning or just memorizing the dataset.

As b-tagging is a two-class classification problem, we will use a single node in the last layer in all models. For binary classification, one node is enough with the output passed through the sigmoid function given as in Equation 3.4.

$$sigmoid = \frac{1}{1 + e^{-z}} \tag{3.4}$$

This function maps the real axis to the range [0,1], or in other words, to probabilities. This will be the output of our models for the classification task. By extending the logic in this section, one can reach the concept of deep neural networks.

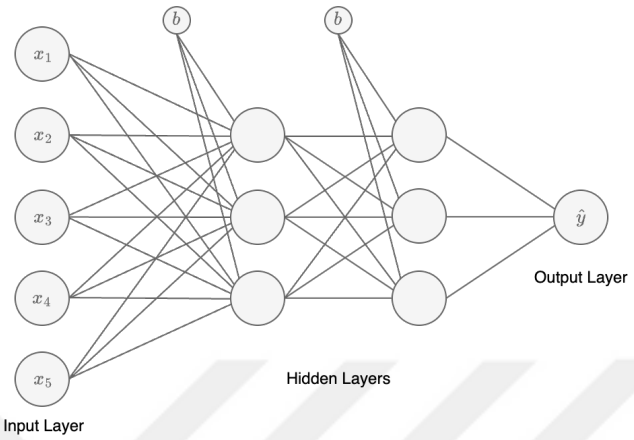
3.2 Deep Neural Networks

The node that we have seen in the previous section is also called a neuron. In DNN, there are many neurons for multiple layers. This depth allows the model to learn more complex behaviors than simple ML models. DNN aims to mimic the behavior of the human brain, which is why the nodes are also called neurons. In the first few layers, the model extracts the low-level information. However, as the depth increases, the model extracts some high-level information; this kind of information is very hard to get with traditional methods.

The simplest model of this kind is a Multi-Layer Perceptron (MLP), and a two-layer MLP is shown in Figure 3.4. Some of the important hyper-parameters of an MLP are the number of hidden layers and the number of nodes in each layer. Using unnecessarily complex models causes the model to over-fit the data. In other words, the model memorizes the training dataset.

The idea is similar to the single node case, but Equations 3.2 and 3.3 get more complex. Firstly, the parameter vector $\mathbf{w}$ becomes a matrix, one dimension representing the input index and the other representing the node index. For simplicity, we will consider an MLP with a single hidden layer. Call the weight matrix representing the relation between the input layer and hidden layer $\mathbf{w}$, and call the matrix representing the relation between the hidden layer and the output layer $\mathbf{z}$. This time, the error function is not directly dependent on $\mathbf{w}$ but rather dependent on $\mathbf{z}$. Using the chain rule, we can rewrite Equation 3.5 for an MLP as:

Figure 3.4: *A diagram of a 2-layered Multi-Layer Perceptron with a variable number of neurons on a variable number of hidden layers. The weights connect each input to each neuron and each neuron to the neurons in the next layer.*



$$\frac{\partial E}{\partial w_{\mathrm{hj}}} = \frac{\partial E}{\partial y_{\mathrm{i}}} \frac{\partial y_{\mathrm{i}}}{\partial z_{\mathrm{h}}} \frac{\partial z_{\mathrm{h}}}{\partial w_{\mathrm{hj}}} \tag{3.5}$$

As we increase the number of layers of an MLP, we get a DNN. This thesis will use DNNs to provide a baseline for more complex models.

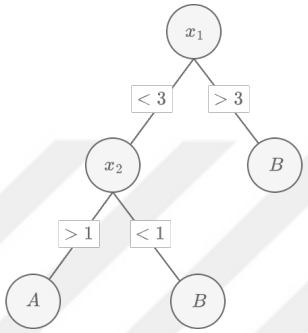
3.3 Extreme Gradient Boosting (XGB)

XGB is a subset of boosted decision trees, which is a subset of ensemble learning models[31]. In ensemble learning, instead of having a single complex model with high performance, we have many small models with low performance (weak models). Then, averaging or voting is done among the weak models; this operation is called bagging. This resembles Democracy, where you trust the wisdom of a not-so-educated crowd. XGB has proven its success in many tasks by giving state-of-the-art results. In the domain of b-tagging, it has also been the dominant model for many years. That is why it is important to include XGB as another baseline.

XGB is a complex model that requires an understanding of some concepts. Decision trees are a non-parametric learning method for classification and regression tasks. A

tree consists of nodes and branches, with the final node being called the leaf node. On the leaf nodes, we have labels of classes, the internal nodes denote features, and the branches denote the rules to be satisfied. Figure 3.5 shows an example of a decision tree. In XGB, what we call weak models correspond to shallow decision trees.

Figure 3.5: *A shallow decision tree with classes A and B. The inputs are checked if they satisfy certain conditions, and the final decision is made iteratively.*



One of the hyper-parameters of XGB is the trees' depth, which can affect both the training speed and the potential of over-fitting arising with increased depth. In ensemble learning, as the logic says, each individual model should indeed be weak. Simultaneously training these models and taking either a weighted average or via another meta-learner, the final result is reached. Here, the meta-learner is just another simple model, usually a Feed Forward Network (FFN), that averages the outputs of the trees.

Although the idea is simple, XGB has been the superior model in many tasks, especially in tabular data. Hence, to feed our jets to XGB, which is not directly tabular data, we have to flatten all the arrays and concatenate them. Since XGB is unable to capture the nature of the data, there should be a penalty in the performance. This might also indicate that the format of the data contains some information.

3.4 Transformers

The transformer is another class of deep-learning model that is based on the Multi-Head Attention (MHA) mechanism [32]. MHA is also constructed upon parallel instances of Scaled Dot-Product Attention. Both attention mechanisms are shown in Figure 3.6. As seen in the figure, the inputs to these mechanisms are not plain features, as in previous cases. This time, the inputs are vectors called Queries (Q), Keys (K) of dimension d_k, and Values (V) of dimension d_v. These vectors are linear projections of embedding **x**. In practice, the computation is done simultaneously on a set of queries, therefore Q, K, and V are matrices. The Scaled Dot-Product Attention output matrix is computed as follows in Equation 3.6.

$$Attention(Q, K, V) = softmax\Big(\frac{QK^T}{\sqrt{d_{\mathrm{k}}}}\Big)V \tag{3.6}$$

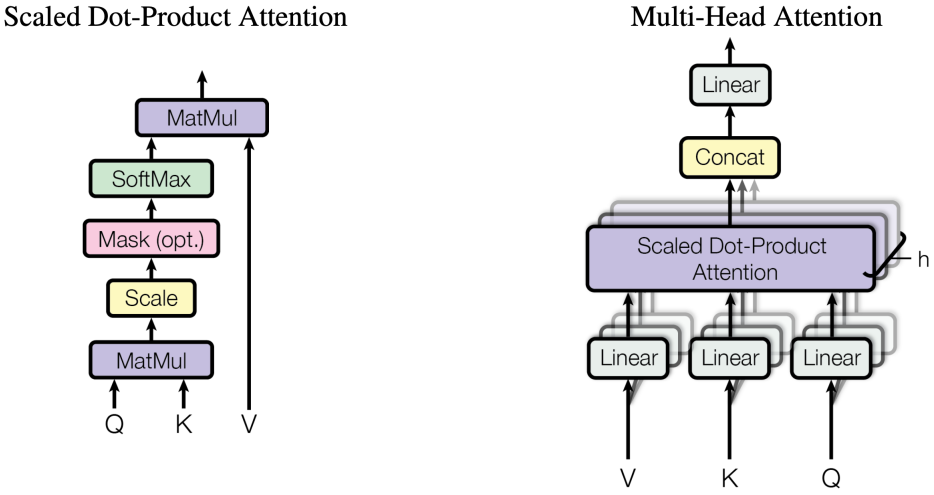
The dot-product, QK^T, can be interpreted as a similarity measure since it is a projection. The scaling is done via division by the square root of the dimension of Q and K vectors. This scaled matrix is passed through a row-wise softmax function to get weights. Finally, this attention matrix is multiplied with vector V to get the output. This calculation is also shown on the left side of Figure 3.6.

In Multi-Head Attention, multiple Scaled Dot-Product Attention blocks are run in parallel. MHA is calculated as follows:

$$\begin{aligned} MultiHeadAttention(Q, K, V) &= Concat(head_1, ..., head_{\mathrm{h}})W^O \\ head_i &= Attention(QW_{\mathrm{i}}^Q, KW_{\mathrm{i}}^K, VW_{\mathrm{i}}^V) \end{aligned} \tag{3.7}$$

Here $W_{\mathrm{i}}^Q, W_{\mathrm{i}}^K, W_{\mathrm{i}}^V$, and W_O are the parameter matrices of the projections, which are shown on the right side of Figure 3.6 as the single linear layers. Since there is only one layer and not more, non-linearity is not in the game. And therefore, we get linear

Figure 3.6: *Scaled Dot-Product Attention on the left, Multi-Head Attention mechanism on the right. Scaled Dot-Product attention given in Equation 3.6 is parallelized with multiple heads and linear layers for embedding, finally concatenating the outputs and passing through a final linear layer before inference. [32]*



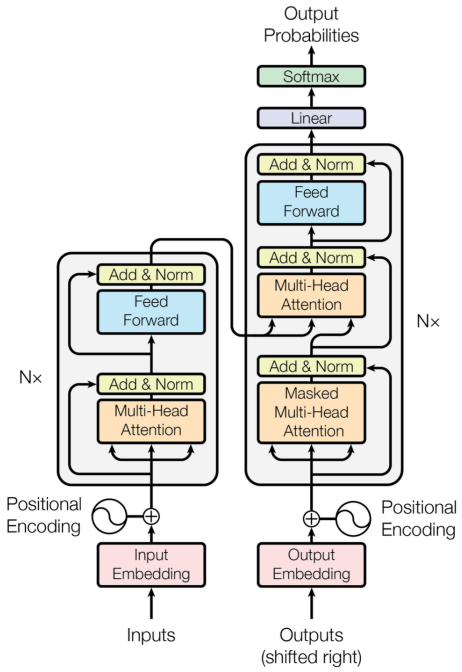
projections $QW_{\text{i}}^{Q}, KW_{\text{i}}^{K}, VW_{\text{i}}^{V}$. The power of MHA comes from parallelism in terms of computation speed and conceptuality. Each parallel instance of a Scaled Dot-Product Attention represents another feature being extracted from the data.

This MHA block is used multiple times in an encoder-decoder architecture, feed-forward networks, and residual connections. The inputs are first passed through an embedding and positional encoder layer. This embedding layer is also called the Tokenizer for large-language models. After that, we have the encoder and decoder blocks, as shown in Figure 3.7. In the encoder block, Q, K, V matrices are fed into the MHA. Then, a residual connection is made to the output as follows:

$$\mathcal{F}(x) = \mathcal{F}(x) + x \tag{3.8}$$

Where $\mathcal{F}(x)$ is the output of a network and x is the input. This connection is made and normalized at the "Add & Norm" layer in Figure 3.7. This output is then given to a feed-forward network with a residual connection.

Figure 3.7: *"Transformer architecture consisting of encoder and decoder parts. Both parts use MHA modules with FFN layers, after all operations the outputs are added and normalized. Multiple skip connections also increase the flow of information."[32]*



The block on the left side of Figure 3.7 is called the encoder block. These encoder blocks are run in series for N blocks. The output of the last encoder block is given as input to a decoder block, similar in architecture. After passing through a series of MHA, addition & normalization, and Feed-forward layers, the output of the decoder is given to a single linear layer, then passed through a softmax function (or sigmoid if the task is binary classification) to get the prediction probabilities.

3.5 Retentive Networks

Retentive Networks are a type of architecture that uses the Retention mechanism [33]. This mechanism takes its inspiration from attention, which was mentioned in the previous section 3.4. As the name suggests, the retention mechanism puts emphasis on forgetting the information. This idea is similar to the Recurrent Neural Networks (RNN),

but the model has no explicit recurrence loop. Instead, retention is applied by introducing some decay constants and causal masking to the attention mechanism. Here, just as the attention mechanism, there is also the query (Q), key (K), and value (V) matrices, with slight changes. This results in two different representations of the exact mechanism: parallel and recurrent representations. There is also the third one, chunk-wise representation, which is the combination of both. Since the best-performing model used the parallel representation, we will not go into the details of the other two. The inputs to the retention mechanism can be can be summarized in Equation 3.9.

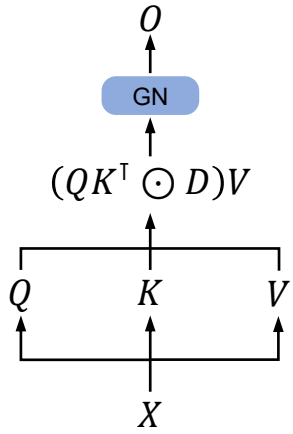
$$
\begin{aligned}
&Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \overline{\Theta}, \quad V = XW_V, \\
&\Theta_{\mathrm{n}} = e^{in\theta}, \quad D_{\mathrm{nm}} = \begin{cases} \gamma^{\mathrm{n-m}}, & \mathrm{n} \geq \mathrm{m}, \\ 0, & \mathrm{n} < \mathrm{m}, \end{cases}
\end{aligned}
\tag{3.9}
$$

Here, W_Q, W_K, and W_V matrices are learnable parameters, and the D matrix is used to introduce the causal masking, combined with the Θ and its complex conjugate $\overline{\Theta}$ exponential decay is also applied, where the name retention comes in. With these altered definitions of the Q, K, and V matrices, we can define retention as in Equation 3.10.

$$
\text{Retention}(X) = (QK^\top \odot D)V \tag{3.10}
$$

Combined with the group norm, we get the parallel representation of retention, represented in Figure 3.8. Just as MHA modules are used for the attention mechanism, a similar module exists in retention: Multi-Scale Retention (MSR). This module parallelizes the model by introducing both multi-heads and multi-scale across heads.

Figure 3.8: *A simplified version of the parallel representation of the Retention mechanism. Q,K, and V matrices are calculated with the given input X and the learnable parameters. Then 3.10 is applied, followed by a group normalization operation to get the final output.[33]*



$$
\begin{aligned}
\gamma &= 1 - 2^{-5-\text{arange}(0,\text{h})} \in \mathbb{R}^{\text{h}}, \\
\text{head}_{\text{i}} &= \text{Retention}(X, \gamma_{\text{i}}), \\
Y &= \text{GroupNorm}_{\text{h}}\big(\text{Concat}(\text{head}_1, \ldots, \text{head}_{\text{h}})\big), \\
\text{MSR}(X) &= \big(\text{swish}(XW_G) \odot Y\big)W_O
\end{aligned}
\tag{3.11}
$$

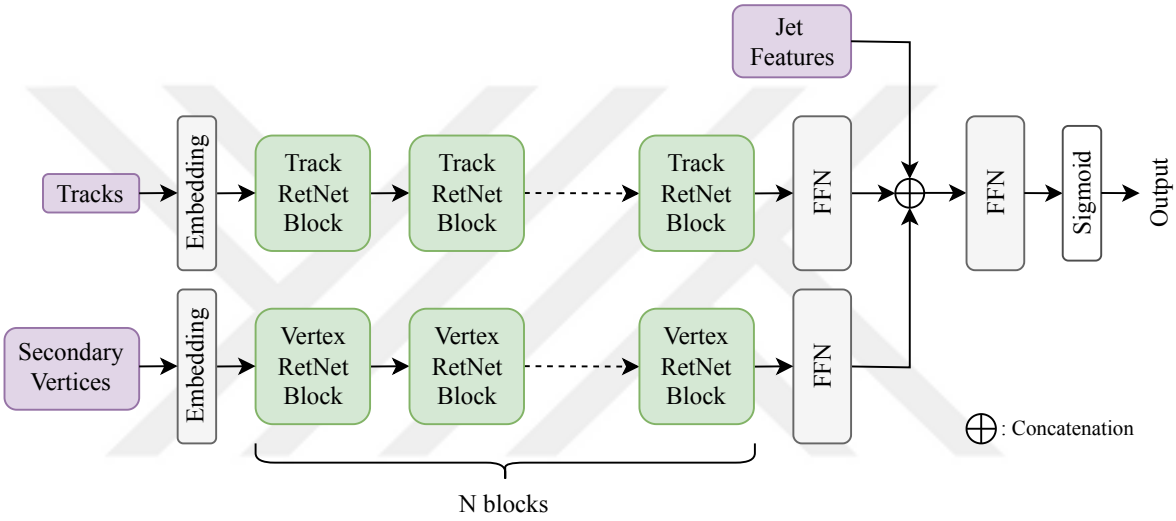
In Equation 3.11 $W_G, W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ are learnable parameters, and X is the input. The outputs from the multi-scaled heads are concatenated and group-normalized. The swish gate is used to increase the non-linearity of the retention outputs. The final output is reached with the addition of new parameters, W_G and W_O.

3.5.1 JetRetNet

Using the MSR blocks, we have constructed an architecture, JetRetNet [34], that is specified for our dataset and the task of b-tagging. The JetRetNet architecture is designed to handle the track and SV features in parallel, as shown in Figure. 3.9. Both inputs are first through a single embedding layer, and then a number of RetNet blocks are applied. A

simple FFN is used to reshape the output of RetNet sequences. Then, these outputs from track and SV pathways are concatenated with the jet features and passed through a final FFN and a sigmoid function to get the final model prediction.

Figure 3.9: *JetRetNet architecture consisting of RetNet blocks and Feed-forward networks (FFNs). Tracks and SVs are processed in parallel by passing through an embedding layer, followed by a series of RetNet blocks and an FFN. The outputs are concatenated with the jet features, and the final FFN with the sigmoid function makes the final decision. [34]*



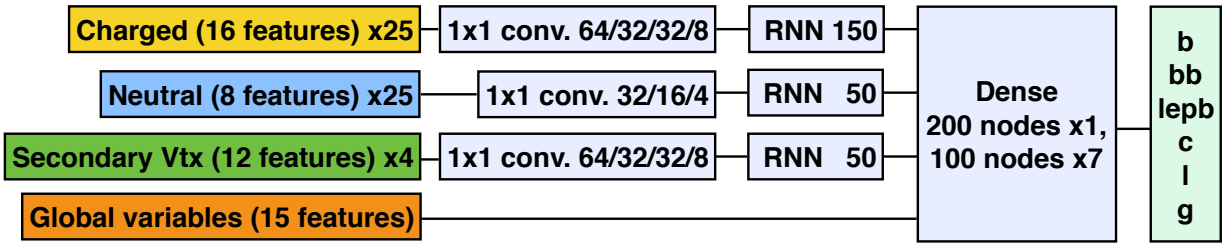
3.6 State-of-the-art models for B-Jet Tagging

3.6.1 DeepJet

DeepJet is one of the first sophisticated Deep Learning (DL) models of the CMS Collaboration for heavy flavor tagging [35]. It utilizes the Convolutional Neural Networks (CNN) and RNN, given in Figure 3.10. The type of input is a bit different compared to what we and the other state-of-the-art models use. Here, the objects are separated into charged and neutral particle flow candidates as well as the secondary vertices. Some global jet features are also included in the final decision after the CNN and RNN layers.

DeepJet uses slightly more than 650 input features.

Figure 3.10: *The architecture of the DeepJet model, with inputs in 4 domains: charged PF, Neutral PF, Secondary Vertex, and Global variables. These inputs, except global variables, are passed through CNN and RNN layers, finally concatenated, and given to a dense network to get the output.[35]*



For example, the first 25 charged particle flow candidates each have 16 features, ordered according to decreasing displacement significance. This input is passed through a CNN with a kernel size of 1x1. This kernel size is preferred as all the objects are treated in the same manner, as the physics suggests. Then, the output from the CNN layers is given to Long Short Term Memory (LSTM) modules [36]. This is done for the other two input types, Neutral candidates and the SVs, and then all outputs from the LSTM modules and the global variables are given into a DNN to get the final prediction of the model.

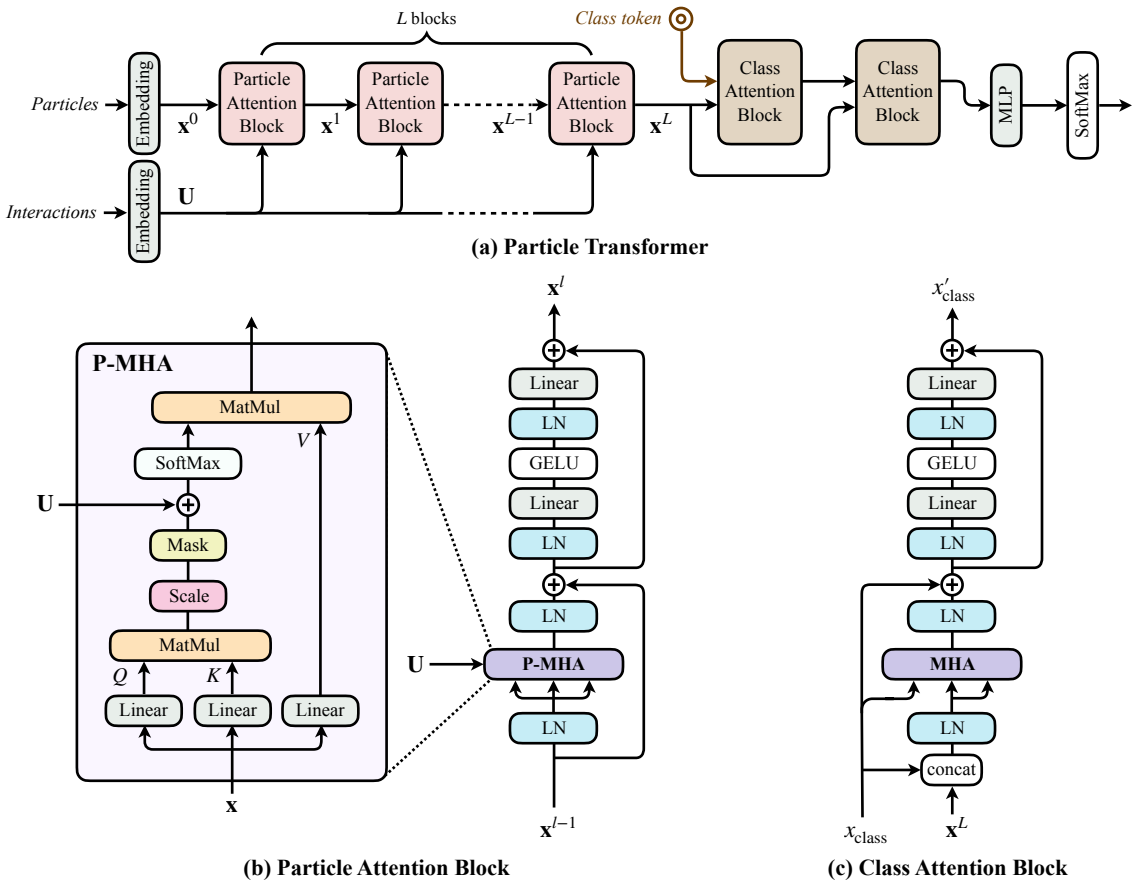
The DeepJet model demonstrates excellent performance; however, the large number of input features poses a significant challenge to inference time and memory efficiency. This high dimensionality slows down inference and increases memory usage, which can be a limitation in practical applications

3.6.2 Particle Transformer (ParT)

As the name suggests, the Particle Transformer [37] is the primary transformer model for particle physics. First published as an event-based jet tagger, it later evolved to perform many other tasks, including b-tagging. The ParT model is best understood visually, architecture given in Figure 3.11. Here, in part a, we have encoder and decoder parts

as in most transformer architectures, with constituents particle and class attention blocks, respectively. The main difference between these two kinds of blocks is the addition of the interaction matrix, **U**, which is, in a way, the dot product of the input column vector with its transpose, giving us a matrix.

Figure 3.11: *The architecture of (a) Particle Transformer (b) Particle Attention Block (c) Class Attention Block. Similar to a regular transformer, the encoder-decoder architecture is seen. P-MHA is also the same as MHA except for the U matrix. [37]*



In this model, the input is separated into three classes of features: kinematics $\big(\Delta\eta$, $\Delta\phi$, $\log(p_{\mathrm{T}})$, $\log E$, the logarithm of the relative p_{T} and energy of the tracks to the jet.$\big)$, particle identification (charge, electron, muon, photon, charged hadron, neutral hadron), and trajectory displacement $\big(\tanh(d_0)$, $\tanh(d_{\mathrm{z}})$, σ_{d_0}, σ_{d_z} $\big)$.

Once more confirming the hype about the transformers, ParT also demonstrates state-of-the-art performance on many tasks, including b-jet tagging. We will use ParT to compare its performance with that of our models.

4. RESULTS AND DISCUSSION

During the training of the models, we used the Pytorch library [38] for most of the models and the XGB library [31] for the training of the XGB. Except for the XGB model, the trainings are done on NVIDIA Quadro RTX 5000 Graphics processing unit (GPU).

In this chapter, we will first introduce the performance metrics and tools that will be used to evaluate the models in Section 4.1. Then, the model training results will be discussed in Section 4.2. In this section, we will compare the performances of the three models: MLP, XGM, and JetRetNet. Later in Section 4.3, we will compare the XGB and JetRetNet models with state-of-the-art models like DeepJet and ParT.

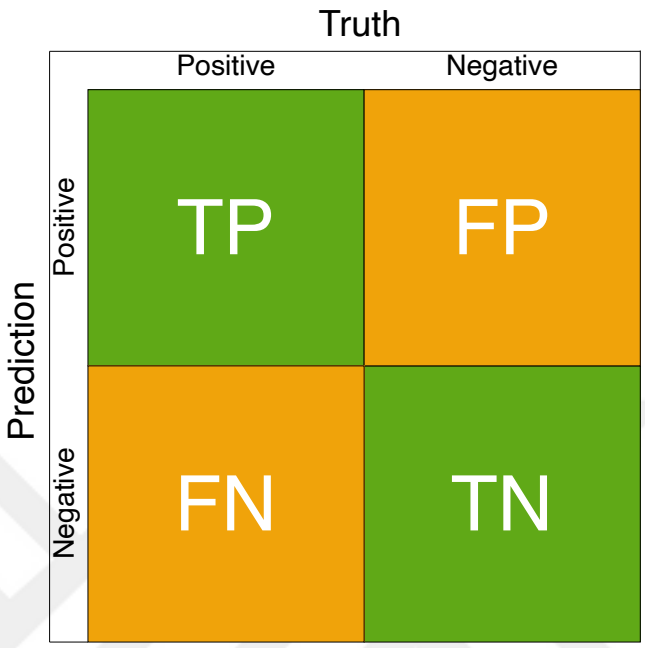
4.1 Evaluation Metrics and Performance Measures

To evaluate the performance of the models, we need to decide on some metrics that give a sense of the models' capabilities.

Many of the metrics commonly used in the literature use the confusion matrix. This matrix compares the actual truth values with the predicted values. Many metrics have been derived from this matrix, some of which we will also use in this thesis. The confusion matrix is best understood visually in Figure 4.1. Here, the matrix is divided into four groups for a binary classification task: True Positive (TP) for the values that are positive and also predicted as positive, True Negative (TN) for the values that are negative and also predicted as negative. These two classes are where the model performs successfully. The actual value is negative for False Positive (FP), but the model's guess is positive, hence the name false positive. Similarly, the truth is positive in False Negative (FN), but the model predicts it as negative.

In an actual confusion matrix, the numbers corresponding to the symbols TP, TN, FP, and FN are used instead. Although this matrix gives much information about the

Figure 4.1: *The confusion matrix, where TP, FP, FN, and TN are true positive, false positive, false negative, and true negative, respectively.*



model's performance, it is still a matrix and does not have a single number that can be used to compare models. Luckily, we are not the first ones to try a classification task. Therefore, people before us have invented many parameters that are now commonly used in the literature.

4.1.1 Accuracy

The most common metric that is used in classification tasks is accuracy, which is defined as follows in Equation 4.1.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

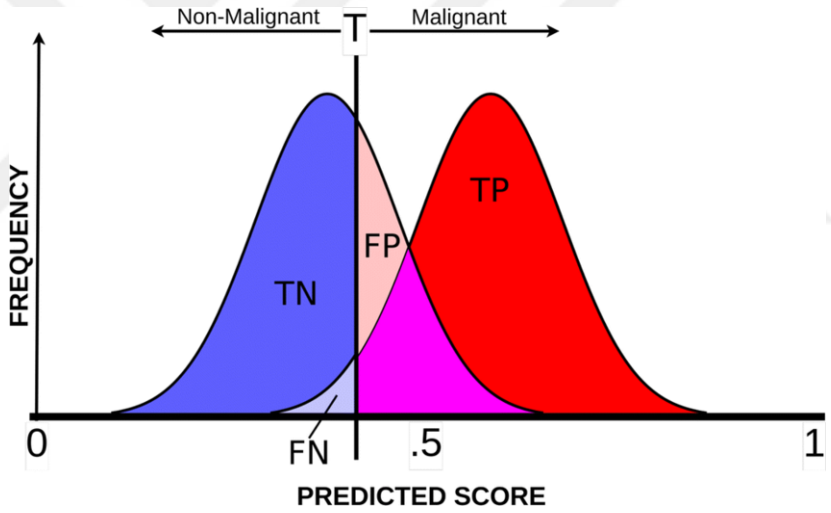
This is the ratio of correctly guessed data points to all data. Although this metric has been widely used historically, it does not contain much information about the misidentified items. It is also highly dependent on class imbalance; it is only valid when both classes have almost the same number of samples. We will monitor its evaluation

during training but not use it as a decisive metric for the model's performance.

4.1.2 Area Under Curve (AUC) Score

The next metric we will use is the Area Under the Curve (AUC), specifically the area under the receiver operating characteristics (Receiver-operating characteristic (ROC)) curve. This is one of the most essential metrics in classification tasks and needs to be adequately understood. First, Let's understand the ROC curve derived from another plot. Figure 4.2 gives the probability distribution for a binary classification task.

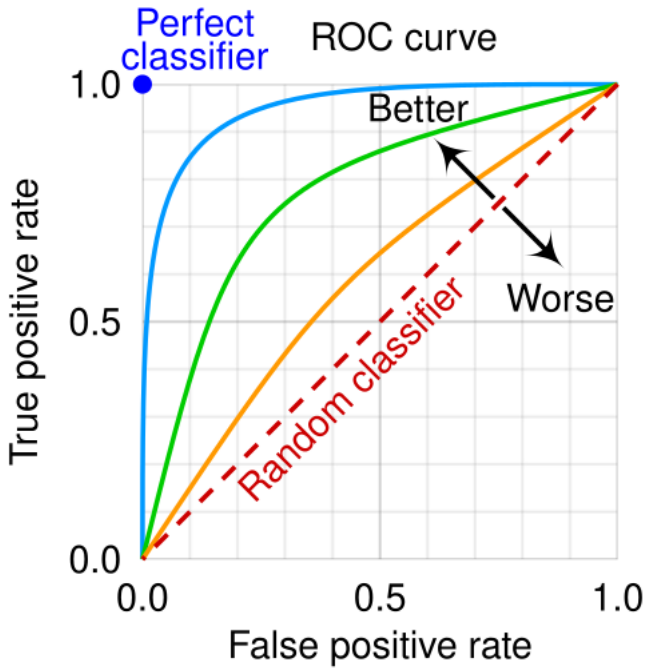
Figure 4.2: *"Prediction probability histogram of a binary problem, where TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively. T is the threshold between what is classified as class 0 and class 1."[39]*



We get such a plot after training the models for the binary classification task. But this plot raises a new question: Where should we put a threshold to discriminate between classes for the best efficiency? This question leads to the ROC curve as follows: By sliding the threshold, T in Figure 4.2, across the model's predicted score axis, one can calculate the true positive rate (True Positive Rate (TPR)) and false positive rate (False Positive Rate (FPR)) defined in Equation 4.2

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \quad \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

Figure 4.3: *"Receiver Operating Characteristic (ROC) curve with False Positive Rate and True Positive Rate. A diagonal shows the performance of a random classifier. Three curved lines from (0,0) to (1,1) that get progressively closer to (0,1) show improving classifiers." [40].*



We get the ROC curve by calculating these two metrics for all T values, given in Figure 4.3. Calculating the area under this curve finally gives us the AUC score, which ranges between 0 and 1. However, in practice, an AUC score of 0.5 means the model is guessing randomly; hence, AUC scores below 0.5 are irrelevant. We will also monitor this metric during training and use it as a decisive metric for the model's performance.

4.1.3 F1 Score

To understand the F1 score, we need to make more definitions: precision and recall given in Equation 4.3. Precision is the ratio of the correctly predicted positive values to all

the positive predictions of the model or the ratio of TP to the first row in Figure 4.1. Recall is a similar concept but from the perspective of the truth: it is the ratio of the correctly predicted positive values to all true positive values, or TP divided by the first column in Figure 4.1 .

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

With these two, we can then define the F1 score in Equation 4.4; this is the harmonic mean of recall and precision.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

The F1 score will also be monitored during training and used as a decisive metric.

4.1.4 Signal Efficiency vs. Misidentification Rate Plots

Although the AUC score gives us a nice metric extracted from the ROC curve, the ROC curve contains more information. We will also compare the model performances on a plot similar to the ROC curve. Actually, just the axes switched in Figure 4.3. Another name for TPR is signal efficiency, and similarly, FPR is also called a misidentification rate. Hence, we get the physicist's way of an ROC curve.

This plot defines the model's working points (Working Point (WP)) as fixed misidentification rates. We will use three WPs: loose, medium, and tight, 0.1, 0.01, and 0.001, respectively, which have also been used in other works [41]. The signal efficiencies will be compared at these fixed WPs for b-jet vs. light-jet and b-jet vs. c-jet cases.

4.2 Performance Evaluation and Analysis

Three different models — MLP, XGB, and Retentive Networks (RetNet) — have been tested for b-jet tagging. After extensive training, this section represents the three best-performing models. The evolution of the loss and the accuracy, AUC, and F1 score

metrics during the training are given in Figures 4.5, 4.4 for JetRetNet and MLP respectively. Here, the performance on the validation set is better in most of the epochs because of the dropout layers. As explained in Section 3.2, some of the weights are set to 0 during training, but in the validation set, the dropout is not in action. Hence, the original weights are used for the evaluation of the validation set, and the model performs better than the training set.

Figure 4.4: *The evolution of the loss and the accuracy, AUC, and F1 score metrics through epochs for the MLP model. For all plots, the values on the training set are given in blue, and the values for the validation set are given in orange.*

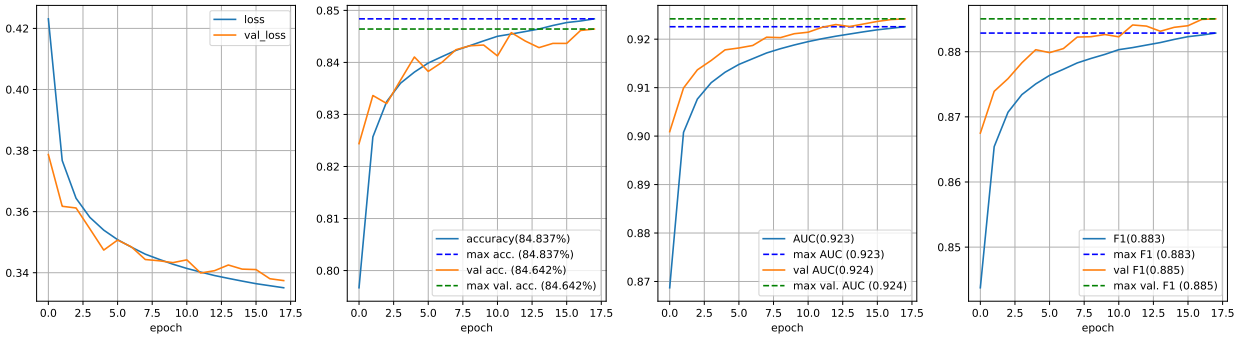
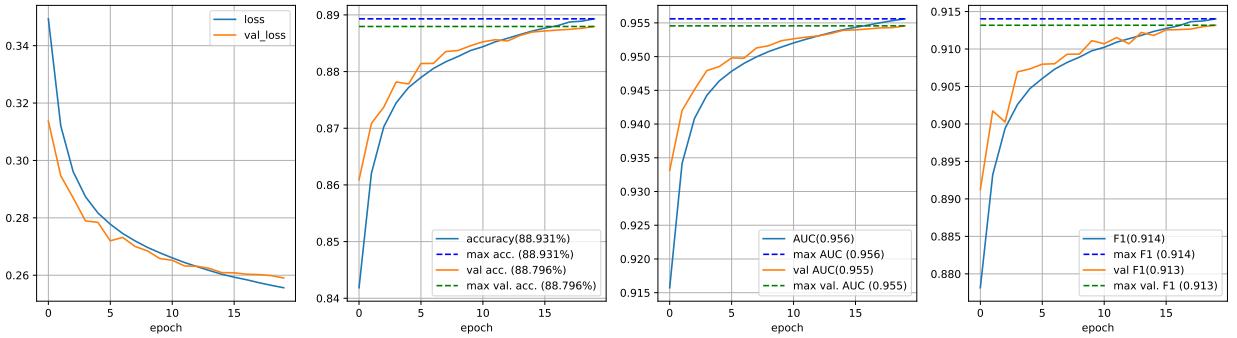


Figure 4.5: *The evolution of the loss and the accuracy, AUC, and F1 score metrics through epochs for the JetRetNet model. For all plots, the values on the training set are given in blue, and the values for the validation set are given in orange.*



Since XGB has its own training methods, we are unable to monitor its performance

through epochs; we can only get the metrics after training. The performances of the three models in the final epoch are summarized in Table 4.1. As can be seen, the JetRetNet model outperforms the other two by a significant margin. The Multi Layer Perceptron (MLP) does its job as it was to show that these novel models have some benefits compared to the simplest deep learning model. XGB is also very competitive but does not surpass JetRetNet.

Table 4.1: *Performance metrics on the validation set for MLP, XGB, and JetRetNet models in the final epoch.*

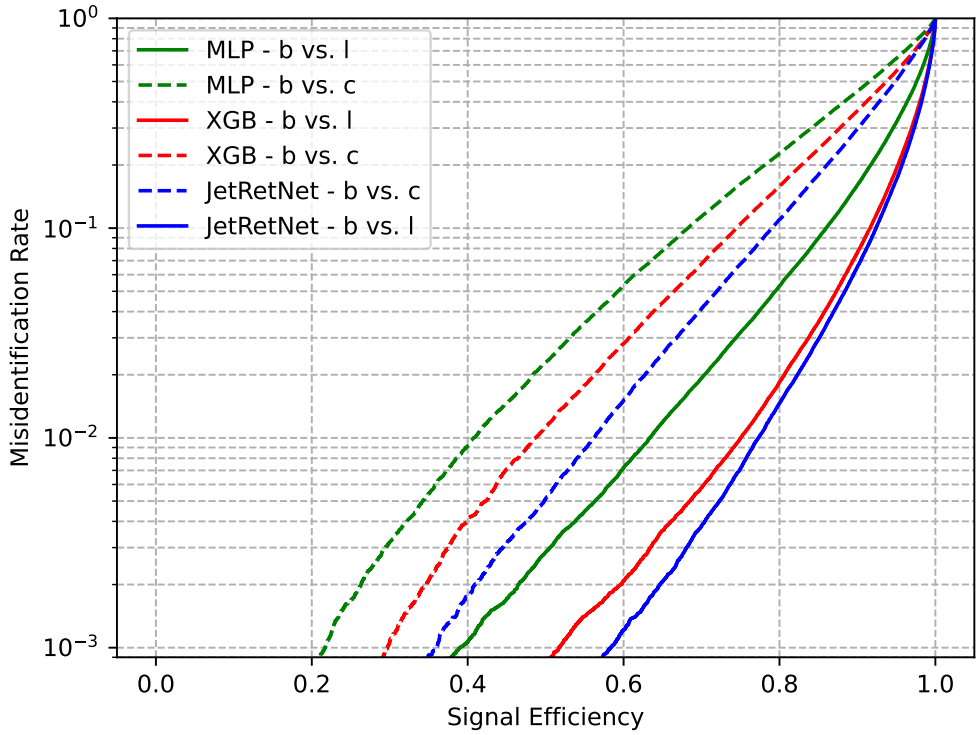
Metric	Accuracy	AUC	F1 Score
MLP	84.61%	0.924	0.885
XGB	87.83%	0.947	0.878
JetRetNet	**88.80%**	**0.955**	**0.913**

Looking at the signal efficiency vs. misidentification rate plots, given in Figure 4.6, gives a better idea of the performance. Here, we have separated the background of the dataset into two subsets: c-jets and light-jets (u, d, s quark jets or gluon jets). Again confirming Table 4.1: on all three WPs, JetRetNet performs the best, MLP gives a good baseline, and the XGB is in between.

It must be noted that we have trained a few XGB models with varying the depth of the trees. Although when the model has a depth of 13, it surpassed the JetRetNet model, it is also highly overfitting to the data. This over-fitting problem is a common problem in XGB models. One has to choose the parameters to minimize the over-fitting. Hence, we choose a tree depth of 11, which balances both over-fitting and the performance measure. If our dataset was very large and general, this issue might have been ignored, but with 1 million jets in the validation data, one cannot be sure of the power of the generalization of the dataset.

From this plot, we can also extract and compare the signal efficiencies of the three

Figure 4.6: *Signal efficiency vs. Misidentification rate of b-jets. Here, the dashed curves are for b-jet vs. c-jet, and the solid curves are for b-jet vs. light-jet classification. The three models, MLP, XGB, and JetRetNet, are shown in green, red, and blue, respectively.*



WP: loose, medium, and tight, given in Table 4.2 for the b-jet vs. light-jet efficiency and in Table 4.3 for b-jet vs. c-jet efficiencies. Again, JetRetNet's superior performance is demonstrated in all cases. Tight (0.001)

We can also compare the inference times of the networks. MLP and JetRetNet models are trained on the GPU, whereas XGB is trained on 36 Central Processing Unit (CPU) cores. Although this is not a proper comparison, we will compare the speed on the GPU for MLP and JetRetNet and the speed on a single-core CPU for XGB, given in Table 4.4. These values are calculated on the validation set of approximately one million jets. Commenting on the table, MLP is the fastest model for inference, but its performance is lacking. The latter is faster for XGB and JetRetNet, but again, this is not a fair comparison

Table 4.2: *Signal efficiencies for b-jet vs. light-jet tagging for three WP on the validation set for MLP, XGB, and JetRetNet models in the final epoch.*

WP (Mistag)	Loose (0.1)	Medium (0.01)	Tight (0.001)
MLP	0.860	0.634	0.390
XGB	0.916	0.750	0.516
JetRetNet	**0.923**	**0.773**	**0.584**

Table 4.3: *Signal efficiencies for b-jet vs. c-jet tagging for three WP on the validation set for MLP, XGB, and JetRetNet models in the final epoch.*

WP (Mistag)	Loose (0.1)	Medium (0.01)	Tight (0.001)
MLP	0.682	0.408	0.217
XGB	0.745	0.487	0.297
JetRetNet	**0.798**	**0.560**	**0.356**

since they work on different kinds of machines.

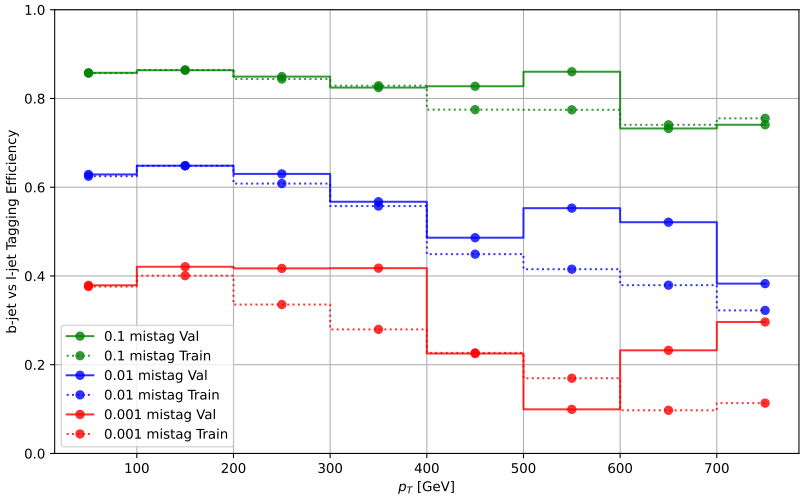
Table 4.4: *Inference times of the MLP XGB and JetRetNet models. Here, XGB ran on a single-core CPU, and the other two models ran on GPU.*

Model	Inference time per Jet (sec)
MLP	4.77×10^{-9}
XGB	1.88×10^{-5}
JetRetNet	8.57×10^{-6}

One can also look at the performances across the Jet p_{T} range. In Figures 4.7, 4.8 and 4.9, b-jet vs. light-jet signal efficiencies are given for all three WP. Performances on both the training and the validation sets are plotted to see the effect of over-fitting. The same plots for b-jet vs. c-jet signal efficiencies are also given in Figures 4.10, 4.11 and 4.12. Since the data is very sparse after jet p_{T} of 400 GeV, especially for b-jet vs. c-jet tagging, the left part of these plots will be more relevant.

Commenting on Figure 4.8, JetRetNet shows stable and good performance on the loose WP for all p_{T} scales with no overfitting. Overfitting starts to be visible for the medium and tight WPs after a p_{T} of 300 GeV. Since the data is sparse after 400 GeV, there are fluctuations. The performance of the XGB case in Figure 4.9 is close to JetRetNet. However, the overfitting is visible in all p_{T} ranges for medium and tight WPs. Although we have selected an XGB model with less overfitting than the more complex model, it still has much higher overfitting than the other two. JetRetNet has very little overfitting, mostly visible in the tight WP, but it is clearly visible in XGB plots for all WPs. This might mean that although XGB is very quick to train, it may not generalize well to unseen data.

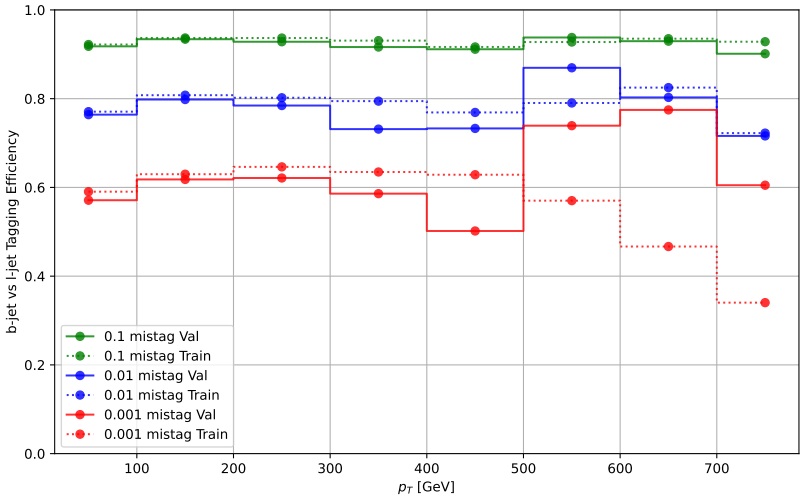
Figure 4.7: *Jet tagging efficiency for b-jet versus light-jet discrimination using the MLP model, as a function of jet p_{T}. Performance is evaluated for various mistag rates (0.1, 0.01, 0.001) on both training and validation datasets.*



4.3 Comparison with Existing Methods

We compare JetRetNet and XGB against state-of-the-art models such as DeepJet [35] and ParT [37] to assess our models' overall performance and competitiveness. These

Figure 4.8: *Jet tagging efficiency for b-jet versus light-jet discrimination using the JetRetNet model, as a function of jet p_{T}. Performance is evaluated for various mistag rates (0.1, 0.01, 0.001) on both training and validation datasets.*



models are widely recognized for their superior performance in b-jet tagging tasks and are considered benchmarks within the particle physics community. Both DeepJet and ParT employ advanced architectures tailored specifically for jet tagging. In Figure 4.13, the performances of XGB and JetRetNet models are compared with DeepJet and ParT. It must be noted that these two models are trained with 140 million jets; hence, this is not a fair comparison. This dataset provides both the richness of the jets and higher generalization power.

Despite that, at loose WP, both models perform reasonably well, even surpassing DeepJet for b-jet vc. c-jet tagging efficiency. As we come to the medium, WP DeepJet and JetRetNet perform almost the same for b-jet vc. c-jet case, while XGB is unable to keep up. This means XGB is not able to capture the details in the dataset for this threshold. The difference is much more evident in the tight WP, where our models are unable to compete with the state-of-the-art. In this tight WP, the statistics are much less than the other two, and the separation demands of the two classes are much higher.

Figure 4.9: *Jet tagging efficiency for b-jet versus light-jet discrimination using the XGB model, as a function of jet p_{T}. Performance is evaluated for various mistag rates (0.1, 0.01, 0.001) on both training and validation datasets.*

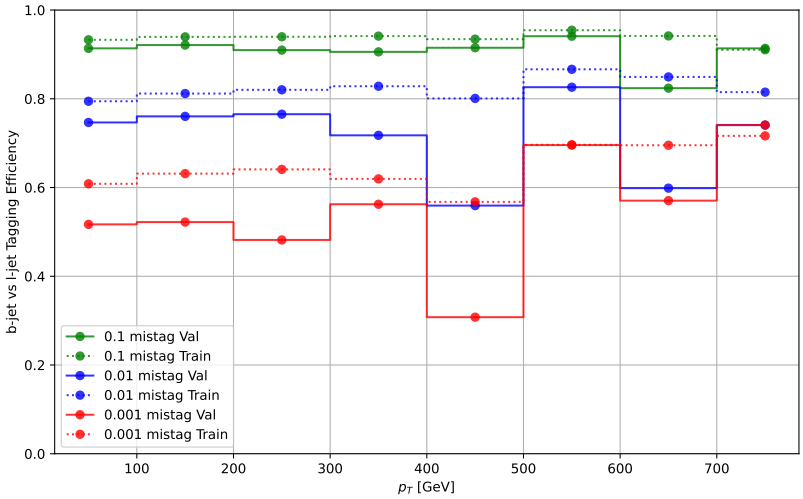


Figure 4.10: *Jet tagging efficiency for b-jet versus c-jet discrimination using the MLP model, as a function of jet p_{T}. Performance is evaluated for various mistag rates (0.1, 0.01, 0.001) on both training and validation datasets.*

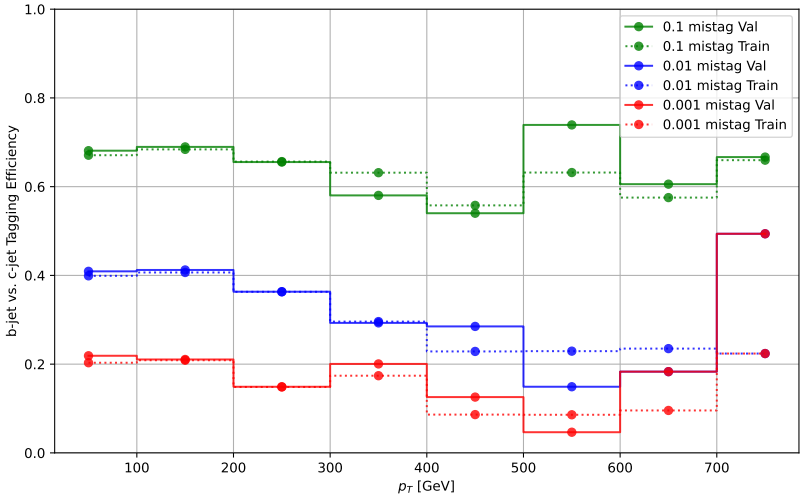


Figure 4.11: *Jet tagging efficiency for b-jet versus c-jet discrimination using the JetRetNet model, as a function of jet p_{T}. Performance is evaluated for various mistag rates (0.1, 0.01, 0.001) on both training and validation datasets.*

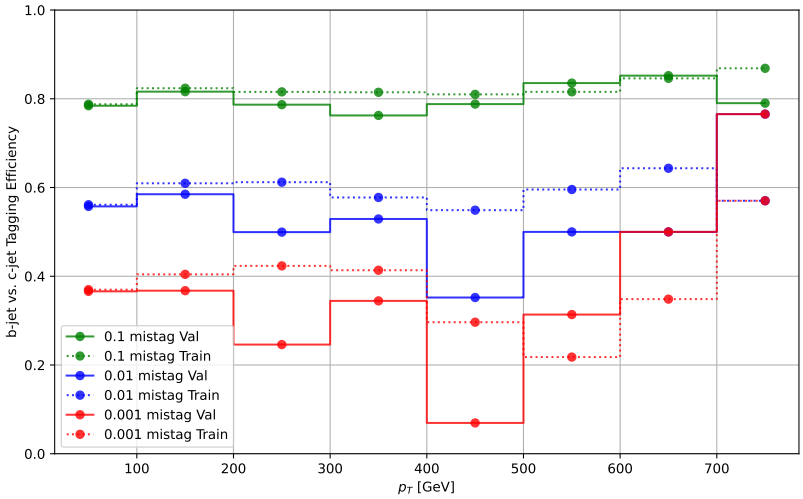
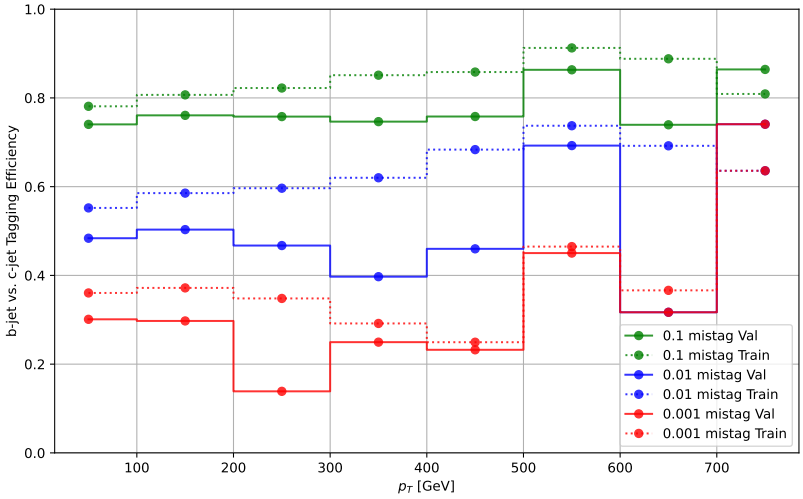


Figure 4.12: *Jet tagging efficiency for b-jet versus c-jet discrimination using the XGB model, as a function of jet p_{T}. Performance is evaluated for various mistag rates (0.1, 0.01, 0.001) on both training and validation datasets.*



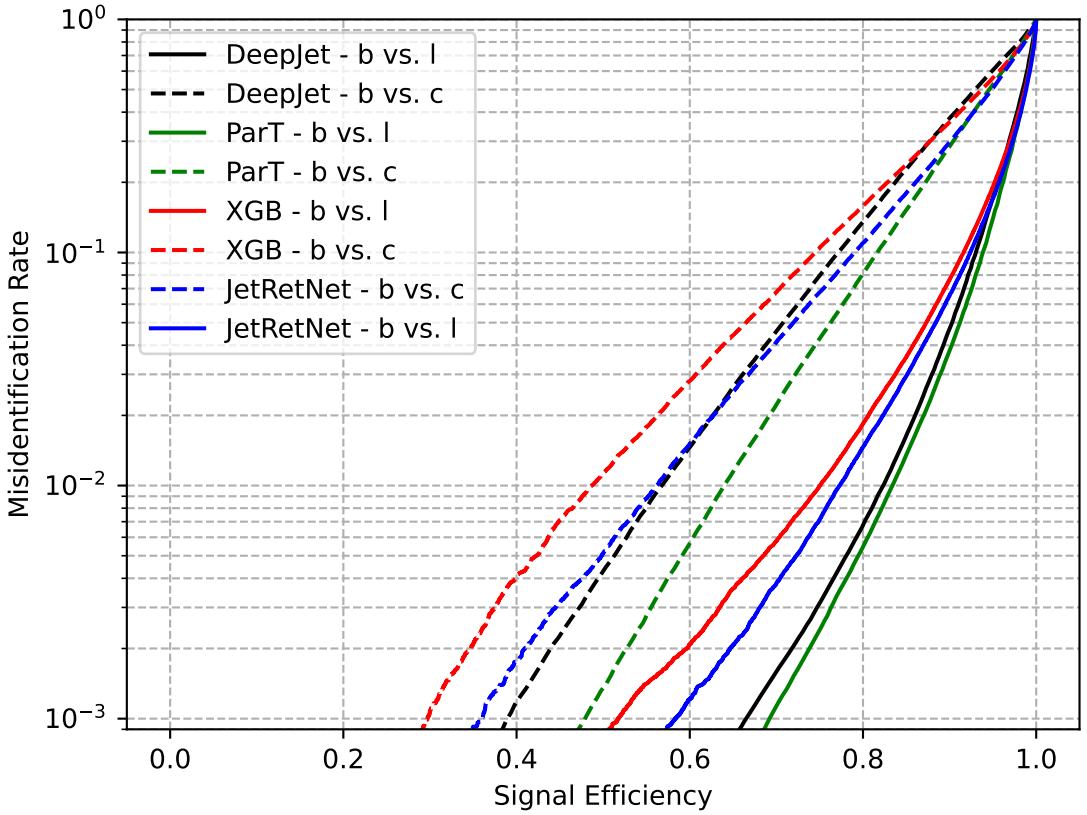


Figure 4.13: *Signal efficiency vs. Misidentification rate of b-jets. Here, the dashed curves are for b-jet vs. c-jet, and the solid curves are for b-jet vs. light-jet classification. The models XGB, JetRetNet, DeepJet, and ParT are shown in red, blue, black, and green, respectively.*

5. CONCLUSIONS

We have trained our models with a semi-leptonic $t\bar{t}$ decay dataset taken from CERN CMS Open Data Portal [25]. After pre-processing data to our desired shape and features, we trained a DNN, an XGB, and a retention-based network called JetRetNet. JetRetNet outperformed the other models by a large margin. By comparing the JetRetNet's performance with the state-of-the-art models, although the comparison is unfair, the potential of the JetRetNet for future use is evident. With a much smaller training dataset, we never expected our models to surpass the state-of-the-art performance, but JetRetNet's performance is very comparable to DeepJet's, especially on the loose working point. Another benefit is the training and inference times of JetRetNet with its low parameter count.

This has been the preliminary work for using RetNet for b-jet tagging. The architecture can be improved further to adapt them for this specific task. Another RetNet-based decoder can also improve the JetRetNet architecture by changing the FFN layer in the decoder part of the model in Figure 3.9.

Also, the hyper-parameter fine-tunings are done within the computational limitations. Hence, all models can perform better by properly fine-tuning the hyper-parameters using a random search or grid search (or a combination of both).

Apart from improving the model architectures and the training procedures, the dataset has been the most important element of b-jet tagging with ML methods. Although significant time has been spent on this topic, many improvements or experiments are still needed. First, another type of event can be considered: hadronic $t\bar{t}$ decays, or the QCD multijet events can be used. It can be further improved by creating a dataset from many decay channels to represent all the b-jets within SM that can emerge in the collider experiments. For example, in our dataset, all the b-jets arise from the decay of the top quark. But the Higgs boson also decays into $b\bar{b}$ most of the time, which is not included in our dataset.

The p_{T} range of the dataset can also be improved to be more evenly distributed. Because of the nature of Jet p_{T} distributions, most of these models perform well in the low p_{T} range but fail drastically in high p_{T} domains. A new dataset can be artificially reconstructed to overcome this effect and maintain stable performance across all p_{T} ranges.

Apart from the selection of the decay channels, one can also change the ordering of the tracks and SVs. We have used the decreasing p_{T} ordering. Some time has been spent selecting this order, but other orderings are used in the literature. For example, decreasing impact parameter or decreasing impact parameter significance are also commonly used [35].

The change of the parameters that are used can be another improvement. As understood, ParT uses either the logarithms or the hyperbolic tangent of some of the parameters [37]. This method can be tried on our dataset and compared by giving the parameters as they are and checking if the performance improves.

Many new features can also be added to the dataset, which will most probably improve the performance. But this has a side-back, a question of inference time and performance. So, in theory, you can give all the inputs from the detector and also many high-level features, but it will be unable to identify a b-jet within the given time. So, it is important to identify the most informative and also sufficient features for the desired performance.

We have only used the tracks and the secondary vertices in the jet. Both of the information are taken in the Tracker part of the CMS Detector. The models can also be extended to include the calorimeter jets. In addition to the way we give tracks and SVs, a 2D calorimeter image of the jet (in the η, ϕ plane) can be fed into the models in parallel. The first thing that comes to mind is to use a CNN for this image, but transformer-based methods can also be helpful. We did not choose this method as b-quarks travel very short distances before they decay, usually just before the tracker or inside it. So before the jet

products arrive at the calorimeters, most of the decay during the jet formation has already occurred. In this work, we have assumed that the later the jet constituent emerges during the process of jet formation, the less information it contains. This is just a logical/intuitive assumption, but one never knows which feature contains what kind of information without trying it out. Hence, the addition of the CaloJets could improve the model's performance.

Another drawback of using only the tracker is that we don't have any information on the neutral particles. To improve the model even further, we can feed the information from the ECAL and HCAL separately to the model by creating two calorimeter images. Although the idea is nice, PF objects do not separate ECAL and HCAL, requiring the use of more raw data. However, an idea similar to the DeepJet can be employed; instead of ECAL and HCAL images, charged and neutral calorimeter images can easily be reconstructed. These two images can be passed through CNNs in parallel, and the outputs can be concatenated to the outputs from the RetNet layers of JetRetNet, as well as the global jet features.

Of course, the dataset size can be improved further. Our models are trained with 4 million jets, while the state-of-the-art models given in Section 4.3 are trained with 140 million jets. This is a very big difference, but also another hint at the potential of the JetRetNet. Further studies on the pile-up and calibrations on the real collision data are also mandatory next steps in this analysis. The model can also be altered to work as an event-based multi-b-jet tagger.

This work introduces the use of some of the novel DL models, like RetNet and XGB, for the problem of b-jet tagging in particle physics. Although this has been a long-standing problem, we show that with the fast-paced tempo of the ML world, there are always new methods to be applied to the old problems. The state-of-the-art models are not out-performed, but this was not expected. The potential of the JetRetNet model is especially demonstrated, and many improvements are argued.

REFERENCES

[1] W. C. contributors, "Standard model of elementary particles," 2012. Accessed: 2024-12-13.

[2] M. E. Peskin and D. V. Schroeder, *An Introduction to quantum field theory*. Reading, USA: Addison-Wesley, 1995.

[3] H. D. Politzer, "Reliable perturbative results for strong interactions," *Phys. Rev. Lett.*, vol. 30, pp. 1346–1349, 6 1973.

[4] V. Trimble, "Existence and nature of dark matter in the universe," *Annual Review of Astronomy and Astrophysics*, vol. 25, no. Volume 25, 1987, pp. 425–472, 1987.

[5] M. Thomson, *Modern Particle Physics*. Cambridge University Press, 2013.

[6] D. J. Gross and F. Wilczek, "Asymptotically free gauge theories. i," *Phys. Rev. D*, vol. 8, pp. 3633–3652, 11 1973.

[7] R. Mazini, "Introduction to monte carlo event generators," 9 2017. Accessed on 2023-09-19.

[8] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-ktjet clustering algorithm," *Journal of High Energy Physics*, vol. 2008, p. 063–063, Apr. 2008.

[9] R. Atkin., "Review of jet reconstruction algorithms," *Jour. of Phys: Conf Series, 645(1):012008*, 09 2015.

[10] R. Ellis, W. Stirling, and B. Webber, "Qcd and collider physics," *QCD and Collider Physics, by R. K. Ellis and W. J. Stirling and B. R. Webber, pp. 449. ISBN 0521545897. Cambridge, UK: Cambridge University Press, December 2003.*, vol. 8, 12 2003.

[11] I. Neutelings, "Standard model particles and masses." `https://tikz.net/sm_particles_masses/`, 2024. Accessed: 2024-12-16.

[12] S. Navas *et al.*, "Review of particle physics," *Phys. Rev. D*, vol. 110, no. 3, p. 030001, 2024.

[13] I. Neutelings, "Standard model decay matrix." `https://tikz.net/sm_decay_matrix/`, 2024. Accessed: 2024-12-16.

[14] G. Arcadi, A. Djouadi, and M. Raidal, "Dark matter through the higgs portal," *Physics Reports*, vol. 842, p. 1–180, Feb. 2020.

[15] T. C. collaboration, "The cms experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, p. S08004, 8 2008.

[16] T. A. collaboration, "The atlas experiment at the cern large hadron collider," *Journal of Instrumentation*, vol. 3, p. S08003, 8 2008.

[17] B. Işıldak, A. Hayreter, M. Hüdaverdi, F. Ilgın, S. Salva, E. Simsek, and S. Güyer, "Investigating the violation of charge parity symmetry through top quark chromo-electric dipole moments by using machine learning techniques," 06 2023.

[18] T. C. collaboration, "Measurement of the $\mathrm{t\bar{t}}$ production cross section and the top quark mass in the dilepton channel in pp collisions at $\sqrt{s} = 7$ tev," *Journal of High Energy Physics*, vol. 2011, July 2011.

[19] I. Neutelings, "Jets in t-tbar events." `https://tikz.net/jets_ttbar/`, 2024. Accessed: 2024-12-16.

[20] I. Neutelings, "Standard model decay pie chart." `https://tikz.net/sm_decay_piechart/`, 2024. Accessed: 2024-12-16.

[21] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," *Journal of High Energy Physics*, vol. 2014, July 2014.

[22] C. Bierlich, S. Chakraborty, N. Desai, L. Gellersen, I. Helenius, P. Ilten, L. Lönnblad, S. Mrenna, S. Prestel, C. T. Preuss, T. Sjöstrand, P. Skands, M. Utheim, and R. Verheyen, "A comprehensive guide to the physics and usage of pythia 8.3," 2022.

[23] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi, "DELPHES 3, A modular framework for fast simulation of a generic collider experiment," *JHEP*, vol. 02, p. 057, 2014.

[24] S. A. et al., "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[25] C. O. D. Portal, "CMS open data: 2016 MiniAOD data," 2023. Accessed: 2024-12-17.

[26] I. Neutelings, "3d axis for cms." `https://tikz.net/axis3d_cms/`, 2024. Accessed: 2024-12-16.

[27] M. Cacciari and G. P. Salam, "Pileup subtraction using jet areas," *Physics Letters B*, vol. 659, no. 1, pp. 119–126, 2008.

[28] T. C. collaboration, "Identification of heavy-flavour jets with the cms detector in pp collisions at 13 tev," *Journal of Instrumentation*, vol. 13, p. P05011–P05011, May 2018.

[29] I. Neutelings, "Jet b-tag visualization with tikz." https://tikz.net/jet_btag/, 2024. Accessed: December 9, 2024.

[30] IBM, "What is gradient descent?." https://www.ibm.com/think/topics/gradient-descent#:~:text=Gradient%20descent%20is%20an%20optimization,each%20iteration%20of%20parameter%20updates., 2023. Accessed: 25.01.2024.

[31] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, p. 785–794, ACM, Aug. 2016.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[33] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei, "Retentive network: A successor to transformer for large language models," 2023.

[34] A. A. Guvenli and B. Isildak, "B-jet tagging with retentive networks: A novel approach and comparative study," 2024.

[35] E. Bols, J. Kieseler, M. Verzetti, M. Stoye, and A. Stakia, "Jet flavour classification using deepjet," *Journal of Instrumentation*, vol. 15, p. P12012–P12012, Dec. 2020.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[37] H. Qu, C. Li, and S. Qian, "Particle transformer for jet tagging," 2024.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[39] R. Medeiros, P. P. Filho, M. Barata Rodrigues, S. Pires Pinheiro da Silva, C. M. Dourado Jr, and V. Albuquerque, "Retracted article: Lung nodule malignancy classification in chest computed tomography images using transfer learning and convolutional neural networks," *Neural Computing and Applications*, vol. 32, 11 2018.

[40] Z. P., "Roc curve," 2016. Accessed: 2024-12-16. Licensed under CC BY-SA 4.0.

[41] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics," *Annual Review of Nuclear and Particle Science*, vol. 68, no. Volume 68, 2018, pp. 161–181, 2018.