JET ENERGY CORRECTIONS WITH DEEP LEARNING

A Thesis

by

Ömer Fatih Dokumacı

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Physics

Özyeğin University
August 2024

JET ENERGY CORRECTIONS WITH DEEP LEARNING

Advisor: Prof. M. Burçin Ünlü, Özyeğin University
Coadvisor: Prof. Bora Işıldak, Yıldız Technical University

Approved by:

Prof. M. Burçin Ünlü, Advisor
Dept. of Natural & Mathematical Sciences
Özyeğin University

Prof. Taylan Akdoğan
Dept. of Natural & Mathematical Sciences
Özyeğin University

Prof. Bora Işıldak, Coadvisor
Dept. of Physics
Yıldız Technical University

Asst. Prof. Hale Sert
Dept. of Physics
İstanbul University

Date Approved: August 2, 2024

Prof. H. Fatih Uğurdağ
Dept. of Electrical & Electronics Eng.
Özyeğin University

To my family and friends

ABSTRACT

The LHC at CERN accelerates and collides protons at enormous energies and currently is the most powerful particle accelerator available. The CMS detector is a sophisticated device designed to record the cascade of particles arising from these collisions. With the aid of Monte Carlo simulations, detected particles are reconstructed and clustered together to create physics objects known as jets. The importance of jets is linked to the indirect study of quarks and gluons, since they are not observed freely in nature. Tasks such as reconstruction of high-energy processes, identification of particle interactions and search for new physics beyond the Standard Model all rely on precise study of jets. Due to the detector response and additional factors, the measured energies of the jets need to be calibrated according to simulated truth values. Hence, in the CMS experiment, a sequence of jet energy correction methods is applied to align the jet energies with the true values. In this thesis, we utilized a deep learning approach to further improve standard jet corrections. In recent years, the incorporation of machine learning algorithms has shown great results in high-energy physics studies. Tasks such as jet tagging and particle reconstruction have benefited from recent developments in deep learning. For the jet energy correction task, which is a regression problem, we used two models: the Deep Sets-based Particle Flow Network and a more straightforward fully connected deep neural network. The results of the corrections are presented primarily by two metrics that allow for a comprehensive understanding of performance: average response and relative jet energy resolution. The models surpassed standard energy corrections on a large scale in both metrics.

ÖZETÇE

Büyük Hadron Çarpıştırıcısı (LHC), yüksek enerji fiziği alanında kullanılan en güçlü parçacık çarpıştırıcısıdır. LHC deneylerinin merkezinde, proton proton etkileşimlerinden kaynaklanan parçacık ışınını kaydetmek için tasarlanmış bir dedektör olan CMS dedektörü bulunmaktadır. Proton çarpışmalarından ortaya çıkan parçacıklar, Monte Carlo simülasyonları aracılığıyla jet ismiyle bilinen objeleri oluşturmak üzere yeniden yapılandırılır ve kümelenir. Jetler, doğada serbest halde bulunamayan parçacıklar olan quark ve gluonların deneysel göstergeleridir. Yüksek enerji süreçlerinin yapılandırılması, parçacık etkileşimlerinin tanımlanması ve Standart Model ötesi fizik arayışları gibi araştırmalar jet çalışmalarına dayanır. Dedektör tepkisi ve bazı ek faktörler nedeniyle jetlerin ölçülen enerjilerinin simülasyondan elde edilen "gerçek" değerlere göre kalibre edilmesi gerekir. Bu nedenle CMS deneyinde jet enerjilerinin gerçek enerjilerine yaklaştırılması için bir seri jet enerji düzeltmesi uygulanmakta. Bu tezde, standart jet düzeltmelerini daha da geliştirmek için derin öğrenme yaklaşımı kullanıldı. Son yıllarda makine öğrenme algoritmalarının dahil edilmesinin, yüksek enerji fiziği çalışmalarında güzel sonuçlar verdiği görülmekte. Yakın zamanda derin öğrenme yöntemlerindeki gelişmelerden, jet etiketleme ve yeniden yapılandırma gibi görevler de payını almış olmakta. Bir regresyon problemi olan jet enerjisi düzeltme işlemi için iki model kullandık: Derin Set tabanlı Parçacık Akış Ağı ve daha basit, tam bağlantılı bir derin sinir ağı. Düzeltmelerin sonuçları temel olarak performansın kapsamlı bir şekilde anlaşılmasına olanak tanıyan iki ölçümle sunuldu: ortalama enerji tepkisi ve relatif jet enerjisi çözünürlüğü. Modellerin her iki ölçümde de standart enerji düzeltmelerini büyük ölçüde aştığı gözlemlendi.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis coadvisor Prof. Bora Işıldak, whom helped me both in the choosing of the thesis topic and in the research process. He was always ready to help in my times of confusion during the research, implementation and writing periods and provided me with great insights and ideas on particle physics and deep learning. Secondly, I would like to thank my thesis advisor Prof. M. Burçin Ünlü for always being there when I needed help, especially for the organization of the formal procedures. And lastly, I want to thank my family, and my dear friend and colleague Onur Kerem Özmen, for always encouraging my academic pursuits and their endless support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The Standard Model of particle physics is considered one of the most successful models in physics. The Large Hadron Collider (LHC) is the world's largest particle collider in the world, designed for experimenting on this model. Compact Muon Solenoid (CMS) detector is a state-of-art device at LHC, specifically designed to record and analyze the traversing cascade of particles resulting from proton-proton (pp) interactions. The cascade of particles resulting from the collisions are recorded as they transverse through the layers of the detector and the detected signals are processed to reconstruct the particles with the aid of Monte Carlo simulations.

A main interest in the CMS experiment is the concept of particle jets—sprays of particles emerging from these collisions. Jets are observable manifestations of quarks and gluons that are produced in pp collisions. Hence, many particle physics analyses such as reconstruction of high-energy processes, particle identification and search for new physics beyond Standard Model rely on jet studies and an accurate detection of jet properties. Jets are clustered by the reconstructed particles according to the parton that initiated the process. The uncertainties in the detection process produces the necessity for jet energy corrections. Many factors can cause so called uncertainty in the detection, such as instrumental effects, non-uniformities in detector response, and the influence of additional pp interactions, known as pileup. These factors introduce biases in the measured jet energies, which if left unaddressed, can compromise the integrity of analyses and hinder the results. Because of this reason, the refinement of jet energy measurements is a critical task in high-energy physics (HEP) experiments. To address these challenges, standard jet energy corrections

(JEC) are conventionally applied to experimental data. These corrections aim to rectify discrepancies arising, ensuring that the reconstructed jet energies faithfully reflect the ground truth of the initial particles. JEC is a very important task in experimental physics since a more accurate measurement not only helps with the testing of particle physics theories but also with the identification of particle properties and discovery of new particles.

The incorporation of machine learning methodologies has significantly transformed the workflow of HEP like many other different fields of research. Notably, deep learning has emerged as a particularly valuable tool, and among its diverse architectures. Multilayer perceptrons, graph neural networks and convolutional neural networks have stood out as exceptionally expressive and versatile choices. These networks find application across a spectrum of tasks, including the task of reconstructing particle tracks to the comprehensive classification of entire events. The standard jet energy corrections can be further enhanced upon the implementation of deep learning algorithms. In this thesis we used two different deep learning models, a fully connected deep neural network and a Deep Sets-based model, the Particle Flow Network to improve the standard jet energy corrections.

CHAPTER II

PARTICLE PHYSICS

Fundamental constituents of matter and the forces through which they interact are understood through particle physics. The Standard Model of particle physics successfully describes the underlying principles of particle interactions, and even predicts new particles and phenomena before they are observed experimentally. Currently, among many particle colliders built to experiment on particle physics, the Large Hadron Collider (LHC) stands as the largest one with many large and small detectors present on the interaction points. The Compact Muon Solenoid (CMS) detector at LHC is of importance for this thesis. This chapter includes some of the theoretical components of particle physics, then the CMS experiment and its structure will be explained.

2.1 The Standard Model

The Standard Model (SM) of particle physics describes not only the electromagnetic, weak, and strong nuclear forces, also the elementary particles that make up matter. The model is proved to be remarkably successful with its power to predict new, unobserved particles and phenomena.

SM classifies all known elementary particles through under two categories: fermions and bosons. 12 elementary particles with spin 1/2 with 12 corresponding antiparticles are called the fermions. Fermions are further divided into quarks and leptons. Quarks are the building blocks of composite hadrons, with the most stable ones being the protons and neutrons which constitutes atomic nuclei. Quarks come with three different generations: I, II and III as seen in Figure 1. A higher generation generally indicates higher mass and less stability. Because of this, most matter in the universe consists of up and down quarks. Unstable, higher generation quarks decay into the

first generation by weak interaction.

4 kinds of gauge bosons of spin 1 are included in the model. Gause bosons are the force carriers of particle interactions. The SM describes the fundamental interactions as virtual gauge boson exchange between the fermions. For example, the electromagnetic force between two charged particles is mediated by an exchange of photons. Strong interaction, responsible for binding of quarks is mediated by gluons and the weak interaction is mediated by W and Z bosons as the force carriers. As far as gravity is concerned, a force carrier is yet to be found. Although graviton is proposed as a mediating particle, it is not observed. So the gravity is one of the unexplained phenomenon in the SM.

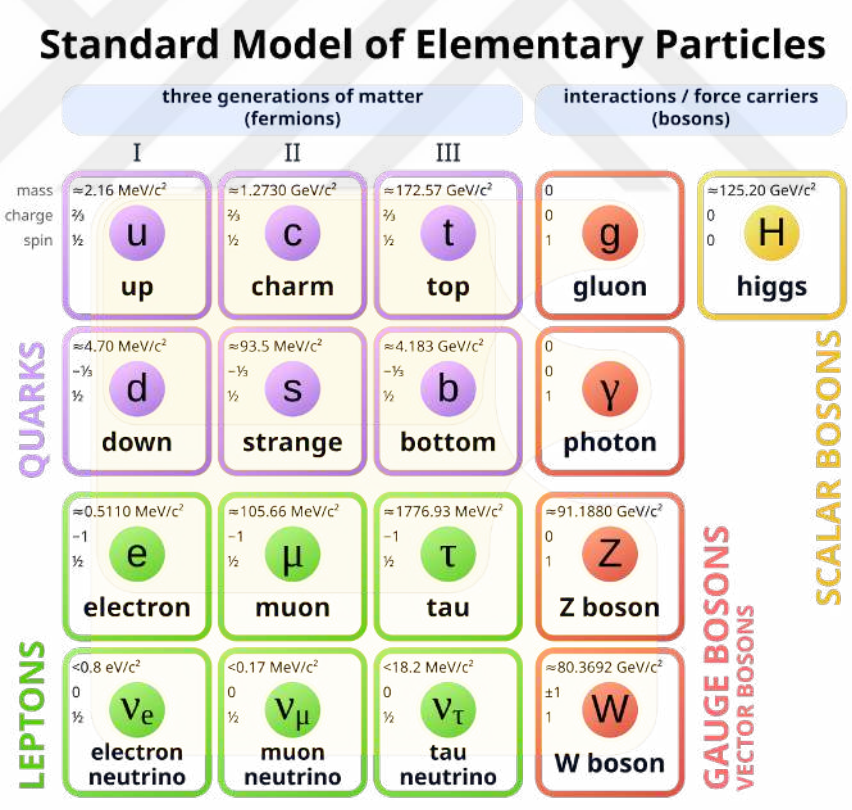


Figure 1: Elementary particles of the Standard Model [1]. The 12 fundamental fermions and 5 fundamental bosons. Brown loops indicate coupling between bosons and fermions.

A boson with spin zero is referred as a *scalar* boson. Higgs boson is a scalar boson that was confirmed recently in 2012 [2] and the theory behind won the 2013 Nobel Prize. Higgs boson is the only known fundamental scalar boson yet and it endows particles with mass through spontaneous symmetry breaking. The SM has achieved rigorous experimental validation and with the confirmation of the Higgs boson at the LHC it could be said that it solidified its place among the most successful models.

2.2 Quantum Chromodynamics

Quantum chromodynamics (QCD) describes the strong interaction between the quarks and gluons and is one of the cornerstones of the SM. Quarks and gluons are collectively known as partons. The mediation of strong interaction is done by gluons between the quarks and the binding of the quarks can form composite the hadrons such as the proton and neutron as a product.

Quarks come with six different flavors: up, down, charm, strange, top and bottom, and three color charges analog to electric charge: red, green and blue. Antiquarks carry the corresponding anticolor charges: antired, antigreen, and antiblue. The term "chromodynamics" is attributed to the color charge that quarks possess. Combination of these three colors create the neutral charged hadrons, with charge denoted as white.

Two important properties of the QCD are the *color confinement* and *asymptotic freedom*. Color confinement states that a free parton with a color charge can not exist. In the scenario that partons are separated up to a certain point, quark-antiquark pairs are spontaneously created due to the increase in energy in the gluon field between them. Partons can only exist with color neutral combinations, hadrons. These white colored combinations do not experience the strong force over long distances and are observed as stable particles. In contrast, asymptotic freedom becomes dominant at high energies, or at very short distances. Partons behave as approximately free particles under such conditions.

2.3 Collision Processes

The LHC accelerates protons to nearly the speed of light and collides two beams of protons at designated interaction points, leading to interactions between quarks and gluons that create a wide range of particles. pp collisions involve hard scattering, hadronization and showering processes. Each process describes a step in the process of evolution of resulting particles. Many of the produced particles are unstable and decay almost instantaneously, but *jets*, narrow cones of hadrons produced in the hadronization process, is a main interest of research in CMS experiment, since they allow extraction of the properties of the original quarks.

Only the electrons, photons, protons and undetectable neutrinos are stable products of the collision. When high energy photons interact with the matter, e+e- pairs are produced. High energy electrons radiate photons due to the *bremsstrahlung* process of charged particles. The produced photons also decay into e+e- pairs. As a result, a cascade of particles emerge from the interactions with the medium. This process is called the *electromagnetic shower* and it continues until the threshold that photons can no longer produce pairs or the average energy of particles is smaller than the critical energy E_c. E_c is the energy in which the rate of ionization and bremsstrahlung are equal. For muons, which are approximately 200 times more massive than electrons, the main energy loss source is ionization instead of the bremsstrahlung since the rate is inversely proportional with mass.

High energy hadrons primarily interact with the nuclei of the medium with strong interaction processes. Hadrons that hard interact with the nuclei produce secondary hadrons (e.g pions and kaons) which continue the collisions and create a shower of hadrons called the *hadronic shower*. Hadronic showers have much more variability compared to the electromagnetic counterpart because of a greater number of final states.

2.3.1 Hard Scatter

Scattering occurs after two protons collide. The term "hard" arises from a large momentum transfer between the interacting partons inside the protons. This large momentum transfer can lead to creation of heavy quarks, gauge bosons, or other exotic particles during the interaction. The likelihood or probability of a particular hard scattering process occurring is described by the matrix element calculated using perturbation theory.

To attain momenta of the constituents *parton distribution functions* (PDFs) are used. PDFs describe the probability distributions of the partons carrying a specific fraction of the hadron's total momentum. The PDF $f(x, Q^2)$ for a quark or gluon at a particular energy scale provides the probability of finding that parton carrying a certain fraction x of the total momentum of the hadron. The variable x is often referred to as the *Bjorken x*. Parton distributions are inferred from experimental data for a specific energy scale before guided to other energy scales Q^2 on account of the non-perturbative behavior of partons that are not eligible to observe as free particles. The dependence on the energy scale is described by the evolution equations, which are derived from QCD. As the energy scale increases, quarks and gluons may radiate additional partons.

2.3.2 Parton Showers

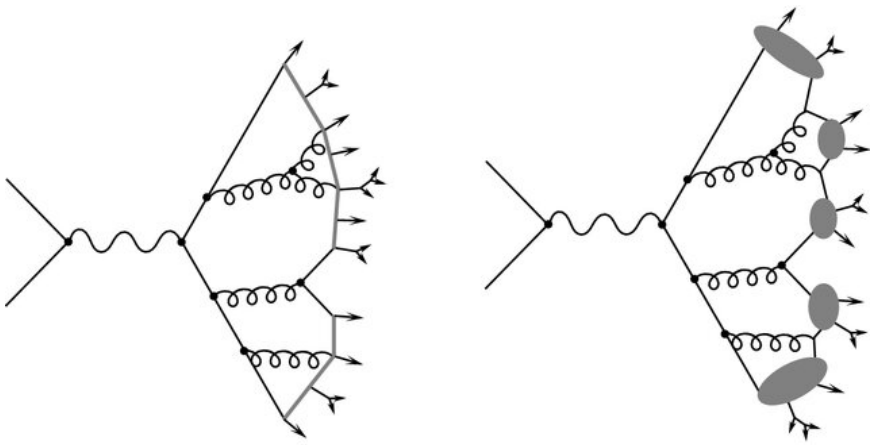
Since partons are accelerate violently due to large momentum transfer after the hard scattering, they emit QCD radiation (gluons). This process is similar to QED radiation (photons) emitted by accelerated charges. Although accelerated products of QCD radiation, the gluons, can emit further radiation because of their color charge. As a result they create quark-antiquark pairs or further radiate more gluons. This cascading effect is known as a *parton shower*. Created partons continue to fragment until they reach an energy scale where non-perturbative QCD effects become

significant.

Albeit parton showers correspond to high-order corrections to the hard scattering, calculation of these corrections is unfeasable. This problem can be overcome by using an approximation scheme where the most significant contributions are considered for each order. The dominant contributions can come either from low energy gluon emissions or the colinear parton splittings.

2.3.3 Hadronization

The decrease in emitted partons energy dictates a point where it is no longer possible to exist as individual particles. This is attributed to the color confinement described in Section 2.2. This limit is known as the hadronization boundary and beyond it the *hadronization* process starts. During the process the partons combine to form color neutral hadrons which obey the color confinement. Hadronization occurs in a non-perturbative regime of QCD, so the direct calculation is not feasable.Because of this phenomenological models have to be used. Figure 2 shows the two models; Lund String Model and the Cluster Model. The String model is based on QCD lattice simulation observations. At sufficiently large distances the potential energy between pair of color sources (a heavy quark and an antiquark pair for example) increases linearly with distance. And this linear relation corresponds to an attraction force that is not dependent on distance. A potential reason for this behavior is gluonic fields self-attraction property. As quarks are seperated, the gluonic field collapses into a string-like (or a tube-like) structure with constant thickness of the order 1 *fm*, roughly the size of a proton. The cluster model on the other hand, groups partons into color neutral clusters. This clustering is based on spatial grouping and color connections. Then, the localized color neutral clusters decay into stable hadrons.



(a) String hadronization (b) Cluster hadronization

Figure 2: Two models for describing the hadronization process: Lund String Model and Cluster Model. The example in the figure is the process of e+e- annihilation to hadrons. String model on the left depicts the formation and breaking of color strings. Cluster model on the right shows the formation and decay of color-neutral clusters [3].

2.3.4 Jet Definition

In order to obey the QCD color confinement, the color charged fragments resulting from the collided protons create other colored particles around them to form an ensemble of color neutral hadrons. Throughout the hadronization process of partons, a cone of hadrons and other produced particles are called *jets*. Since quarks and gluons cannot exist freely in nature due to color confinement, study of them can be accomplished with jet related studies.

Several algorithms are present to define and reconstruct jets from the detected hadrons. Common algorithms include cone algorithms and sequential clustering algorithms such as the k_t and anti-k_t algorithms. These clustering algorithms group nearby particles into a single jet based on their momenta and angular separations. Clustering algorithms will be explained later in Chapter 4.

Specific shapes and substructures of jets depend on the underlying physics of the hard scattering process and subsequent parton showers and observables like jet mass, jet substructure, and jet shapes provide information about the internal structure of

jets.

2.4 Large Hadron Collider

Among many particle accelerates present, the LHC at the European Organization for Nuclear Research (CERN) stands as the most powerful one so far. It is a circular accelerator that lies in an underground tunnel, with a circumference of roughly 27 km. The LHC can accelerate protons up to center-of-mass energies of 14 TeV and speed of 0.999999990 c, where c is the speed of light. In such conditions, for an accelerated proton, it takes less than 90 μs to travel 26.7 km around the main ring. The feat of creating such a strong magnetic field is done by superconducting magnets present at the tunnels. Close to absolute zero temperatures are maintained to preserve superconducting properties of the magnets.

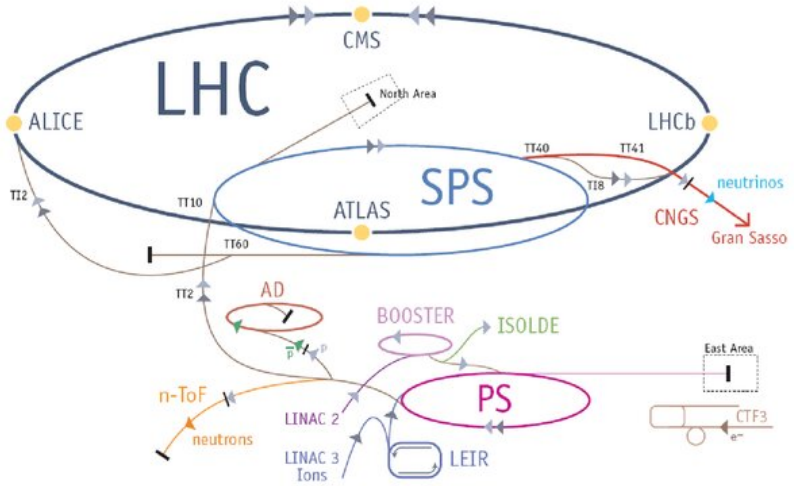


Figure 3: Schematic of accelerator chain of LHC [4]. Four main detectors at the main ring; ATLAS, CMS, ALICE and LHCb can be seen.

ATLAS, CMS, ALICE, and LHCb are the four main detectors located at the crossing points around the ring. Aside the four main detectors, there are many other smaller scale detectors for different experiments. The rate of events of a given collision process at LHC is proportional to the cross section. The luminosity is defined as the

number of events per cross section and is an important value to characterize collider performance. A higher luminosity means more data to analyze and a higher chance of detecting rare particle interactions. The luminosity L is defined as:

$$L = \frac{1}{\sigma}\frac{dN}{dt} \tag{1}$$

where N is the number of protons per bunch and f is the bunch crossing frequency. The cross section σ represents the effective collision area. ATLAS and CMS are the two high luminosity experiments at LHC that is able to reach a peak luminosity of $L = 10^{34} cm^{-2} s^{-1}$.

Time interval between successive bunches of protons circulating in the ring is known as *bunch spacing*. The LHC operates with a radiofrequency (RF) system that accelerates and bunches protons before they are brought into collision. Bunches are grouped with the intervals of 25 ns between the collisions. Multiple pp collisions can occur for a bunch crossing. This phenomenon is known as the *pileup* and can increase background signals by affecting the detector response and make it harder to determine where the collision products originate from. The methods to mitigate pileup effect will be presented in Chapter 4.

2.5 Compact Muon Solenoid

The CMS is a 14,000-tonne general-purpose detector located at one of the four collision points. The CMS can be considered very compact for all the detector material present. The diameter of the detector is about 15 meters and length is about 22 meters. It ensures that particles are detected at a wide range with around 4π coverage. The CMS is built around a powerful solenoidal magnet that bends charged particle trajectories. The magnet can generate a magnetic field of roughly 3.8 T [5] ,which is about 10^5 times the magnetic field of earth.

The CMS experiment adopts a right-handed Cartesian coordinate system defined with the origin at the detector center. shown in Figure 4. The x-axis is pointed

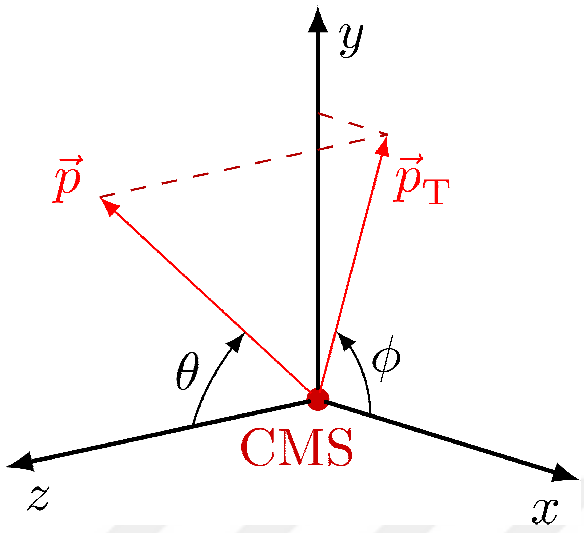


Figure 4: The diagram of 3D coordinate system of the CMS [6].

radially inward with respect to the beamline into the ring center. The y-axis is perpendicular to the ring plane pointing vertically upward. And the z-axis aligned with the beam direction. Aside from the Cartesian, cylindirical coordinates are also used as well. The polar angle θ is measured from the z-axis, and the azimuthal angle ϕ is measured in the $x - y$ plane from the x-axis. From there spatial coordinate *pseudorapidity* η can be defined as:

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

Pseudorapidity is the angle relative to the beam axis and is an important coordinate for collider experiments. Differences in pseudorapidity are invariant under Lorentz boosts along the beam direction. So pseudorapidity remains consistent regardless of the particle's energy or momentum in the beam direction.

The CMS detector consists of several concentric layers built on top of each other as shown in Figure 5. The innermost component is the silicon tracker, which consists of pixel and strip detectors. When a particle transverses this layer, electromagnetic interaction with the silicon layer produces a signal. By using all the signals left by the particle it is possible to identify the track. The second layer is the *electromagnetic calorimeter*, or ECAL in short. ECAL is made of lead tungstate ($PbWO_4$) crystals

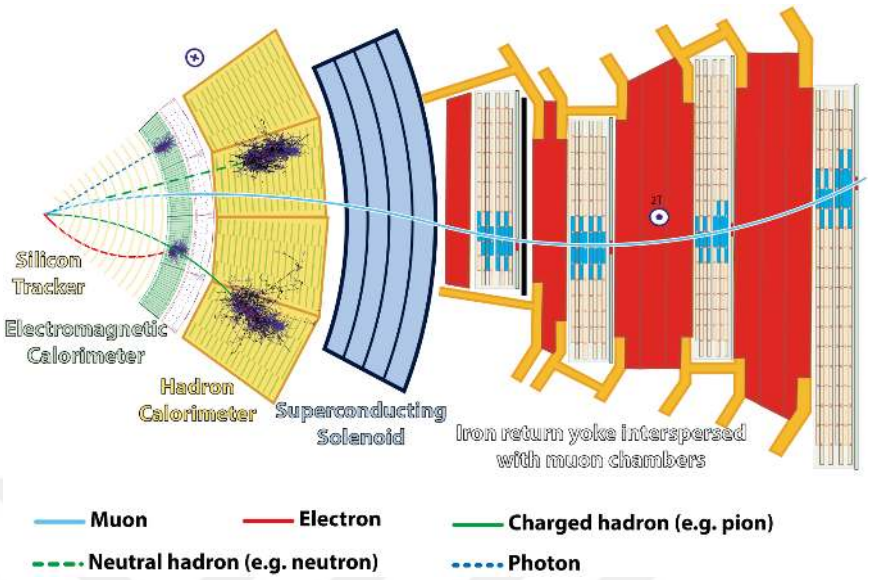


Figure 5: A transverse slice of the CMS detector. Particles detected by each sub-detector layer and the trajectories is can be seen [7].

and spatially covers up to $|\eta| < 3.0$. The main purpose of this layer is to measure photon and electron energies by completely stopping them. As these particles pass through, they scintillate the crystals and produce light in small bursts which can later be used to determine the energies.

Hadrons pass through the ECAL layer and are stopped by the outer layer called the *Hadron Calorimeter* (HCAL), covering the spatial region of $|\eta| < 5.2$. A significant amount of the detector's volume is occupied by the HCAL which is made of high density absorber layers to induce particle showers and brass and plastic scintillators to measure the energy. The energy resolution of the HCAL is directly related to the scintillator response and the depth of the calorimeter.

Muons on the other hand are heavy and penetrating particles and can transverse any of the calorimeter layers with almost zero energy loss. The outermost layer of the detector responsible for muon detection is called the *muon system*. Drift Tubes (DT), Cathode Strip Chambers (CSC), and Resistive Plate Chambers (RPC) as different types of sub detectors are present at the muon system. The DT system consists of cylindrical tubes filled with a gas and is located in the barrel region ($|\eta| < 1.3$) of

the system. Muons transversing a DT ionizes the gas, and the electrons drift to the anode to produce a detectable pulse. Depending on the timing of the signal it is possible to determine the crossing point of the muon in the cell. The CSC system is situated in the endcap ($1.3 < |\eta| < 3.0$) region. CSCs are multi-wire proportional chambers with cathode strips. Muons ionize the gas in the CSCs, and the resulting electrons drift towards anode wire planes and cathode strip planes. Perpendicular strips and wires result in a two position coordinate for each passing particle. RPCs are additional detectors consisting of two parallel plates, a positively-charged anode and a negatively-charged cathode separated by a gas volume, positioned both in the barrel and end cap regions of the system. Passing muons ionize the gas and the knocked out electrons hit other atoms causing a stream of electrons. A muon's momentum is determined by the pattern of hit strips. RPCs offer excellent time resolution, making them valuable for triggering purposes.

Performing at peak performance, about 10^9 proton collisions take place every second inside the CMS detector. The sheer volume of data produced is too massive to be stored entirely for analysis. Therefore, a triggering system is present to select events of interest based on predefined criteria. The triggering system operates in two levels: the Level-1 (L1) Trigger and the High-Level Trigger (HLT). The L1 Trigger is the first level of event selection with the primary focus of reduction of the event rate to a manageable level for further processing, operating at the bunch crossing rate of 40 MHz. The L1 trigger works by using a combination of custom hardware processors to make rapid decisions based on simplified information from calorimeters and muon detectors. The L1 Trigger can effectively reduce the event rate to around 100 kHz by selecting events that exhibit features indicative of processes of interest. The HLT on the other hand is a software-based trigger. It analyzes the full event information without simplifications applied on the L1 Trigger. It further reduces the event rate to a level suitable for permanent storage and later analysis.

CHAPTER III

EVENT SIMULATION

Replication of experimental data is needed for various reasons: it helps in understanding the detector's response, validating theoretical models, designing new analysis techniques, and optimizing detector performance. In short, event simulations allows for extracting information from the CMS detector. Both the physical processes described in Chapter 2.3 and the detector's properties can be simulated. Simulation of particle collisions are generated with Monte Carlo methods based on current theoretical models. Monte Carlo event generators such as PYTHIA [8] and HERWIG [9] are tools with the capability of simulating physics processes such as hard scattering, parton showers, hadronization and underlying event activity.

The other side of the coin regarding the event simulation is the detector simulation. Unless the generated particles are passed through a detailed simulation of the detector, the simulation only represents a theoretical interaction. GEANT4 [10] is a detector simulation software often used at the LHC experiments for its complete functionality. Using GEANT4, it is possible to simulate any setup or detector and the source of radiation. Physical processes that occur as particles travel through different detector components, including electromagnetic and hadronic interactions, decays, and energy deposits are all included. The output of a detector simulation is then digitized to produce data similar to what electronic components of the real detector would record.

3.1 Monte Carlo Simulations

Monte Carlo methods are computational algorithms that use repeated random sampling to solve problems. Monte Carlo methods start with identification of a statistical distribution and draw random samples to get a set of output values. Each output

parameter corresponds to one specific outcome in the simulation.

To start off, creation of a deterministic model which can mimic the real-world scenario as much as possible is required. Then the most likely values of input parameters, known as the *base case*, are used to transform inputs to the desired output with mathematical description of the model. This step is known as the *static model generation*. The next step is *input distribution identification*, where we identify the uncertainty due to stochastic nature of the input parameters. Underlying probability distributions are often extracted by the aid of historical data. After this step, the core of the Monte Carlo methods known as *random variable generation* comes. Random samples (or numbers) are generated by the identified statistical distribution of input variables. Sets of generated random samples correspond to possible scenario of input values. These samples are then fed into the model to transform into output values in a recursive manner to generate a collection of possible output scenarios. After a sufficient number of outputs are accumulated, it becomes possible to perform any desired analysis on these outcomes.

3.2 Simulation of Physics Processes

Monte Carlo event generators designed for HEP experiments are able to simulate all the physics processes involved in a pp collision. There are many event generators present with PYTHIA, HERWIG being *general-purpose* generators that simulate the event, as well as the hadronization process. There are also *Matrix Element* generators like the MadGraph5_aMCat@NLO [11] that deliver the event at the parton level and leave simulation of hadronization to general-purpose generators. Event generators divide the processes into a step by step sequence. Figure 6 shows this sequence.

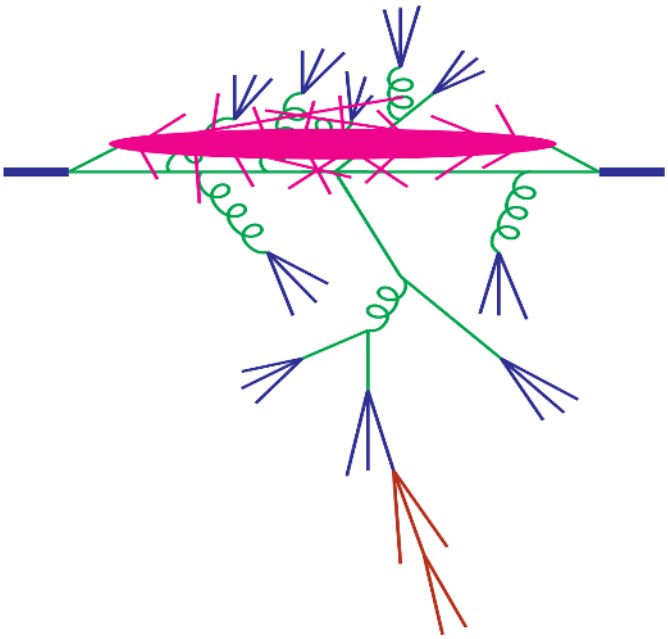


Figure 6: The structure of a pp collision. Each color corresponds to a different stage of simulation [12]. Colors of parton showers, hadronization, underlying event and unstable particle decays are respectively green, blue, purple and brown.

Firstly, the hard scattering is handled. PDFs show how the momentum and energy of a hadron are distributed among its constituent partons, so it has to be calculated. The lowest order perturbation theory gives the distribution of the outgoing partons, so it is considered straightforward to simulate the hard scattering. The evolution of incoming and outgoing partons in the hard scattering process is described by the second step in the simulation, the parton shower. This step is simulated in a sequential manner with respect to the momentum transfer scale. At the hadronization limit, where the momentum scale is at the lowest, the perturbation theory breaks down and the hadronization process starts to begin. As described in Chapter 2, the hadronization calculations are not feasible, so it is necessary to turn to the String and Cluster models to describe how partons are confined into hadrons. While PYTHIA uses the String Model, the HERWIG uses the Cluster Model. For the underlying event

the two simulation methods are *multiple-parton interaction models* and *diffraction models*. Since the underlying event usually involves soft interactions, with small momentum transfers, perturbative QCD is not applicable.

3.3 Detector Simulation

Simulating the detector is the next step of the event simulation process. The simulation is not a "realistic" picture of the event unless detectors' impact and response is added to the simulation chain. Knowing how the detector responds is the gate to understanding much needed particle-level properties. Without the detector simulation the data represented is purely theoretical.

In the context of LHC detector simulation comes after the event simulation, simulation of the passage of the produced particles through the experimental apparatus. This involves simulating the interactions with the detector material, propagation through the detector's intricate geometry, and the generation of detector signals. For the detector simulation GEANT4 is very actively used in CERN. It treats one particle at a time, in a step by step fashion. To determine the step length, geometrical boundaries and the cross sections of undergoing physics processes are used. Then, the energy deposits of the simulation get digitized to produce detector response in terms of voltage signals. Simulation capabilities of GEANT4 include ionization, bremsstrahlung, electric and magnetic fields, hadronic interactions and showers.

CHAPTER IV

EVENT RECONSTRUCTION

By converting raw detector data into meaningful measurements of particle properties, reconstruction enables precise identification and measurement of particles produced in an attempt to bring measurements back to particle-level. The *Particle Flow* (PF) algorithm [13] is used for the purpose of reconstruction and identification of all stable particles generated in an event. With a careful investigation of all CMS subdetectors' signals, it is possible to determine the track, energy and type of produced particles. Identified particles are aggregated through the application of diverse jet clustering algorithms, thereby constructing physics objects known as jets. Due to interactions within particle detectors and effects such as non uniform detector response, jets that are reconstructed from detector signals often exhibit discrepancies from the true generator-level energies and momentum of the original particles produced in collisions. *Jet energy corrections* (JEC) are a series of correction methods employed to correct these discrepancies.

4.1 Particle Flow Algorithm

The PF algorithm is implemented in the CMS experiment for the purpose of individual identification and reconstruction of all particles arising from pp collisions. The algorithm achieves this by integration of all sub detector signals present in the CMS detector. Then, the PF algorithm creates a global event description that includes all particles and their properties. An individual reconstructed particle identified within the event is known as a PF candidate. PF candidates include electrons, muons, photons, charged hadrons, and neutral hadrons. Each PF candidate represents a potential particle produced in the collision, and the algorithm accurately identifies

and reconstructs these candidates to construct higher-level physics objects, jets for example. The workflow of the PF algorithm is shown on Figure 7.

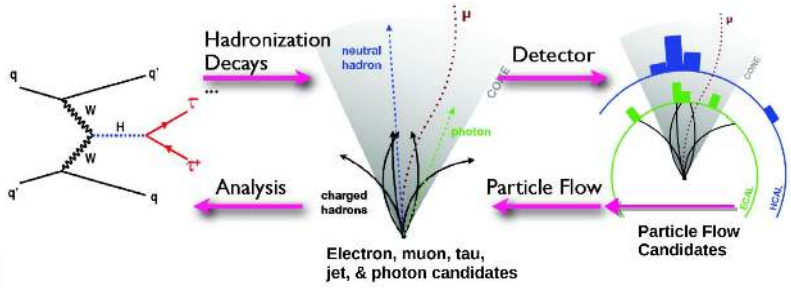


Figure 7: The diagram of the PF algorithm workflow [14]. Particle decays and final state particles are shown on the top half. These particles leave tracks and deposits in the detector. The bottom half shows how the PF candidates are extracted from detector signals and fed as an input to the PF algorithm.

4.1.1 Track Reconstruction

Reconstruction process starts with the tracks. It begins by identifying potential track candidates referred to as seeds, from clusters of hits in the inner tracking detector. Initial estimation of particle trajectories are obtained from these seeds. Once seeds are identified, track fitting is performed, where parameters are iteratively adjusted such as the track's position and momentum to best match the observed hits in the detector. The parameters that relate to position and momentum are adjusted using the *Kalman Filter* algorithm to optimize the track's agreement with observed hits. In this step overlapping tracks due to dense environments are an especially big problem. Pattern recognition techniques and iterative procedures are used for disentangling these overlapping tracks. The track's full parameters can be determined by performing a fit to the complete set of hits. The entire process of track reconstruction, including seed identification, pattern recognition, and track fitting, is repeated iteratively multiple times to find the most optimal track solution.

4.1.2 Calorimeter Clustering

Since electromagnetic and hadronic showers spread out over multiple calorimeter cells, the energy deposits must be clustered. A certain threshold is defined for potential particle interactions. These thresholds correspond to twice the standard deviations of the electronics noise in the ECAL. For the -barrel region it is roughly 80 MeV and up to 300 MeV in the end-caps. In the HCAL, the threshold is set at around 800 MeV.

After identifying calorimeter cells where the energy deposits surpass this threshold, neighboring cells are then added to the initial cluster if their energy exceeds a predefined noise level. Clustering algorithms can be used to distinguish between electromagnetic and hadronic showers. Finally, maximum likelihood estimation is utilized to construct clusters of energy deposits.

4.1.3 Linking Algorithm

A particle corresponds to several PF elements in the various subdetectors, so a linking algorithm is necessary to connect the PF elements from different sources. A linking algorithm should consider various factors such as the spatial proximity, timing information of the track and energy deposit, and also expected energy deposition patterns for different particle types. The linking procedure implemented in PF algorithm tests pairs of elements within the event. Pairings are restricted to nearest neighbors in the $\eta - \phi$ plane using a k-dimensional tree to prevent computational inefficiencies in the process. The quality of the link between two chosen elements depend on the distance metric. Direct or indirect links result in formation of *PF blocks*, associating elements based on common connections.

More specifically, the process starts by linking charged-particle tracks to calorimeter clusters. Tracks are extrapolated to the preshower, ECAL, and HCAL, with links formed if the extrapolated track position falls within the cluster area. For

bremsstrahlung photons, tangents to *Gaussian-Sum Filter* tracks are extrapolated to the ECAL to link clusters within specified η boundaries. Photon conversions are handled by linking tracks that originate from the same conversion vertex. Additionally, calorimeter clusters are linked by matching HCAL clusters to ECAL clusters and preshower clusters based on proximity. Secondary vertices indicate nuclear interactions and they are identified by linking multiple tracks to a common vertex. As the last step, tracker tracks are linked to muon detector information to form global and tracker muons. With this linking method, particles can be accurately identified and reconstructed.

4.2 Jet Clustering

Jet clustering algorithms are used to identify and reconstruct these jets and find the parton that initiated the collimated beam. Clustering particles into jets also comes with the benefit of a more precise measurement of energy and momentum of the initial parton. Such an algorithm can also aid in distinguishing between signal jets and background jets. There are some key considerations in the design of a jet clustering algorithm. The jet size and area are the factors that determine a jet's susceptibility to soft radiation, with a larger radius being important for accurately capturing the mass and energy of hadronized particles, while a smaller radius helps reduce contamination from underlying event and pile-up effects. Additionally *infrared* and *colinear* (IRC) safety are essential. An algorithm that does not satisfy colinear safety can alter the number and contents of jets when a hard particle splits. Infrared safety mitigates sensitivity to soft gluon additions that can potentially affect perturbative QCD calculations.

Among the many methods for clustering, sequential clustering algorithms are among the most commonly used. The jets used in our study are also clustered using this method. Sequential algorithms iteratively combine pairs of particles based on a

distance metric until a predefined stopping criterion is met. The sequential recombination algorithm family includes the k_t, Cambridge/Aachen and anti-k_t algorithms, which all share a similar method of clustering. The distance metric between the particles i and j is defined as:

$$d_{ij} = min(k_{t,i}^{2p}, k_{t,j}^{2p}) \frac{\Delta_{ij}^2}{R^2}$$

where $k_{t,i}^{2p}$ and $k_{t,i}^{2p}$ are the transverse momenta of particles i and j, $2p$ is a parameter that defines a particular clustering algorithm, R is a parameter that sets the characteristic size of the resulting jets and Δ_{ij}^2 is the angular separation between the particles in the $\eta - \phi$ plane, defined as:

$$\Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2.$$

The distance d_{ij} is calculated for each particle pair and the distance from the selected entity to the beam axis $d_{iB} = p_{ti}^a$ is calculated for each cluster i. If $d_{ij} = d_{iB}$, the selected cluster is considered an independent jet and removed from iteration. The steps are executed repeatedly until all clusters have been identified as jets.

The p parameter governs the importance of energy relative to other geometric scales such as Δ_{ij}^2. Different p values correspond to different jet clustering algorithms. For $p = 1$ we get the inclusive k_t algorithm with a preference for clustering soft particles with nearby particles. In the case of $p = 0$ we get the Cambridge/Aachen algorithm. This algorithm has no preference for clustering order and is considered intermediate between k_t and anti-k_t algorithms. The p = -1 case corresponds to the anti-$k_t$algorithm, which is widely used in CMS and ATLAS. The anti-k_t algorithm prioritizes clustering hard particles with softer ones by defining the distance metric as the inverse of the k_t metric. This results in a preference for merging particles with higher transverse momentum which makes it particularly suitable for reconstructing jets from high-p_T particles like those originating from the decay of heavy particles or the production of high-p_T jets.

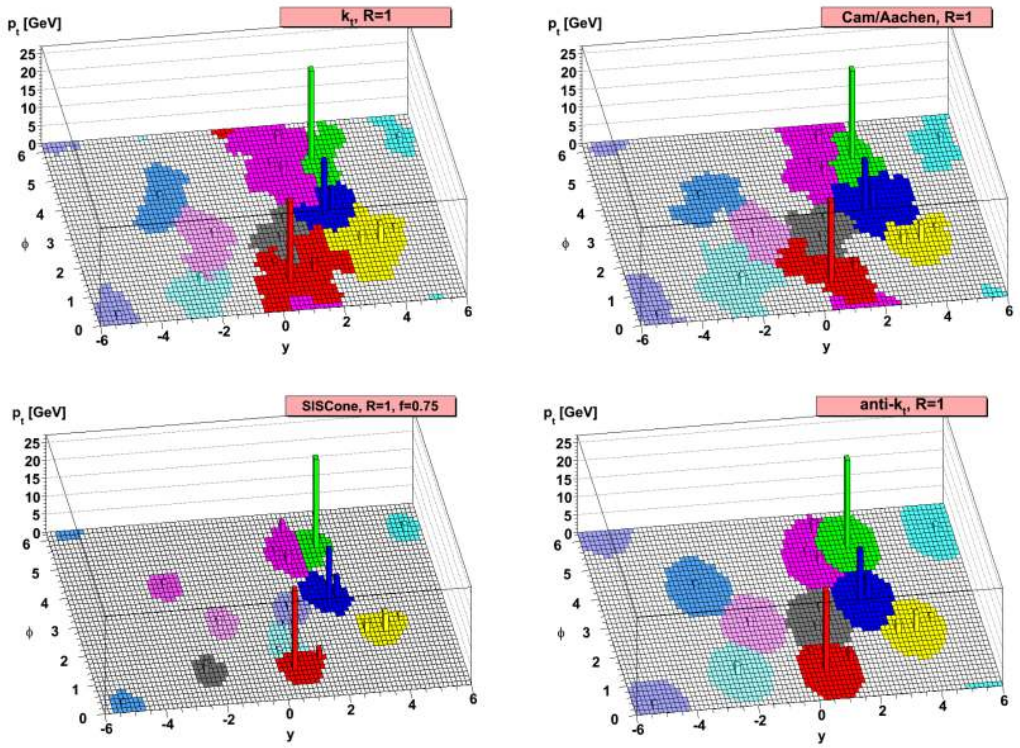


Figure 8: A parton-level event, generated with HERWIG, shows catchment areas of the clustered hard jets under different clustering algorithms [15].

Out of the three algorithms, the anti-k_t algorithm is the most popular. Apart from being infrared and IRC safe, it gives a more proper conical shape for the produced jets relative to other algorithms. The comparison can be seen in the Figure 8.

4.3 Jet Energy Corrections

There are several factors which could lead to inaccuracy in the measurement of reconstructed jet's transverse momenta, p_T. Finite energy resolution in the detector response, non uniformities in the detector material, and fluctuations in the particle showers within the calorimeters can cause a difference between p_T values of the measured and particle level jets. The intent of the JEC is to relate energy of the reconstructed jets to the energy of a final state particle jet by applying a multi-level

correction sequence step by step. Workflow of JEC can be seen in Figure 9.

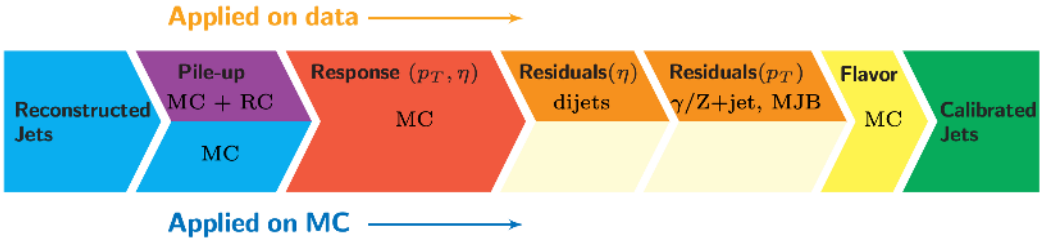


Figure 9: The CMS jet calibration workflow [16]. Aside the residual corrections that is applied to only the data, all corrections are applied to both the simulation and the data. Calibrated jets are acquired after implementation of the shown correction sequence.

4.3.1 Pileup Energy Subtraction

Additional pp collisions occurring simultaneously with the primary collision of interest is called the pileup. These additional collisions result in an increased background activity in the detector, leading to higher energy deposits in the calorimeters that can distort the reconstructed jet energies. *Charged-Hadron Subtraction* (CHS) is a standard method developed by the CMS Collaboration that can subtract the effect of pileup caused by charged particles. CHS begins by associating charged hadrons detected in the tracker with the primary collision vertex and additional pileup vertices reconstructed in the event. Each charged hadron is assigned to the vertex with which it has the smallest transverse distance, dxy, in the $r-\phi$ plane. Then, charged hadrons related to pileup vertices are subtracted from the reconstructed jets to mitigate their contribution to the jet energy. This subtraction process is generally performed by removing the four-momentum of charged hadrons associated with pileup vertices from the jet's four-momentum. Since charged particles generated in a pp collision constitute 60% percent of all particles on average, CHS can only mitigate energy of that 60% portion. A jet area based approach is used to mitigate the effects of the neutral particles in jets. With the assumption that the energy uniformly distributed inside

the detector, corrected transverse momenta becomes:

$$p_T^{corr} = p_T^{raw} - \rho.A_{jet}$$

where p_T^{raw} is the raw energy, ρ is the median p_T density and A_{jet} is the jet area, a measure of the jet's susceptibility to pileup.

This method does not work optimally for singular collisions since uniform distribution of neutral particles assumption requires averaging over multiple collisions. In the dataset we used for this thesis, the pileup contamination is mitigated using the CHS algorithm. However, recently CMS introduced *PileUp Per Particle Identification* (PUPPI) [17], a newer approach that overcomes the limitations of CHS algorithm. Since CHS is used in our data we do not go into details of the PUPPI algorithm.

4.3.2 Simulated Response Corrections

Response corrections are applied to jets that have undergone pileup subtraction. The deviation of responses between the detector itself and particle level truth is handled by the derived correction factor. These corrections are derived from multijet samples that have been reconstructed with a highly detailed model of the detector based upon GEANT4 and PYTHIA packages which provide an accurate depiction of the response. Particle-level jets are matched to reconstructed jets with the condition of radius R to be less than half of it. For an anti-k_t algorithm with $R = 0.4$, the matching condition corresponds to $R < 0.2$. In bins of particle-level transverse momenta p_T^{gen} and reconstructed η, simulated particle response is then defined as:

$$R_{particle} = \frac{\langle p_T \rangle}{\langle p_T^{gen} \rangle}$$

where angle brackets correspond to average values for p_T^{gen} and reconstructed η bins. Response corrections directly targets the detector induced response difference.

4.3.3 Residual Corrections

The next step after pileup and simulated response corrections are a series of residual corrections. Events where a reference object is balanced by a jet energy wise where no missing transverse energy is considered. This reference object should have a known p_T and can be another jet, Z boson or a photon. These corrections are derived using data from dijet events. The response of jets in different regions of the detector is compared to that of central region jets with $|\eta| < 1.3$. Residual corrections are described by two different constraints:

$$R_{balancing} = \frac{p_T^{jet}}{p_T^{ref}},$$

$$R_{mpf} = 1 + \frac{\vec{p_T}^{miss} . \vec{p_T}^{ref}}{(\vec{p_T}^{ref})^2}$$

where $R_{balancing}$ and R_{mpf} refers to the p_T-balance method and *Missing Transverse Momentum Projection Fraction* (MPF) method respectively. The p_T-balance method directly compares reconstructed jet momentum to the momentum of the reference object. The MPF balances the reference object against the whole hadronic recoil. These methods correct for biases introduced by initial- and final-state radiation (ISR and FSR) and jet energy resolution (JER) effects.

Residual corrections are divided into two categories. Firstly, relative corrections are applied to address the discrepancies in jet response as a function of jet η. These corrections are applied to make a more uniform jet response across different detector regions by comparing jets in different η regions to those in the central region $|\eta| < 1.3$. Secondly absolute corrections are applied to adjust the overall jet energy scale as a function of p_T to ensure the corrected jet energy matches the true energy across different p_T ranges. The absolute corrections are derived comparing a known reference object to the reconstructed jet p_T.

4.3.4 Jet Flavor Corrections

Under assumption that a jet has originated from a certain parton flavor, it is possible to correct the jet to the particle level. These differences are caused mainly by different amounts of jet fragmentation energy and variations in particle substructure. The neutral hadron fraction of the jet substructure is the most prominent factor in response. Jets originating from u and d quarks tend to have the highest response. Gluons, with their higher color charge undergo a softer jet fragmentation, which results in a decrease in calorimeter response due to more particles going under detector acceptance. Flavor corrections can be derived from MC simulations and from data driven samples.

CHAPTER V

ARTIFICIAL NEURAL NETWORKS

In recent years, the emergence of powerful artificial intelligence subfields such as artificial neural networks (ANNs) or machine learning in general has revolutionized not only academic research, but also industries ranging from healthcare [18] and finance [19] to entertainment and transportation. The ability of machine learning algorithms in parsing vast amounts of data, learning from them, and making predictions or decisions without being explicitly programmed for specific tasks marked a paradigm shift in how we deal with data. Uncovering patterns otherwise too complex for human perception can be easily overcome by the power of statistical analysis and adaptive learning provided by these algorithms. Alongside the advancement of parallel computing, these new methods made it possible to solve complex problems otherwise almost impossible to deal with. HEP is one of the fields that has had its share of this development. Jet tagging [20], particle reconstruction [21] and jet calibration [22] are examples among many other applications of deep learning methods.

Essentially, machine learning can be divided into three categories as *supervised learning*, *unsupervised learning* and *reinforcement learning*. In supervised learning the model learns from a labeled dataset provided with both input data and corresponding correct outputs. Artificial neural networks are an example of this. The model learns mapping from inputs to outputs and makes predictions based on the learned mapping for new, unseen data. In contrast unsupervised learning uses unlabeled data, which means the input data has no corresponding output tags and the "right answer" is not told. The algorithm explores the structure to extract hidden patterns without knowledge. Reinforcement learning involves the interaction of an

agent with its environment. The agent's actions follow a policy based on rewards and penalties. The objective is a strategy that will maximize the cumulative reward.

5.1 Deep Learning

Deep learning essentially represents a class of machine learning algorithms specifically designed to automatically learn hierarchical representations of data. These algorithms are characterized by their ability to extract complex patterns and features from large amounts of raw data. An ANN with multiple layers is known as a *deep* neural network, or DNN for short. Fully connected DNNs are sometimes referred as *Multilayer Perceptrons* (MLP). The type of the output determines the type of prediction, which can either be a *regression* or a *classification* task. While continuous values for outputs correspond to regression, discrete labels are referred to as classification.

Let $D = \{[x_1, x_2, \ldots, x_N], [y_1, y_2, \ldots, y_N]\}$ be the observations based on a set of N samples. An ANN aims to approximate a function f that satisfies the condition:

$$f(x_i) = y_i$$

for each input x_i. The input layer of the neural network consists of N neurons, each corresponding to an element of the input vector $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^N$. Elements of X are denoted as $x_1, x_2, ..., x_N$ and represent a feature of the problem. The hidden layers of the network use the input data to extract hierarchical representations and capture complex patterns to make a prediction $\hat{y}$. These layers are interconnected through weighted transitions that can be represented by the multiplication of a weight matrix $\mathbf{W}$ with a vector corresponding to the output from the previous layer. The number of units in each layer determines the dimensionality, or *depth* of that layer.

5.1.1 Neurons

Artificial neurons are fundamental units of computation in ANNs. They are analogous to the neurons in a biological brain. Each neuron in an ANN processes input data,

applies a transformation, and passes its output to subsequent neurons. If L denotes the total number of layers in the network, for each hidden layer l, the operation of a chosen neuron is defined as:

$$\mathbf{z}^l = \mathbf{W}^l.\mathbf{a}^{l-1} + \mathbf{b}^l$$

where $\mathbf{W}^l$ represents the weight matrix of size $n^l_{out} \times n^{l-1}_{in}$ for layer l, $\mathbf{b}^l$ is the bias vector of size n^l_{out}, $\mathbf{a}^{l-1}$ is the activation vector from the previous layer. Weighted sum of the inputs for layer l is denoted as $\mathbf{z}^l$. $\mathbf{z}^l$ becomes the input of a function called an *activation function* to represent activation of the next layer, $\mathbf{a}^l$:

$$\mathbf{a}^l = g(\mathbf{z}^l)$$

where $g(\cdot)$ is the element-wise applied activation function. Activation functions introduce non-linearity to the network so that more complex decision boundaries are captured. Without activation functions, the network, no matter how many layers are in it, would essentially be a linear regression model. Hence, there are many different activation functions in use with different strengths and weaknesses.

5.1.2 The Learning Process

Training data with $[x_1, x_2, \ldots, x_N]$ as the input and $[y_1, y_2, \ldots, y_N]$ as the output is sampled and given into the model during the training phase of a neural network with the objective to make a prediction $\hat{y}$, by using the input $\boldsymbol{X}$. The difference between the predicted output $\hat{y}$ and the actual output y is quantified by some *loss function* L. The next step is to update weights and biases to minimize the loss function. An optimization algorithm like *gradient descent* iteratively updates the parameters based on the gradients of the loss function with respect to the parameters with:

$$\mathbf{W}_{new} = \mathbf{W}_{old} - \alpha \frac{\partial L}{\partial \mathbf{W}_{old}}$$

$$\mathbf{b}_{new} = \mathbf{b}_{old} - \alpha \frac{\partial L}{\partial \mathbf{b}_{old}}$$

where α is a hyperparameter that controls the step size of the parameter updates and is called learning rate. The learning rate needs to be adjusted very carefully, since a too high learning rate can lead the training process to overshoot the global minima, while a low learning rate can result in a very slow convergence and there is a risk that training will stall or stop improving significantly, as updates may not be enough to reach the global minimum or escape the local minimum. Figure 10 shows the gradient algorithm moving towards the global minimum.

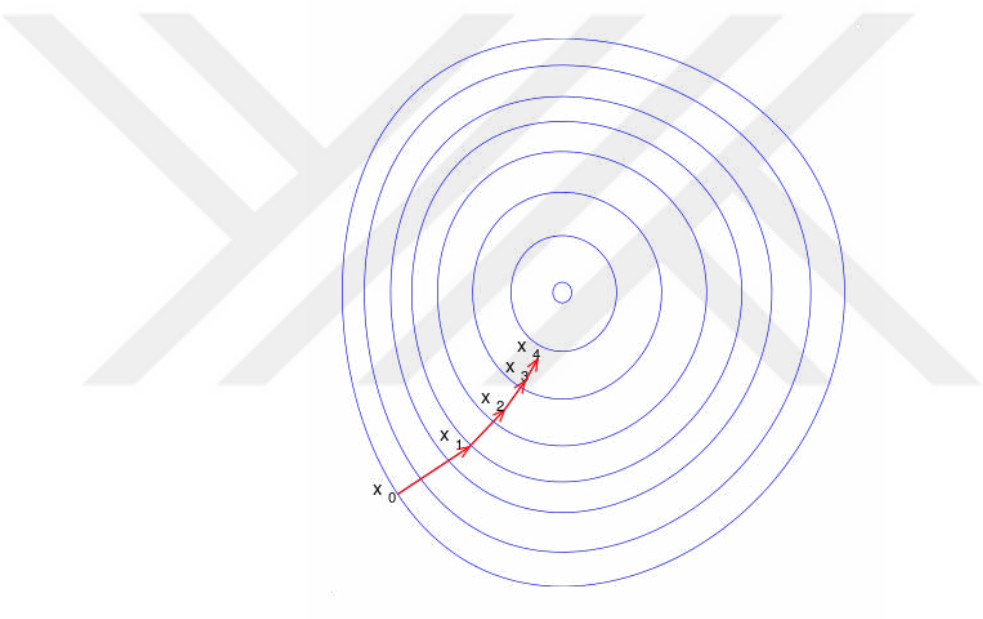


Figure 10: Illustration of gradient descent algorithm moving towards the global minimum on level sets [23].

Calculating the gradient of the loss function with respect to the weights directly requires an exponential number of operations since the number of required operations would scale exponentially with the number of edges in the network's computational graph. For larger networks it simply becomes unfeasible to use such a method. But the use of chain rule and calculating the gradient one layer at a time, then iterating backwards from the output layer to the input layer allows us to avoid such redundant calculations. This algorithm is called *backpropagation* and it was first published by

S. Linnainmaa in 1970 [24] but it was P. Werbos [25, 26] who applied it to neural networks in the way that has become standard.

An MLP is a type of feedforward network that consists of fully connected nodes as seen in Figure 11.

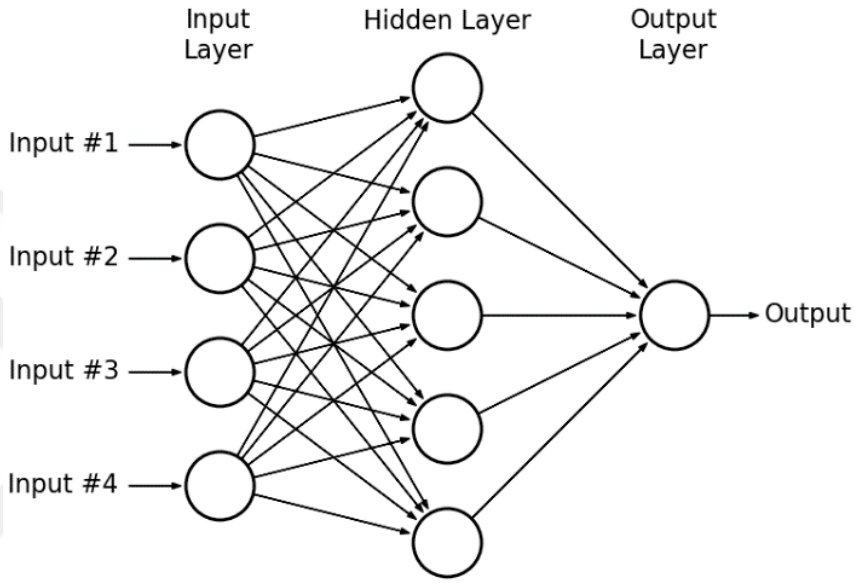


Figure 11: Representation of a multilayer perceptron with one hidden layer [27].

During the forward propagation phase, the input is transformed by the linear combination of weights and biases in the network. An application of an activation function follows. The network's prediction is then provided by the final layer output. MLPs are *universal approximators*. At sufficient depth and number of neurons an MLP can approximate any function. This is known *Universal Approximation Theorem*. But compared to other machine learning algorithms, selection of appropriate hyperparameters can be a formidable task.

Most ANN models are designed to work with fixed-dimensional vectors as the input data. In such a case the order of the elements is directly linked to the output through the activations. For example, the Recurrent Neural Networks (RNNs) are specifically designed for sequential data, hence the importance of input element order

is everything. One example is natural language processing, where different permutations of words can manifest different meanings. However, many real-world problems involve data in the form of sets where the order of elements is irrelevant, and the models must be invariant to permutations of the input elements.

Recently a new approach, *Deep Sets,* has surfaced to overcome this problem [28]. The framework handles sets as inputs, which can ensure the *invariance* or *equivariance* to permutations of the set elements. A function f is permutation invariant if its output does not change when the order of the input set elements is permuted. For a set $X = [x_1, x_2, ..., x_N]$ and any permutation π, the function is invariant if:

$$f(X) = f(\{x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(N)}\}).$$

And f is permutation equivariant if permuting the input set elements results in a corresponding permutation of the output:

$$f([x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(N)}]) = [f_{\pi(1)}(x), f_{\pi(2)}(x), \ldots, f_{\pi(N)}(x)].$$

The main theorem in the study states that any permutation invariant function f operating on a set X with elements from a countable universe for suitable ϕ and ρ transformations can be decomposed in the form:

$$f(X) = \rho\left(\sum_{x \in X} \phi(x)\right).$$

Each element x in the input set X is first transformed by ϕ. The transformed elements are then aggregated using a permutation-invariant function, which can be sum, mean, max or min of the transformed elements depending on the preference. ρ is another neural network, usually an MLP that takes the aggregated information and transforms it to produce the final output.

5.2 Common Methods and Algorithms

5.2.1 Activation Functions

Activation functions are essential for bringing nonlinearity to the table and approximation of nonlinear continuous functions. Without the activation functions, an ANN only produces the output as a linear function of inputs, even if the depth and width (number of neurons) are large. There are many examples of different activation functions, but for this thesis we only use and consider the *Rectified Linear Unit* (ReLU) which is expressed as:

$$\mathrm{ReLU}(x) = \max(0, x).$$

ReLU comes with the advantage of sparse activations. For a randomly initialized network, many of the network units are not active since initializing the weights to zero is a common practice. Vanishing gradient problems are also better handled by ReLU compared to a function that saturates both directions (e.g Sigmoid Function) [29].

5.2.2 Loss Functions

A loss function is the measurement of the model's prediction error. So the selection of it largely depends on the specific requirements of the task at hand. For a regression problem the common choices are the *Mean Absolute Error* (MAE) and the *Mean Squared Error* (MSE). During the gradient descent, the new parameters are determined by the gradient of the loss function with respect to the old parameters, so the absolute necessity of continuity, differentiability and well defined gradients must be said. In this thesis, MAE is chosen as a loss function and this choice will be discussed in Chapter 6.

5.2.3 Hyperparameters

The characteristic configuration settings used to control the behavior of the training algorithm and the structure of the model itself are called the *hyperparameters*. Hyperparameters are not adapted by the algorithm, thus need to be determined before the training. Since performance of the algorithm largely depends on the tuning of these parameters generally, they should be chosen cautiously.

Architecture of an ANN is defined by the number of nodes in a layer, number of layers and the connections in between. Model hyperparameters encompass the overall architecture, type of activation functions used, and regularization methods. Too few layers or neurons might lead to an instance called *underfitting*, if the model is too simple to capture the underlying patterns, on the other side too many can cause *overfitting*, where the model complexity is too high and noise is captured instead of the signal.

Another type of hyperparameters are algorithm based; batch size, learning rate and number of epochs can be given as an example. Batch size is the number of training samples processed before the parameters are updated. Here, a tradeoff is involved since a smaller batch size generally introduces more noise into the gradient, which can help with generalization and act as a regularizer but it leads to slower training per epoch. Larger batch sizes improve the accuracy of gradient estimates and lead to faster convergence, but increases the memory usage and may generalize poorer and overfit the data.

Learning rate determines the size of the steps that an optimizer algorithm takes to reach the minimum of the loss function. A high learning rate can aid for the model to converge faster with the risk of overshooting the global minimum. A low learning rate on the other hand, can lead to more stable convergence but it can take a long time and get stuck in local minima.

An epoch is one complete loop through the entire training dataset. The choice

of number of epochs should be sufficient for the model to learn effectively. But too many epochs can lead to overfitting since after some training, the model can also potentially learn from the noise and details that do not generalize well in the data.

5.2.4 Optimization Algorithms

The update of the weights and biases is achieved by an optimizer. Relatively recently *Adam* (Adaptive moment estimation) optimizer [30] has gained popularity. In this thesis the Adam was the choice of optimizer, so other optimizers are not going to be explained. Adam defines two moving averages for each parameter: the *first moment*, which is the average of the gradients and the *second moment*, which is the average of squared gradients. These moments are initialized to zero. At each time step t, the gradient (g_t) of the loss function with respect to the parameters is computed. Then, the first moment estimate m_t and the second moment estimate v_t is updated according to:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where β_1 and β_2 are tunable hyperparameters that control the exponential decay rates. They influence how much weight is given to the past gradients versus the current gradient in the calculation of the moving averages. Initialization of the moments could introduce bias. Bias-corrected moments are computed with:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

Then, the parameters are updated according to:

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where θ represents the parameters and ϵ serves as a small constant that prevents division by zero. Adam is adaptive in the way that the learning rate for each parameter

is individually based on moment estimates. Due to this adaptive nature, Adam is a fast converging optimizer.

5.2.5 Batch Normalization

Batch normalization (BatchNorm) [31] normalizes the inputs of each layer so that they have a mean μ_B of zero and a standard deviation σ_B^2 of one. This normalization practice ensures that the input to each layer remains approximately in the same distribution, regardless of changes in the distribution of earlier layers' outputs. For x values over a mini-batch $B = \{x_1, x_2, \ldots, x_M\}$, BatchNorm performs:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta.$$

Here ϵ is a constant added to denominator to prevent division by zero. γ and β are learned parameters that scale and shift the normalized values.

In an ANN, each layer receives inputs from the previous layer and as the training goes, the weights of these layers are updated to minimize the loss. These updates change the distribution of outputs from the previous layers, which become the inputs to the next layers. This continuous change in input distribution which is known as *internal covariate shift* can slow down the training and impact convergence. Batch normalization was initially proposed to deal with this phenomenon. By normalizing the inputs it is possible to achieve a faster and more stable convergence. Additionally, since statistics of μ_B and σ_B^2 differ for each mini-batch, they introduce a form of noise into the activations of the network. This noise can have a regularizing effect and in turn, eliminating relying heavily on any single activation pattern which can improve generalization capabilities.

5.2.6 Dropout

Dropout [32] is a regularization method developed to prevent overfitting. The approach is simple in nature, as it randomly deactivates a proportion of neurons in the network during each training iteration. Figure 12 represents dropout method visually.

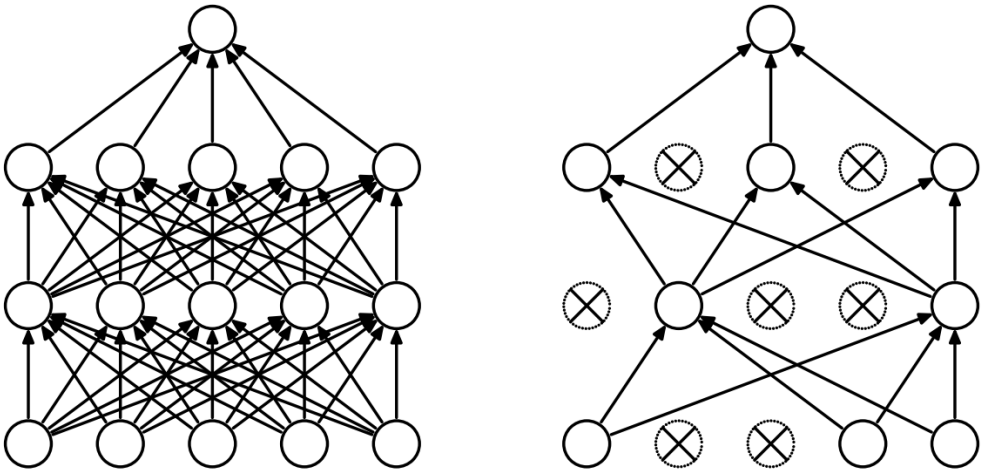


Figure 12: A visual representation of dropout [32]. On the left an ANN with 2 hidden layers is shown. On the right, same network with dropout is presented. Crossed nodes are the dropped units.

The probability to set the output of each neuron to zero is called the *dropout rate* and is usually between 0.2 and 0.5. This random dropping of units can prevent over dependence of the model on specific neurons. As a result, the model is forced to learn more robust features and generalize better to unseen data.

CHAPTER VI

JET ENERGY REGRESSION

With machine learning techniques becoming widespread in the HEP community, common problems in the field such as b tagging, jet reconstruction and calibration have become a target of implementation. Improving the jet energy measurements in the CMS experiment is an especially important task. Reconstructed jet energy is affected by many factors including finite resolution of the detector, pileup effect and undetected weakly interacting particles. A step by step correction sequence is applied to jets in the reconstruction process, which we explained earlier. But the jet energy corrections can be further improved using deep learning methods. In 2020, a study by the CMS Collaboration showed that a feed forward neural network can provide significant enhancement in the energy resolution for b jets [33]. In this work, our goal is to bring the reconstructed jet momenta p_T^{reco} to generator level jet momenta p_T^{gen} using a set of jet features as the training input. Such a task is considered a regression problem, and the aim is to predict a continuous variable, the jet p_T.

6.1 Dataset

The particle jet samples in the dataset we used are generated by MC methods with PYTHIA8 event generator from simulated pp collisions at $s^{\frac{1}{2}}$= 13 Tev and derived by the CMS OpenData effort [34] for jet related studies. A simulation of the CMS detector is utilized for the emerging particles. The reconstruction process involved using the signals from the detector simulation with the PF algorithm. For each collision event PF candidates are clustered using the anti-k_T algorithm with radius parameter $R = 0.4$ from final-state particles that occur from the parton hadronization process. The reconstructed jet p_T values were corrected with the standard jet corrections. For

pileup mitigation the CHS algorithm was employed. The dataset contains variables of jets on a high-level, particle-level and generator-level jet by jet basis for 22 million jets in total. The distribution of p_T^{gen} with respect to η^{gen} and number of jets with corresponding p_T^{gen} value can be seen in Figure 13.

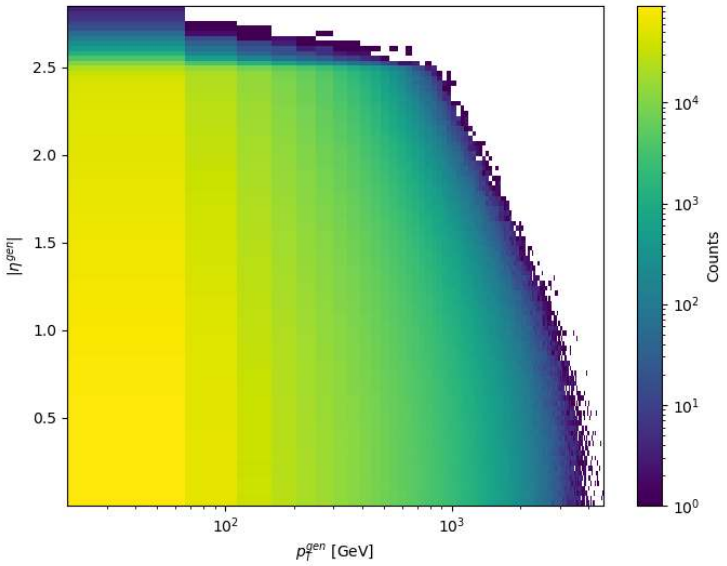


Figure 13: Distribution of the jets in the dataset with respect to p_T^{gen} and $|\eta^{gen}|$ as a heatmap.

Jets are made of multiple particles. The dataset includes global jet properties, as well as information on jet constituents called the PF candidates. The global jet input features we used to train the models are listed in the Table 1. The logarithm of reconstructed p_T is taken to reduce the width of the distribution. Apart from the p_T, angular coordinates η and ϕ, and the jet mass and catchment area are also present in our list of input features.

Table 1: Overview of jet-level features used for training. Jet-level features describe a jet's overall properties.

Feature	Description
$\log p_T$	Log transformed jet p_T
η	Jet pseudorapidity
ϕ	Jet azimuthal angle
m	Jet mass
A	Jet catchment area
$p_T D$	Fragmentation distribution of a jet's p_T
σ_2	Jet minor ellipse axis
multiplicity	Jet constituent multiplicity

The jet fragmentation function is expressed as:

$$p_T D = \sqrt{\frac{\sum_i p_{\mathrm{T},i}^2}{\left(\sum_i p_{\mathrm{T},i}\right)^2}}.$$

The jet fragmentation function describes how the momentum of a jet is distributed among its constituent particles and it can help to distinguish between the gluon and quark initiated jets. Gluons have a softer fragmentation compared to quarks, hence it is likely that quark initiated jets with hard constituents carry the majority of the jets energy. In the case of a jet made of only one particle that holds all the momentum, $p_T D = 1$, on the contrary if the jet is made of an infinite number of particles, then $p_T D = 0$.

Another useful property of the jets is the minor ellipse axis of a jet. The minor axis σ_2 describes the spatial distribution of the jets. The anti-k_T algorithm forms the jets into conical shapes. It is a sound approach to approximate a jet as an ellipse and then, the minor axis represents the direction in which the jet is least extended in $(\eta - \phi)$ plane. A more collimated (or "narrow") jet comes with a shorter minor axis,

which means that constituents are tightly clustered around the jet axis.

Jet catchment area is the effective area in the detector that the incoming jet occupies in the $(\eta - \phi)$ plane. It is used in the standard correction procedures to mitigate pileup contamination and deep learning algorithms can benefit from it too since it provides additional context about the jet's spatial extent.

Jet constituent (PF candidate) features are also included in the training data. It is beneficial to include PF candidates in our task since it describes the inner structure of the jets. Each jet constituent has five associated features that can be seen in the Table 2.

Table 2: Table of PF features that describe jet's inner structure.

Feature	**Description**
$\log p_{T_i}$	Log transformed particle p_T
η_i	Particle pseudorapidity
ϕ_i	Particle azimuthal angle
θ_i	Particle polar angle
R_i	Particle's distance to the jet center

PF candidate features are; the log transformed p_{T_i}, with i index that represents ith candidate, the η_i , ϕ_i and θ_i spatial features and the distance R_i from the particle to the center of the jet.

6.1.1 Preprocessing

Very low p_T regions in the dataset can potentially handicap the learning process since pileup intensity in these regions are very high. Also the forward region in the detector suffers from a lack of tracking information which can reduce the reconstruction quality of the jets. These issues can be handled by pruning the dataset where $p_T^{gen} > 20$ and $|\eta^{gen}| < 2.5$.

Since the jet constituent numbers are not homogeneous across the dataset and vary with accordance to each jet, a padding procedure is applied. PF candidate features are limited to jets with the number of constituents $PF_n < 100$ to reduce the dataset size and make padding process more manageable. This limit is chosen based on the available computational resources and distribution of the number of constituents in the dataset. The distribution of jets and jet constituent numbers prior to constituent number cut with respect to p_T^{gen} is shown in Figure 14.

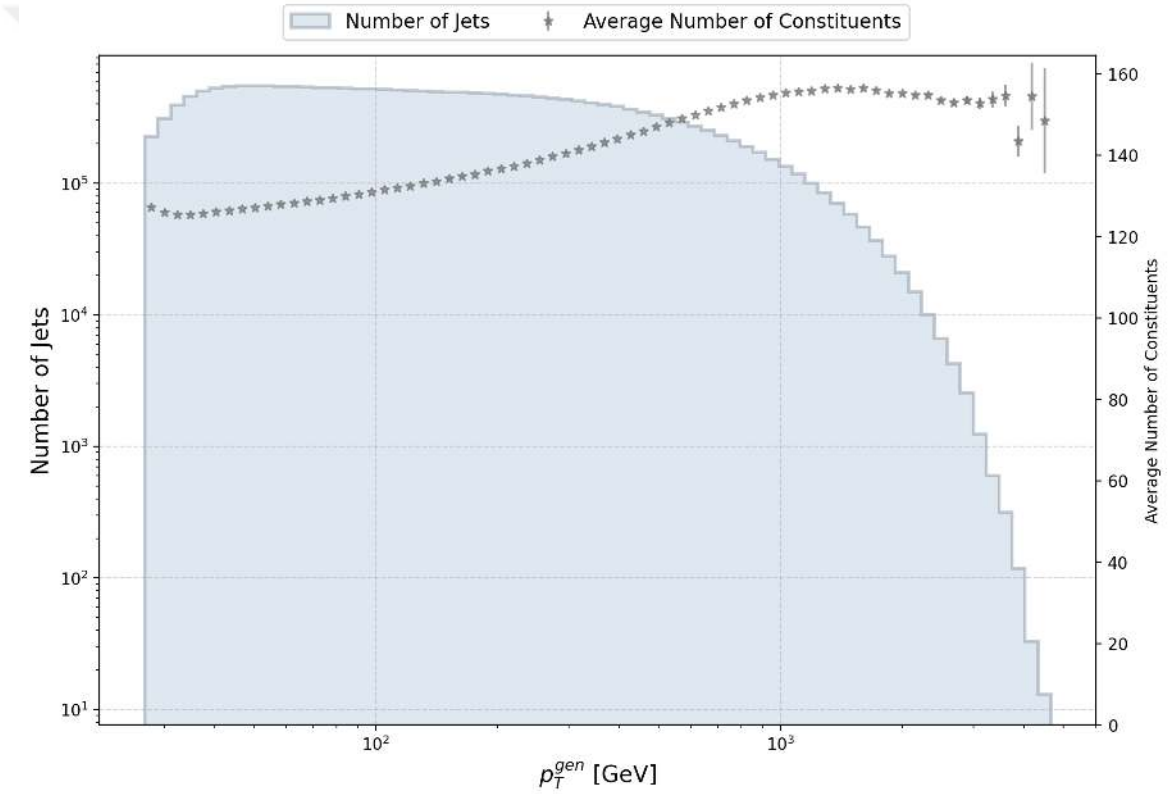


Figure 14: Distribution of jets and the corresponding number of constituents with respect to generator-level p_T. The primer y-axis represents number of jets logarithmically. While high p_T region is less populated, the average number of jet constituents is higher.

The number of jets show a decreasing exponential distribution while the average number of constituents increasing trend with respect to p_T. Smaller jet sample sizes

also lead to higher variability and larger standard errors in high p_T regions. The dataset after the cuts is reduced to 5.2 million jets which are divided into training, validation and test sets with a ratio of 6:2:2.

Since the input features scale in different ranges, it is beneficial to normalize and standardize the distribution. The *Standard Scaler* transforms the data such that the mean of each feature becomes 0 and the standard deviation becomes 1. One of the main benefits of this kind of standardization is to ensure that the gradient steps are more uniform, which can lead to a faster convergence. Optimization algorithms can also perform better since the loss function contours become more spherical after the feature scaling.

6.2 Target and Loss Function

In the regression task we laid out, the goal is to predict the p_T^{gen} from a set of input features we listed in the previous section 6.1. But as we can see in the Figure 13, p_T^{gen} is an exponentially decreasing distribution with a sparse range in magnitude and variance. In order to address this issue we define a new regression target, we logarithmically transform $\frac{p_T^{gen}}{p_T^{reco}}$ to get a nicer target distribution, which is centralized around zero as seen in Figure 15.

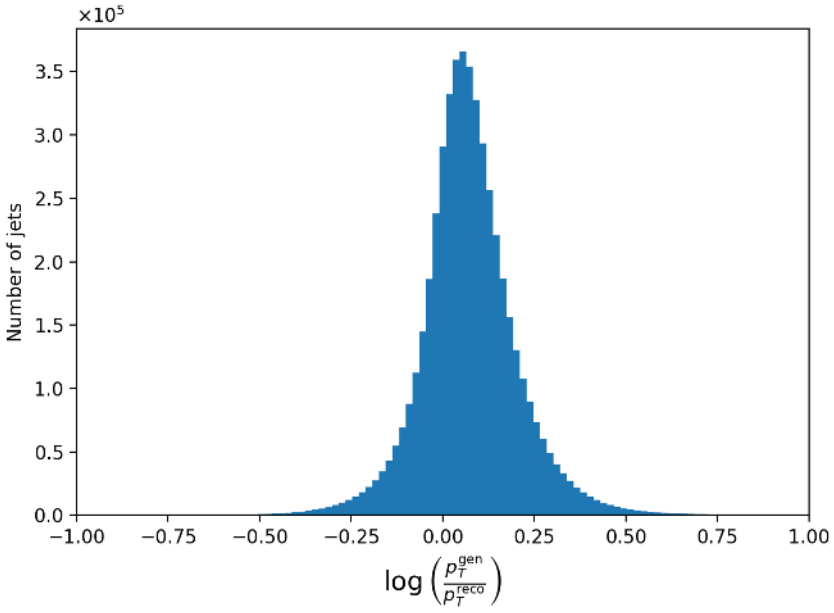


Figure 15: Log transformed regression target distribution.

The regression target then becomes:

$$y = \log\left(\frac{p_T^{gen}}{p_T^{reco}}\right).$$

Application of the exponential function to the model's prediction gives us the correction factor. This correction factor can later be multiplied by p_T^{reco} to get the corrected transverse momenta p_T^{corr}.

The MAE, also known as L1 loss, was chosen as the loss function in this thesis. The MAE measures the average magnitude of the errors between predicted values and actual values. Additionally, poorly reconstructed jets with absolute target value smaller than 1 are not considered. Hence the loss function becomes:

$$L = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| I_{|y_i|<1}.$$

One advantage of the MAE is that it treats all errors equally in terms of the absolute magnitude. This property makes the MAE more robust to outliers. On the contrary, a loss function like MSE heavily penalizes large errors by squaring them.

Inconsistencies of MSE potentially makes it an inappropriate indicator of average error [35]. The search for a loss function that is less sensitive to outliers is inspired by the study done by the CMS Collaboration on energy resolution for b jets [33].

6.3 Models

In this section we describe the models we used for the jet energy regression task. We have two models, a fully connected MLP and Particle Flow Network (PFN), a Deep Sets based model emerged for particle physics implementations [36]. The performance comparison of the models is presented in the Chapter 7. The comparison of these two models can be interesting since the fully connected setup takes all the jet-level features and PF candidate features as input altogether, while the PFN maps PF candidate features through a Deep Sets block, then concatenates the jet-level features to form an input for the MLP block. This contrast can show how effective the mapping of PF candidate features separately is in comparison to a more traditional approach.

6.3.1 Fully Connected Deep Neural Network

The fully connected deep neural networks, also referred as MLPs, take a fixed input vector $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^d$ and d is the dimensionality. For L number of hidden layers, the network's operation is:

$$\mathbf{a}^{(0)} = \mathbf{x}$$

$$\mathbf{a}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right)$$

$$\mathbf{y} = \mathbf{a}^{(L)} = \mathbf{W}^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)}$$

where $\mathbf{y}$ is the output vector, or the *prediction* of the model. The model is fully connected in a sense that, every neuron in one layer is connected to every neuron in the subsequent layer.

The architecture of the implemented MLP model boasts 6 hidden layers and with

units of (512, 256, 128, 64, 128, 256) with a bottleneck in the fifth layer. The architecture can be seen in Figure 16.

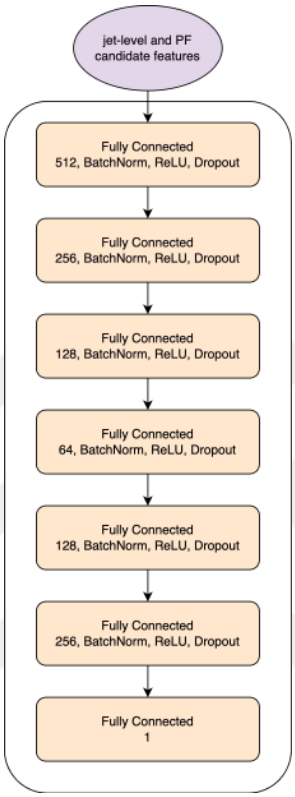


Figure 16: Illustration of the fully connected model architecture.

Bottleneck layers can potentially force the network to compress information to form a more abstract representation of the input data. This abstraction can lead to better generalization and reduce noise. Each layer includes BatchNorm, dropout and ReLU. And the last layer with 1 unit for mapping to the regression target.

6.3.2 Particle Flow Network

Originally PFN was designed for jet tagging purposes, but can be tweaked for regression as well. The PFN architecture is based on the Deep Sets framework to work with variable-length unordered sets as model input. For an input set of $X = \{x_1, x_2, \ldots, x_N\}$, where $\mathbf{X} \in \mathbb{R}^d$, x_i represents the feature vector for the ith particle, and N is the total number of particles in the set. A ϕ function maps the input sets

into a learned representation, called latent features h:

$$h_i = \phi(x_i).$$

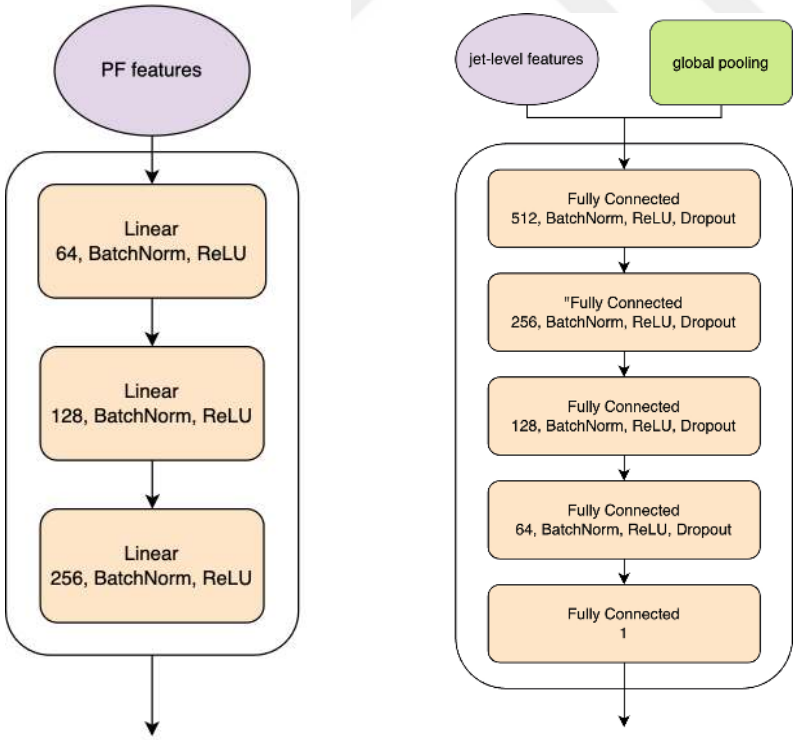
The transformed representations $\{h_1, h_2, \ldots, h_n\}$ are then aggregated using a permutation-invariant operation. Summation was chosen as the global pooling method in this thesis. The aggregation of features can be expressed as:

$$H = \sum_i h_i.$$

And the network's complete operation can be described as the function:

$$f(X) = \rho\left(\sum_{i \in V} \phi(x_i)\right)$$

where ρ is the global network that takes the pooled representation and maps it to the final output, and usually is an MLP. The PFN architecture as implemented in this thesis can be seen in Figure 17.



(a) Deep Sets block. **(b)** Fully connected block.

Figure 17: Illustration of the PFN architecture as implemented in the thesis.

A Deep sets block with units (64, 128, 256) takes the PF candidate features as the input. Each layer utilizes BatchNorm and each layer ends with ReLU activation functions. The outputs are aggregated using global sum pooling and concatenated with jet level features to form the input of the ρ, the MLP block consisting of units (512, 256, 128, 64) with BatchNorm, dropout and ReLU again. Finally, an output layer with 1 unit to predict the regression target.

6.4 Training and Implementation

Both models were implemented in the PyTorch framework [37]. PyTorch provides a robust tensor library that supports a wide array of matrix operations much used for deep learning tasks. Many useful libraries for ML related operations and algorithms are available. PyTorch also supports GPU acceleration. Through utilization of GPU's, intensive matrix operations performed in deep learning tasks are handled in much better performance compared to a CPU. In the thesis, the training of the networks were accomplished with a powerful setup that has an Intel(R) Core(TM) i9-10980XE CPU with 18 cores supporting clock speeds up to 3.00GHz with a 128 GB RAM. Tensor operations were handled in parallel using two Nvidia Quadro RTX 5000 GPU cards through CUDA support.

In the training of neural networks, an epoch is one complete pass through the entire training dataset. During an epoch, the network is trained by batches, subsets of the training dataset with predetermined number of samples (512 in our case) from the dataset. Passing an entire dataset through the network at once is a computationally expensive and memory-intensive practice, hence usually data is divided into smaller batches. Both models use Adam optimizer since it is well suited for training with batches due to its adaptive nature.

Validation loss is also computed in each epoch to infer the model's generalization performance and detect any potential overfitting. Both models were trained for 100

epochs with an early stopping mechanism, where the best validation loss is updated for each epoch on the training loop and a determined number of consecutive epochs -10 in our case- without improvement in validation loss stops the training loop. Early stopping assumes the model's learning capabilities are saturated at that point and training does not come with any benefits for any longer. Overfitting is a common problem in networks where the model's performance in validation loss starts to get worse after an initial plateau. Early stopping is a common mechanism used to prevent this scenario.

CHAPTER VII

RESULTS

For comparison of deep learning models in terms of performance, many metrics should be considered. Each evaluation metric should give insight on performance from a different perspective. First of all, the model complexity and computational time, which is called the inference time, is evaluated. The second metric is the mean energy response. Third, and arguably the most important one is the improvement in jet-energy resolution.

7.1 Model Complexity

Model complexity is a metric that is affected by factors such as the model architecture, number of trainable parameters and the number and type of operations required for a forward pass. Larger models generally have a higher complexity and it takes a longer time for a model to process an input and produce an output. Inference time is defined as the time it takes for a model to make predictions for one set of inputs. The Table 3 shows the number of parameters and inference time of the neural networks.

Table 3: Complexity table of the models.

	Parameters	**Inference time** (ms)
PFN	352769	5.636
FC	477377	4.091

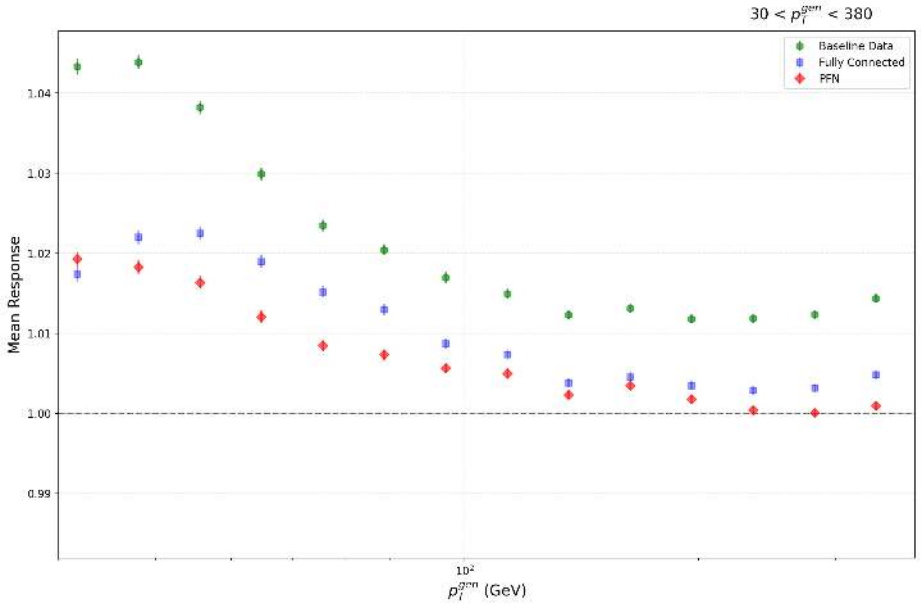
In terms of model parameters, the FC setup is a larger model due to its architecture, which includes more hidden layers with sizes (512, 256, 128, 64, 128, 256) compared to the PFN's MLP block with sizes (512, 256, 128, 64). Inference time

is calculated for each testing batch and then averaged for the whole test set after the training loop. The FC model is slightly faster than the PFN although having a higher complexity, potentially due to the additional computations involved in processing. The PFN is architecture handles variable-length unordered sets, PF features through its Deep Sets block, which can contribute to a higher inference time.

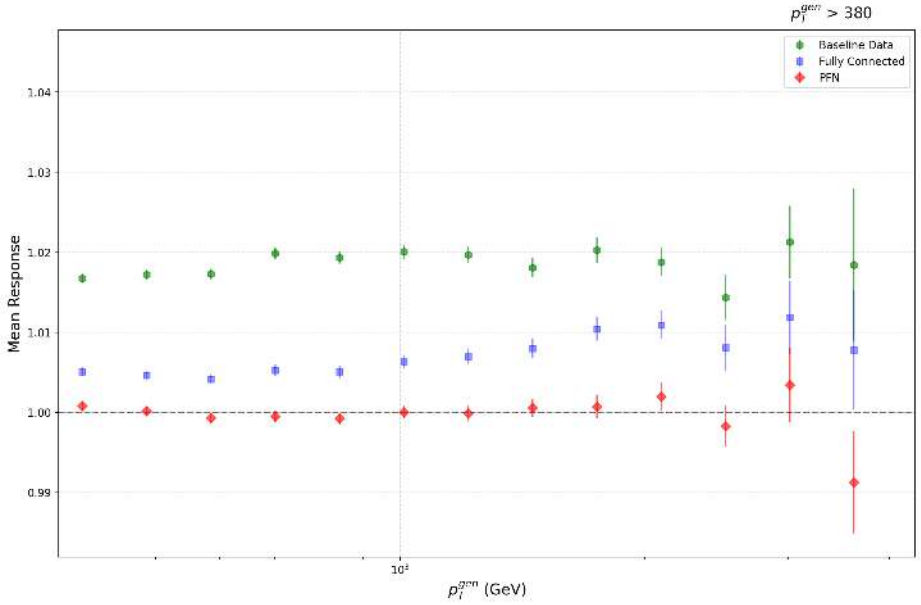
7.2 Mean Response

The mean energy response is defined as $R = \frac{p_T^{corr}}{p_T^{gen}}$. The corrected momentum p_T^{corr} can be extracted after the exponential function is applied to the predictions and the resulting correction factor can be multiplied by p_T^{reco}.

Accuracy is a term usually used in classification problems, but in the context of regression it can be interpreted as closeness of the predicted values to the true values. The mean response should ideally be 1, in this sense accuracy can explain how close the corrected p_T values are, on average, to the generator-level p_T values. Results of mean response of the models are shown on Figure 18.



(a) Mean response for $30 < p_T < 380$ region.



(b) Mean response for $p_T > 380$ region.

Figure 18: Mean response of the models in logarithmic p_T^{gen} bins. Green data points represent response of the standard jet energy corrections. The graph is split into two parts for the regions $30 < p_T < 400$ and $p_T > 400$.

Error bars in the Figure are standard errors of the mean values calculated with

statistical bootstrapping method. The response is resampled 10000 times to generate multiple samples that mimic the original data. In terms of response, both models performed above standard jet energy corrections in almost every bin. PFN performed better than the FC setup in almost every bin. For the very high p_T range, the PFNs performance deteriorates more than the FC setup. This could be attributed to the lower number of high p_T jets in the dataset as presented in Figure 14 earlier. While the FC network takes all the input features together, the PFN aggregates the mapping of the PF candidate features through a Deep Sets block. Hence, mapping process of the PFN might be influenced more from low populated areas in comparison to a more simple approach of using all features as the input for one network.

7.3 Jet Energy Resolution

Relative jet energy resolution is a metric derived from response and can give more insight on evaluation of performance. The definition used in this thesis is interquartile range (IQR) of response divided by median response:

$$\bar{s} = \frac{R_{75\%} - R_{25\%}}{R_{50\%}}.$$

IQR is a measure of statistical dispersion and median measures the central tendency of the data. Both the IQR and median are robust measures. Exclusion of the highest and lowest 25% of the data makes IQR less sensitive to extreme values. And median provides a better central location in the presence of skewed data or outliers compared to the mean since it represents the middle value of the data. Figure 19 presents the relative resolution of the models.

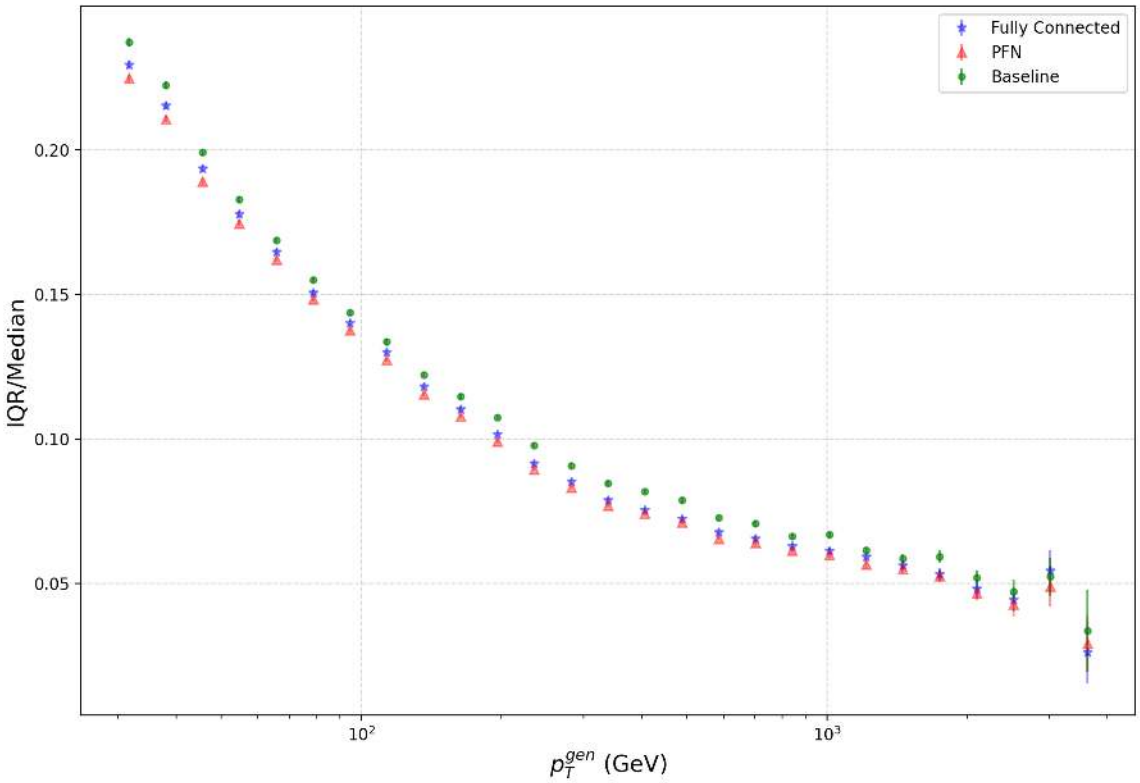


Figure 19: Relative energy resolution with respect to p_T^{gen}, logarithmically binned. Green data points show performance of the standard jet energy corrections.

Relative resolution can be seen as analogous to *precision* in classification tasks, both concepts are concerned with the spread or variability of predictions. By definition, a smaller value for relative resolution indicates tighter clustering of the data points which can be interpreted as a higher precision. PFN performs better than the FC network in almost every bin in terms of relative resolution as seen in Figure 19.

To further evaluate the relative resolution it is beneficial to calculate the improvement in relative resolution, denoted by β. Which can be calculated with:

$$\beta = 1 - \frac{\bar{s}_{\text{Model}}}{\bar{s}_{\text{Standard}}}.$$

Improvement in relative resolution indicates how much better the model's resolution is compared to the baseline resolution. β was calculated in several p_T intervals as can be seen in Table 4.

Table 4: Comparison of improvement in relative resolution produced by PFN and FC setup.

Interval (GeV)	**Model**	β [%]
$30 < p_T < 100$	PFN	**4.82**
	FC	3.19
$100 < p_T < 300$	PFN	**6.58**
	FC	4.57
$300 < p_T < 800$	PFN	**9.03**
	FC	7.16
$800 < p_T < 1200$	PFN	**8.06**
	FC	5.41
$p_T > 1200$	PFN	**8.96**
	FC	5.48

While both models perform better than standard jet energy corrections, improvements largely depend on the p_T region. Largest improvements are seen in $300 < p_T < 800$ region as Table 4 shows. This can be influenced by several factors. Firstly, jets with high p_T generally result from high-energy interactions. These interactions cause the substructure of the jet to become more complex. Secondly, at high p_T, partons are more likely to undergo hard splittings. Hence, jets are formed with a larger number of constituent particles within. A higher substructure complexity and multiplicity increases the amount of available information for networks to extract and learn. Another factor is that, at high p_T regions, the primary interaction's signal is much larger compared to the contributions from pileup, so the impact is much smaller. This results in a higher quality data to learn from. For the region where $p_T > 1200$, the sample size reduces exponentially and variability in number of

constituents becomes very large. Hence, the obtained results are expected to become more unstable.

CHAPTER VIII

CONCLUSION

The motivation behind the work in this thesis stemmed from the limitations of standard jet energy corrections in the CMS experiment, which often struggle with the complexities and high dimensional nature of the data produced in particle collisions. While the standard jet energy corrections perform well, there is still much room for improvement in the process. In this thesis, deep learning models were implemented to improve the standard corrections.

Regarding the deep learning, the correction task is a regression problem, the models predict a continuous value, the jet transverse momentum. The two models employed in this work are a fully connected deep neural network and a Particle Flow Network. The fully connected setup can be considered a more straightforward approach to the task, while the Deep Sets-based PFN offers a more sophisticated approach that is capable of handling variable-length inputs and capturing the complex internal structure of jets. Originally the PFN was designed with the classification task in mind but it can be applied to regression as well.

The performance of the models was evaluated based on two key metrics: mean response and relative jet energy resolution. Results demonstrate that the both models significantly outperform the standard corrections. Particularly the PFN shows great performance, exceeding the FC setup in relative resolution and in mean response, only with the exception of very high energy scales. The FC network offers a slightly lower inference time compared to the PFN. However, considering the overall performance improvements, the PFN stands out as a more robust and accurate model for jet energy corrections.

REFERENCES

[1] Wikipedia contributors, "File:standard model of elementary particles.svg," 2024.

[2] "New results indicate that particle discovered at CERN is a Higgs boson." https://home.cern/news/press-release/cern/new-results-indicate-particle-discovered-cern-higgs-boson.

[3] F. J. Boge and C. Zeitnitz, "Polycratic hierarchies and networks: what simulation-modeling at the LHC can teach us about the epistemology of simulation," *Synthese*, vol. 199, pp. 445–480, may 14 2020.

[4] "Lhc Guide." https://cds.cern.ch/record/2255762/.

[5] T. C. Collaboration, "The CMS experiment at the CERN LHC," *Journal of Instrumentation*, vol. 3, pp. S08004–S08004, aug 14 2008.

[6] I. Neutelings, "Cms coordinate system." https://tikz.net/axis3d_cms/.

[7] "Introduction – CMS Detector Pre-Exercise." https://cms-opendata-workshop.github.io/workshop2023-lesson-cms-detector/01-introduction/index.html.

[8] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, "An introduction to PYTHIA 8.2," *Computer Physics Communications*, vol. 191, pp. 159–177, 6 2015.

[9] M. Bähr, S. Gieseke, M. A. Gigg, D. Grellscheid, K. Hamilton, O. Latunde-Dada, S. Plätzer, P. Richardson, M. H. Seymour, A. Sherstnev, and B. R. Webber, "Herwig++ physics and manual," *The European Physical Journal C*, vol. 58, pp. 639–707, nov 20 2008.

[10] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, and H. Araujo, "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, pp. 250–303, 7 2003.

[11] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, and T. Stelzer, "Madgraph 5 : Going Beyond." https://arxiv.org/abs/1106.0522, jun 2 2011.

[12] M. H. Seymour and M. Marx, "Monte Carlo Event Generators." https://arxiv.org/abs/1304.6677, apr 24 2013.

[13] T. C. Collaboration, "Particle-flow reconstruction and global event description with the CMS detector," *Journal of Instrumentation*, vol. 12, pp. P10003–P10003, oct 6 2017.

[14] G. Petrucciani, A. Rizzi, and C. Vuosalo, "Mini-AOD: A New Analysis Data Format for CMS," *Journal of Physics: Conference Series*, vol. 664, p. 072052, dec 23 2015.

[15] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-kt jet clustering algorithm," *Journal of High Energy Physics*, vol. 2008, pp. 063–063, apr 16 2008.

[16] "Cms JEC Run I legacy performance plots." https://cds.cern.ch/record/2052170.

[17] D. Bertolini, P. Harris, M. Low, and N. Tran, "Pileup Per Particle Identification." https://arxiv.org/abs/1407.6013, jul 22 2014.

[18] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, J. D. Hipp, L. Peng, and M. C. Stumpe, "Detecting Cancer Metastases on Gigapixel Pathology Images." https://arxiv.org/abs/1703.02442, mar 3 2017.

[19] H. Ghoddusi, G. G. Creamer, and N. Rafizadeh, "Machine learning in energy economics and finance: A review," *Energy Economics*, vol. 81, pp. 709–727, 6 2019.

[20] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, "Jet-images — deep learning edition," *Journal of High Energy Physics*, vol. 2016, pp. 1–32, jul 13 2016.

[21] D. Belayneh, F. Carminati, A. Farbin, B. Hooberman, G. Khattak, M. Liu, J. Liu, D. Olivito, V. B. Pacela, M. Pierini, A. Schwing, M. Spiropulu, S. Vallecorsa, J.-R. Vlimant, W. Wei, and M. Zhang, "Calorimetry with deep learning: particle simulation and reconstruction for collider physics," *The European Physical Journal C*, vol. 80, 7 2020.

[22] D. Holmberg, D. Golubovic, and H. Kirschenmann, "Jet Energy Calibration with Deep Learning as a Kubeflow Pipeline," *Computing and Software for Big Science*, vol. 7, aug 23 2023.

[23] O. Alexandrov, "Illustration of gradient descent on a series of level sets." https://commons.wikimedia.org/wiki/File:Gradient_descent.svg, aug 7 2012.

[24] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT*, vol. 16, pp. 146–160, 6 1976.

[25] P. J. Werbos, *Applications of advances in nonlinear sensitivity analysis*, pp. 762–770. Berlin/Heidelberg: Springer-Verlag, 1 2005.

[26] P. J. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. John Wiley Sons, mar 31 1994.

[27] H. Mohamed, M. Zahran, and O. Saavedra, "[PDF] ASSESSMENT OF ARTIFICIAL NEURAL NETWORK FOR BATHYMETRY ESTIMATION USING HIGH RESOLUTION SATELLITE IMAGERY IN SHALLOW LAKES: Case STUDY EL BURULLUS LAKE.," 2015.

[28] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep Sets." https://arxiv.org/abs/1703.06114, mar 10 2017.

[29] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks." https://proceedings.mlr.press/v15/glorot11a, jun 14 2011.

[30] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." https://arxiv.org/abs/1412.6980, dec 22 2014.

[31] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448–456, JMLR.org, 2015.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[33] T. C. Collaboration, "A Deep Neural Network for Simultaneous Estimation of b Jet Energy and Resolution," *Computing and Software for Big Science*, vol. 4, oct 30 2020.

[34] K. Kallonen, "Sample with jet properties for jet-flavor and other jet-related ML studies JetNTuple_QCD_RunII_13tev_MC." https://opendata.cern.ch/record/12100, jan 1 2019.

[35] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, 2005.

[36] P. T. Komiske, E. M. Metodiev, and J. Thaler, "Energy flow networks: deep sets for particle jets," *Journal of High Energy Physics*, vol. 2019, 1 2019.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch." https://openreview.net/forum?id=BJJsrmfCZ.

VITA

Ömer Fatih Dokumacı began his studies in physics at İstanbul University's Physics Program in 2017. After receiving his Bachelor of Science degree in 2021, he started the Physics Master's Program at Özyeğin University the same year.