Revisiting Deep Learning for Particle Physics

Thesis submitted in partial fulfilment
of the requirements for the degree of

Master of Science in
Computational Natural Sciences
by Research

by

Jai Bardhan
2018113008
jai.bardhan@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Revisiting Deep Learning for Particle Physics" by Jai Bardhan, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Dr. Subhadip Mitra

To my parents and my brother,
for their unconditional love and support

Acknowledgments

I want to express my sincere gratitude to my advisor, Dr. Subhadip Mitra, for his guidance, encouragement, and support throughout this research work. His expertise and insights have been invaluable in shaping my work and helping me overcome the challenges I faced. I am incredibly grateful for his encouragement and support for all my personal and career goals. I would also like to sincerely thank Dr. Tanumoy Mandal for his valuable feedback and guidance through the projects.

I am grateful to my close friend and partner in crime, Cyrin Neeraj. His mentorship, aid, and friendship made my research experience fun and rewarding. His contribution and feedback to my thesis enhanced its quality considerably. I would also like to thank my PhD lab mates, Arvind Bhaskar and Maaz Khan, whose comforting words and genial attitude during stressful hours enabled me to tackle this project. I particularly thank A.B. for his insightful discussion and help with the experiments and results.

I would also like to thank my friends (`./, NSMJ, Wing Group`) – Ahish, Yoogottam, Pranav K., Sartak, Pranav T., Sarthak, Jaidev, Arpan, Jivitesh, Vikrant, Rohan, Aman, and especially to my old roommate Bhavyajeet Singh and MS lab mates Animesh Sinha and Kalp Shah. I am grateful for their companionship and encouragement through the years, which helped me through some tough times. My discussions with them (especially with A.S., K.S., B.S. and A.D.) provided me with ample constructive criticism, improving my work significantly.

I am thankful to my juniors for motivating me to work hard on my thesis and complete it on time. I am also grateful to them for teaching me to take a break and have fun. My final years in the university would not have been so memorable if it wasn't for them.

Finally, I would like to thank my family for their unconditional love and support. They have always supported me in pursuing my dreams and goals. They have moved heaven and earth to ensure I have access to an excellent educational institute and meet all my needs. I could not have done this without them.

Abstract

The Large Hadron Collider (LHC) experiment is searching for exotic beyond the Standard Model signal processes. These processes occur at extremely low rates compared to the Standard Model ones making it difficult to extract these signals effectively. Classical cut-based methods that rely heavily on hand-crafted strategies fail to perform well on these exotic signals. For the next stage of runs at the LHC, there is an urgent need to develop newer algorithms more capable of identifying these rare signals. Deep learning methods have recently gained traction to assist these searches due to their exceptional ability to analyse complex patterns and show immense potential to improve the sensitivity of experiments. In this thesis, I describe a few approaches to adapt and incorporate deep learning techniques into collider experiments. First, I present a case study of a phenomenological search for a hypothetical heavy quark, where I augment the standard search strategy with a simple deep-learning model to yield better sensitivity and reach. In the case study, I also discuss the caveats of such methods and point to a potential solution. Secondly, I present a simple adaptation of the deep learning training strategy to align it with physics goals. Specifically, I study modifications to the standard Cross Entropy loss and propose a new loss function, based on the Lovasz extension, to directly optimise the sensitivity metric (Z-score). Finally, I present a novel generative model based on the GAN (Generative Adversarial Framework) to generate particle jets. This model can quickly generate jets from different processes and assist jet studies.

Contents

List of Figures

List of Tables

Chapter *1*

Introduction

The Large Hadron Collider began its operation in 2010 and successfully discovered the Higgs boson in 2012. Since then the LHC has undergone multiple upgrades and expanded its search for Beyond the Standard Model (BSM) particles. Currently, Run III is underway, with the collider operating at a higher energy than its previous runs. The LHC is planned to undergo another upgrade (the High-Luminosity LHC, HL-LHC), which will increase the number of collisions. The increased collisions will allow physicists to study processes with smaller cross-sections. A high-energy upgrade, which will enable the LHC to probe higher energies, is also under consideration. These next series of runs at the LHC will search for rarer BSM signal processes. These processes occur with small production rates amidst an overwhelming amount of background processes. Traditional cut-based analyses, where kinematic cuts on hand-crafted physical features are applied, have found isolating the signal processes challenging. This has led to the adoption of multivariate techniques such as decision trees, SVMs, and Gaussian mixture models, among others. However, as we scale to the subsequent runs, these methods would also find it challenging to provide sufficient experimental sensitivity for these rare signals. This has motivated the physics community to explore newer analysis methods, such as deep learning models, which show much promise in improving experimental sensitivity. These methods perform better than other multivariate techniques due to their ability to learn and model complex interactions within the input. Furthermore, their ability to construct complex high-level features from raw data can easily replace the need for hand-crafted features.

While adopting these deep learning models, they are generally treated as black boxes with limited insights into their predictions. Typically, they also require a vast amount of data to make sensible predictions. This thesis explores some of the questions regarding these models and their alignment with particle physics in their applications to current experimental searches and phenomenological projections. Specifically, the thesis we explore the following:

1. We pick a typical new physics projection study at HL-LHC to compare the performance of two widely used ML models — a boosted decision tree (BDT) model and a deep neural network (DNN) model. The signal we pick for the study is the pair-production of a heavy bottom-

like quark from proton-proton collisions. We show that a simple DNN shows significant improvement over a BDT model. DNNs are generally considered to be less interpretable than BDT models. To address this, we also study the application of the method of Integrated Gradients, an interpretability framework, to gain insights into the DNN model. By designing intelligent baselines for the dataset to compare predictions, we see what features the model considers important during the classification process.

2. We identify some simple modifications to the deep learning pipeline that would align the training process with physics goals. We study the improvements in terms of robustness and the alignment of goals through modifications to the simple Binary Cross Entropy loss, introducing two losses: (1) weighted Cross Entropy loss, and (2) surrogate loss for maximising Z-score.

3. With the rise of data-driven methods, there is a significant need for obtaining larger datasets. Deep learning-based generative models can provide a faster, albeit less accurate, alternative to Monte Carlo simulations. Finally, we develop a deep learning-based generative model (Generative Adversarial Network) for generating particle jets at the LHC.

The thesis is organised as follows:

Chapter 2-4 provide the required background for the rest of the thesis — **chapter 2** gives a quick introduction particle physics, **chapter 3** provides relevant background on experimental setups and methods, and **chapter 4** introduces machine learning theory and architectures.

Chapter 5-7 discuss the research problems introduced here. Specifically, **chapter 5** discusses our first question, i.e., the application of DNNs to a phenomenological study, **chapter 6** discusses the alternative losses for DNN training, and **chapter 7** presents a novel generative model based on Generative Adversarial Network (GAN) framework to resolve the data generation issue prevalent in particle physics studies.

Chapter 8 provides the summary and conclusion of the research work presented in the thesis. It also offers some future directions that one may explore to further improve the techniques discussed.

Part I

Background

Chapter *2*

Primer on Particle Physics

The Standard Model (SM) provides the most successful description of nature to date. It unifies all fundamental forces except gravity and, therefore, encompasses the known physics up to TeV scale energies. In the major part of this thesis, we talk about isolating new physics signals at collider experiments; it is important to understand the SM phenomena and how they manifest at the colliders. This chapter gives a brief introduction to the particle content of the SM and the fundamental interactions it describes. For a review of the current state of the SM, we refer the reader to [1, 2].

2.1 Particles in the Standard Model

2.1.1 The Fermions

Fermions are the building blocks of matter and follow the Pauli exclusion principle, which means that no two fermions can occupy the same quantum state. The Standard Model has 12 types of elementary particles that are spin-1/2 fermions. These fermions can be classified into two groups: quarks and leptons. Quarks and leptons have different properties and interactions. Both quarks and leptons carry electric charges and interact through the electromagnetic force. Furthermore, quarks also carry colour charges and interact with each other through strong force.

The fermions are divided based on their generation. Both the quarks and leptons have three generations, each with two particles. The first generation consists of the up and down quarks and the electron, and the electron neutrino. The first-generation fermions are stable under normal conditions and form almost all visible matter in the Universe. For example, the proton and neutrons are constructed through the colour-neutral combinations of up and down quarks. The second generation consists of the charm and strange quarks and the muon and the muon neutrino. The third generation consists of the top and bottom quarks and the tau and the tau neutrino. Increasing generations have higher masses than the previous generation. The second

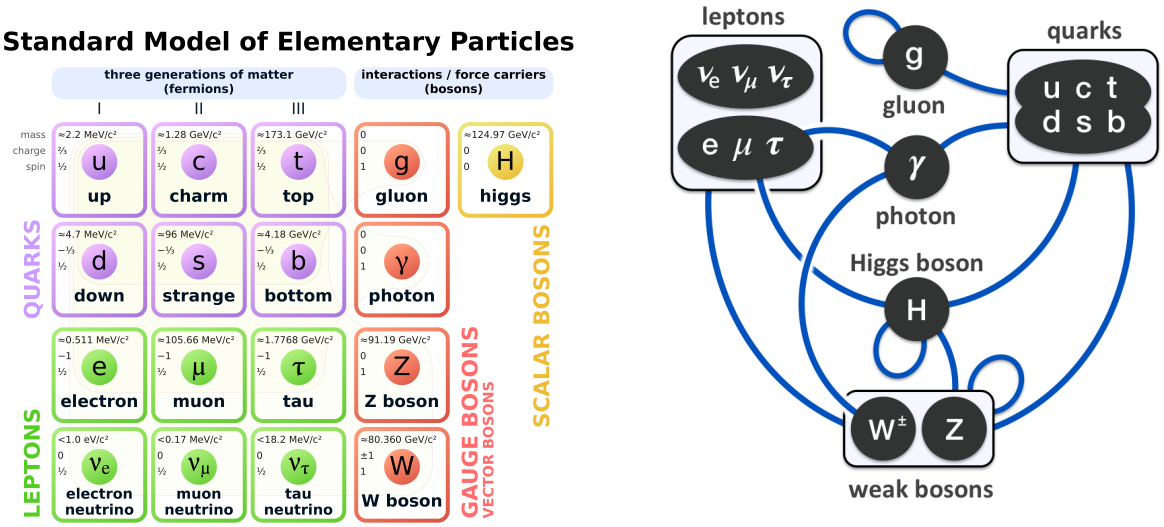


Figure 2.1: The Standard Model of Particle Physics: The particle content (left) and the interactions (right).

and third-generation fermions are unstable and can decay into lower-generation fermions. For example, a muon decays into an electron, an electron anti-neutrino, and a muon neutrino.

2.1.2 The Bosons and the Fundamental Forces

Nature is governed by four fundamental forces that describe how matter and energy interact. These forces are the electromagnetic force, the strong force, the weak force, and the gravitational force. The Standard Model also has 4 types of elementary particles with spin 1 (vector or gauge) or spin 0 (scalar) called bosons, which follow the Bose-Einstein statistics. Some of these bosons mediate these fundamental forces between the fermions. Gauge bosons are described through gauge theories, whereby the gauge bosons preserve the local gauge symmetries of the specific field.

The electromagnetic force is a long-range force that acts on electrically charged particles, such as quarks and electrons. It can be attractive or repulsive depending on the sign of the charges. The electromagnetic force is described by quantum electrodynamics (QED). The photon (γ) (or the quantum of light) mediates the electromagnetic force. It is massless and chargeless and carries unit spin. It interacts with electromagnetically charged particles.

The strong force is a short-range force that acts on particles with colour charges, such as quarks and gluons. It is attractive and binds quarks together to form hadrons, such as protons and neutrons. It also binds protons and neutrons together to form atomic nuclei. The strong force is described by quantum chromodynamics (QCD). The gluon (g) mediates the strong force. It has spin 1, zero mass and colour charges. It can interact with any particle with colour charges.

The name "gluons" comes from the fact that these act as a glue to hold the quarks together in hadrons, such as protons.

The weak force is a short-range force that acts on all fermions, including neutrinos. It is responsible for various types of radioactive decay, such as beta decay and alpha decay, as well as nuclear fission reactions. In the modern description of the Standard Model, the electromagnetic force and the weak force have been unified to give the electroweak theory (EW), which also describes the weak force. The $W^{\pm}$ and Z bosons mediate the weak force. They have spin 1, non-zero mass, and electric charge (except for the Z boson, which is neutral). Since they are massive, they travel slower than the speed of light.

The gravitational force is a long-range attractive force that acts on all objects with mass and energy. Currently, gravity cannot be completely explained using QFT. However, it proposes a hypothetical boson called the graviton with spin 2, which would mediate the quantum gravitational field.

The four fundamental forces have different strengths or coupling constants, which measure how strongly they interact with matter and energy. The strongest force is the strong force, followed by the electromagnetic force, the weak force, and finally, the gravitational force. At sufficiently high energies, the coupling constants of these forces approach a common value, suggesting the existence of a hypothetical unified theory of forces.

Apart from the force mediating bosons, Standard Model also incorporates the Higgs boson (H), which is a special boson that does not mediate any force but instead gives masses to the particles through the Higgs mechanism. It has spin 0, non-zero mass, and zero electric charges. It interacts with any massive particle, such as quarks, leptons, and W, Z bosons. The Higgs field manifests as the Higgs boson, which permeates all space and breaks the electroweak symmetry at low energies.

2.2 Interactions in the Standard Model

Interactions are described through perturbative QFT. In QFT, the fundamental objects are fields and elementary particles (fermions and bosons) are the excited states of their corresponding fields. The perturbative calculations can be represented graphically by the use of Feynman diagrams, which provides a convenient way to visualize and compute the scattering amplitudes for different processes. Higher-order calculations correspond to Feynman diagrams with more vertices and generally involve the use of virtual particles or loops.

2.2.1 Quantum Electrodynamics

Quantum electrodynamics (QED) is the framework that describes how electrically charged particles interact through the electromagnetic force. The photon (γ) is the boson that mediates the

electromagnetic force in QED. In QED, the $U(1)$ gauge symmetry ensures the local conservation of electric charge. The Abelian nature of the $U(1)$ symmetry implies that there is only one force-mediator which is the photon.

QED calculations are perturbative, meaning that they can be done by expanding the probability amplitude in a series of terms with increasing powers of the fine-structure constant (α), which measures the strength of the electromagnetic interaction. The higher-order terms correspond to more complicated processes involving more virtual photons and loops. These perturbative calculations are performed with the aid of Feynman diagrams.

2.2.2 Quantum Chromodynamics

Quantum chromodynamics (QCD) is a non-Abelian gauge theory that describes how particles with colour charge interact through the strong nuclear force. Colour charge is a property of quarks and gluons that determines their interactions in QCD. There are three types of colour charges: red, green, and blue, which have nothing to do with visible colours. The analogy is drawn because we do not see coloured particles in nature; therefore, stable particles are "colourless". This means they have either no net colour charge (such as protons and neutrons) or equal amounts of colour and anti-colour charge (such as mesons). Colour confinement is the phenomenon in that colour-charged particles cannot be isolated and, therefore, cannot be directly observed in normal conditions. This is because the strong force becomes stronger as the distance between the colour-charged particles increases, making it impossible to separate them without creating new particles. Unlike in QED, where there is only one type of photon, there are eight different types of gluon in QCD, drawn from the $SU(3)$ group. The $SU(3)$ gauge symmetry ensures the conservation of colour charge in QCD. Gluons can be assumed to carry two kinds of colour charge: one for emission and one for absorption. For example, a gluon that carries a red-antigreen colour charge can be emitted by a red quark and absorbed by a green quark.

QCD calculations are perturbative at high energies and short distances, which means they can be done by expanding the probability amplitude in a series of terms with increasing powers of the coupling constant (α_S), which measures the strength of the strong nuclear interaction. QCD is not perturbative at low energies and long distances, so it cannot be solved analytically for describing hadrons. For practical purposes, hadron interactions are described using form factors that fit the experimental data.

Hadrons–QCD Bound States: Hadrons are composite particles of quarks and gluons bound by the strong force. Hadrons can be classified into two types: baryons and mesons. The former are hadrons comprising of three quarks with different colour charges (i.e, the colourless combination of red, green, blue colour charges) or three antiquarks with different anti-colour charges (i.e, the colourless combination of antired, antigreen, antiblue colour charges). Baryons have half-integer spin and obey Fermi-Dirac statistics. Examples of baryons are protons and neutrons. The latter

are hadrons comprising of a quark-antiquark pair with opposite colour charges (such as the colourless combination red-antired or green-antigreen colour charges). Mesons have integer spin and obey Bose-Einstein statistics. Examples of mesons are pions and kaons.

2.2.3 Weak Interactions

Weak interactions are mediated by three bosons: the W boson ($W^{\pm}$) with unit electric charge and the neutral Z boson. Unlike the photon and the gluon, which are massless, the Z and W bosons are massive, with masses of about 80 GeV and 90 GeV, respectively. The mass of these bosons implies that the weak interactions have a very small range of the order of 10^{-17} m. It also implies that they can decay into other particles, such as quarks, leptons, or photons. Weak interactions describe how particles change their flavour, such as quarks changing their type. These interactions are described by the $SU(2)_L$ gauge symmetry, i.e., the interacting particles are invariant under local $SU(2)$ transformations. The $SU(2)_L$ symmetry implies that only particles with left-handed chirality (i.e., those with their spins and momenta pointing in the opposite directions) have nonzero weak isospins. Weak isospin is the quantum number that distinguishes between the left-handed particles forming doublets, such as up and down quarks or the electron and the electron neutrino, etc. Right-handed particles can interact with the Z boson through their hypercharges.

2.2.4 The Role of the Higgs Boson

Higgs field fills the Universe and gives mass to all other elementary particles. The Higgs boson is a scalar boson, meaning it has no spin, unlike all other elementary particles. It is a massive boson with a mass of 125GeV. The Higgs field interacts with all other particles in different ways. The strength of this interaction determines how much mass a particle acquires from the Higgs field. The Higgs boson's mass is related to the strength of the Higgs field and its self-interaction. A heavier Higgs boson implies a stronger Higgs field and a stronger self-interaction.

The Higgs boson is unstable and decays almost instantly into other particles after it is created. The most methods of production at the LHC is through gluon fusion. This process involves two gluons colliding and fusing into a virtual top quark-antiquark pair, emitting a Higgs boson.

2.3 The Standard Model is an Effective Theory

The Standard Model has been remarkably successful in explaining a wide range of phenomena and experiments, but it also has some limitations and puzzles that remain unsolved. Gravity is the weakest of the four fundamental forces. It is responsible for the attraction between masses, such as the Earth and the Moon. However, gravity is not included in the Standard Model because

it is incompatible with QFT. There is no quantum theory of gravity that can describe how gravity works at tiny scales, such as near black holes or during the Big Bang. One major challenge in physics is finding a way to unify gravity with the other forces in a consistent theory. Dark matter (DM) is another puzzle that does not fit into the SM. DM is inferred to exist from its gravitational effects on visible matter, such as galaxies and clusters of galaxies. The nature and origin of dark matter are unknown, and none of the particles in the Standard Model can account for it. There are many candidates for dark matter particles, such as neutrinos, axions, or weakly interacting massive particles (WIMPs), but none have been detected.

There are many other aspects of the Standard Model that are not fully understood or explained, such as the origin and hierarchy of masses and couplings of the particles, the number and types of generations of leptons and quarks and the asymmetry between matter and antimatter in the universe, among others. These open questions motivate physicists to hunt for new physics beyond the SM using theoretical models and experimental tests.

Chapter *3*

Collider Experiments

LHC experiments offer testing grounds for our conjectures about nature. These experiments collide protons at high energy to probe the physics at small scales mainly in two ways: 1) to verify SM predictions at high precision and 2) to search for well-motivated, new-physics models beyond the SM. Before searching for a theoretically promising model, its collider prospects are benchmarked using state-of-the-art software that simulates various detector environments. It is, therefore, crucial to understand a typical LHC experiment, the pipeline for the projection studies and how we statistically measure the effectiveness of a particular experiment. In this chapter, we briefly look at the physics of a collider experiment and the statistics behind making a discovery or ruling out a model using these experiments. For a detailed explanation, see Refs. [3, 1, 2].

3.1 Colliders

Controlled particle physics experiments are performed using particle colliders. In a collider, two beams of opposing particles collide at high energies in order to study their internal structure and subsequent decay [4]. The collider can either be a ring accelerator or a linear accelerator. Linear accelerators, such as the Stanford Linear Accelerator Center (SLAC), can achieve high energies without significant energy loss, but due to the linear topology, they require large areas for construction (which eventually limits the maximum energy attainable). Ring accelerators, such as the Large Hadron Collider (LHC), are built due to their smaller size and ring topology, which allows for continuous acceleration since particles can transit indefinitely. However, as particles accelerate in ring colliders, they lose energy due to the magnetic field needed to maintain the particles within the collider (and thus, they also have an upper limit to the maximum energy achievable) [5]. Similarly, one can also classify current colliders based on what kinds of collisions they investigate. Electron-positron colliders give high purity and allow us to study the decay precisely, while proton-proton colliders allow us to probe QCD at higher energies, albeit at significantly noisier final states [1].

3.2 Energy and Luminosity

Apart from the type of collisions, colliders are also parameterised by their centre of mass energy and luminosity. Collision of two beams of particles accelerated to high energies E_1, E_2 respectively gives us access to a centre of mass energy $E_{\text{cm}} \approx 2\sqrt{E_1 E_2}$, assuming a small crossing angle. Most collider studies are done between particles of equal mass, giving us a centre of mass energy $E_{\text{cm}} = 2E_b$ where E_b is the energy of one of the particle beams.

To investigate high-energy particle physics phenomena, we require not only sufficiently high energies but also a large number of events (collisions). Fewer events of interest would not provide sufficient statistics or opportunities for physicists to study the phenomenon.

The cross-section of the process under investigation and the time integral of the instantaneous luminosity $\mathcal{L}$ determine the number of events of interest,

$$N_{\text{exp}} = \sigma_{\text{exp}} \cdot \underbrace{\int \mathcal{L}(t)dt}_{\text{Integrated Luminosity}} \tag{3.1}$$

The cross-section σ_{exp} provides a measure of the probability of the process to occur, which depends on the theoretical model and the centre of mass energy. The luminosity $\mathcal{L}$ is a measure of the rate of collisions, which depends on the beam parameters such as the number of particles per bunch, the number of bunches per beam, the beam size and divergence, and the crossing angle [5, 1]. A new physics phenomenon generally has a low cross-section and therefore requires high luminosity to be studied.

3.3 Collision Event

Many phenomena at various scales and energies are involved in a collision, which is a complicated physics process. In order to understand and analyse a collision event, we can divide it into four main stages.

1. **Hard Scattering:** This is the stage where the initial partons (quarks or gluons) from the colliding protons interact at incredibly short distances and high energies. This stage can be described by quantum field theory (QFT) using perturbative quantum chromodynamics (QCD). The outcome of this stage can be calculated by finding the scattering amplitude $\mathcal{M}$ from the Feynman diagrams that describe the possible processes.

2. **Parton Shower:** This is the stage where the final partons produced in the hard scattering emit multiple gluons and quarks, resulting in a cascade of partons with lower energies and longer distances. This stage cannot be described by perturbative QCD because the coupling constant becomes large, and the calculations become divergent. Instead, this stage can be

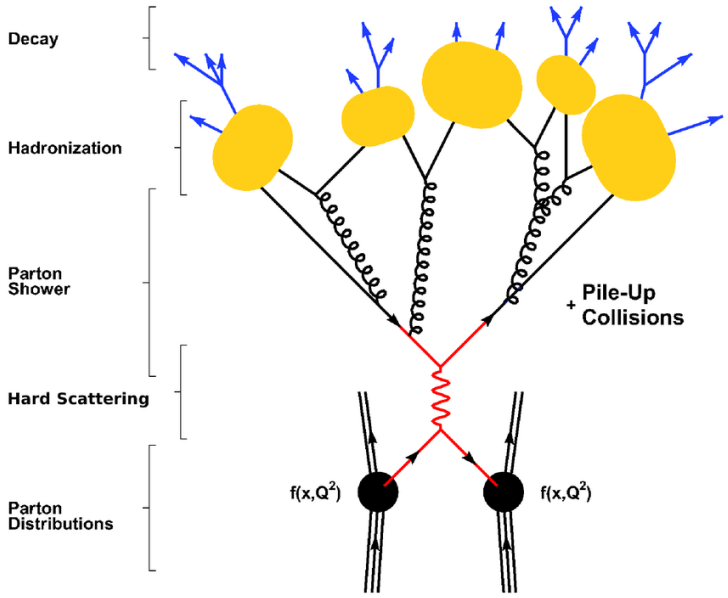


Figure 3.1: Schematic of a Hadronic Collision at the LHC. *Image credits [13]*

simulated by using classical approximations of QCD, such as the DGLAP [6] or CCFM [7] equations [1], which describe how partons split.

3. **Hadronization:** This is the stage where the partons from the parton shower combine together to form colourless hadrons, i.e., mesons and baryons. This stage also cannot be described by perturbative QCD because it involves non-perturbative effects and confinement. Instead, this stage can be modelled by using phenomenological approaches, such as the Lund string model [8] or the cluster model [9, 10], which describe how partons form bounded states [1].

4. **Detection and Clustering:** This is the final stage where the hadrons from the hadronization, along with leptons and photons, reach the detector and are measured by various components. This stage can be described using detector simulation software, such as GEANT4 [11] or DELPHES [12], which accounts for various detector efficiencies. After the detection, clustering algorithms are applied to group together hadronic objects into jets, which serve as a substitute for the original partons.

3.4 Large Hadron Collider

The Large Hadron Collider is a ring collider. It is a proton-proton collider which means that it is used to study the QCD interactions at high energies. It currently operates at a centre of mass energy of 6.8TeV, with a maximum design value of 7 TeV. In the coming years, it will undergo an

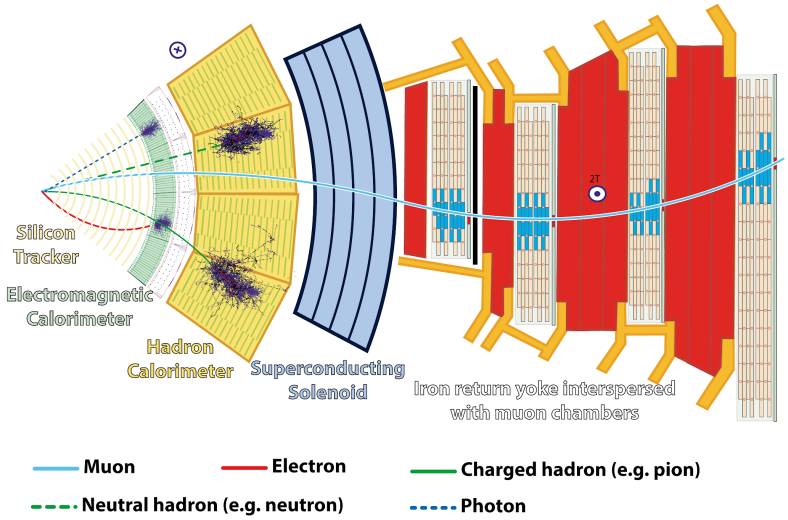


Figure 3.2: Cross Section of the CMS detector. *Image credits: CERN, TensorFlow*

ambitious upgrade to HL-LHC, with the LHC Injector Upgrade, that would improve the integrated luminosity to 3ab^{-1} [14].

3.4.1 LHC Detectors

The LHC detector is composed of four collision points (detectors) (such as ATLAS, CMS, LHCb and ALICE). Here we briefly explain the CMS detector since the overarching design remains similar between these experiment detectors. For a complete review, we refer the reader to [15, 16].

The Compact Muon Solenoid detector (CMS) detector is located at one of the collision points of LHC. The CMS detector has a cylindrical shape with several layers of components arranged in a concentric design. These components help us record and identify the collision event by recording the properties of the particles of the event. This is done by:

1. **Bending Particles:** A powerful solenoid magnet bends the particles that emerge from the collision event. Bending particles help us determine: (1) the charge of the particle since positively and negatively charged particles bend in opposite directions; (2) the momentum of the particle, as particles with higher momentum bend lesser than the ones with lower momentum.

2. **Tracking particles:** To calculate the momentum and identify a particle, the path of the particle is tracked, using silicon trackers, as it bends. The tracker can reconstruct the path of heavy muons, light leptons, hadrons and short-lived particles such as the b quark.

3. **Measuring Energy:** Information about the particle's energies is crucial to identifying the particle. The CMS detector has two energy calorimeters, ECAL and HCAL. The Electromagnetic Calorimeter (ECAL) is the inner calorimeter with the main goal of measuring

the energy of electrons and photons, which completely stop there. Hadrons, composed of quarks and gluons, fly through the ECAL and are stopped at the Hadronic Calorimeter (HCAL), which measures their energy content. Hadrons also deposit some energy content at the ECAL as they travel through it.

4. **Detecting Muons:** The muon is the final particle that the CMS directly measures. They can pass through the calorimeters, so special sub-detectors are needed to detect them as they travel through the CMS detector.

3.4.2 What is seen by the detector?

Different particles interact with the detectors differently, which allows us to reconstruct and identify them.

- **Electrons:** These are charged particles that lose most of their energy in the ECAL. They also bend under the magnetic field, which allows us to determine their charge and momentum. However, it is still challenging to identify them with high accuracy and precision because they can be confused with other particles that deposit energy in the ECAL. The identification depends upon several other factors, such as the ratio of energy deposited in the ECAL to HCAL and isolation from other particles, among others.

- **Photons:** These are neutral bosons that also mainly interact with the ECAL. However, unlike electrons, they do not bend under the influence of the magnetic field, and they do not leave any tracks in the detector. Therefore, we can reconstruct photons as clusters of energy in the ECAL that are not associated with any tracks. The photon identification also depends on several variables, such as the shape of the shower, isolation from other particles, etc.

- **Muons:** These are heavier than electrons and therefore lose less energy in the ECAL and the HCAL. We can reconstruct muons as tracks that match with hits in the muon chamber. The muon identification relies on several variables, such as the number of hits in the tracker and muon chamber, the quality of the track fit, and isolation from other particles.

- **Charged Hadrons:** These particles interact strongly with the HCAL, where they create showers of secondary hadrons. They also bend under the magnetic field and leave tracks in the tracker. We can reconstruct them as clusters of energy in the HCAL that have matching tracks.

- **Neutral Hadrons:** These particles also interact strongly with the HCAL but do not have an electric charge and therefore do not bend under the magnetic field. They do not leave any tracks and are reconstructed as energy deposits in the HCAL that are not matched with any tracks.

- **Jets:** Jets are sprays of stable hadrons that originate from final state partons of hard scattering. These partons undergo a series of emissions, followed by hadronization to form the spray of stable hadrons, which is the jet. At the detector, we observe groups of stable hadrons originating from different partons. Therefore, we need to cluster these stable hadrons to reconstruct the jets. These are two types of clustering algorithms — (1) Cone-based algorithms, which try to find stable cones that contain the detected hadrons, and (2) Sequential clustering algorithms, which iteratively combine the detected hadrons to form stable clusters [17]. The generalized k_T algorithm is a sequential clustering algorithm that uses the following distance measure to hierarchically combine particles:

$$d_{ij} = \min(p_{T_i}^{2p}, p_{T_j}^{2p}) \frac{\Delta R_{ij}}{R^2}, \tag{3.2}$$

  where p_{T_i} and p_{T_j} are the transverse momenta of particles i and j, ΔR_{ij} is the angular separation between the two particles, R is a radius parameter, and p decides the weight given to softer or harder particles. $p = 1$ gives us the k_T algorithm, which clusters softer particles first to form harder subclusters that are then clustered together [18]. $p = 0$ gives us the Cambridge-Aachen (C-A) algorithm, which clusters spatially closer particles first, followed by more distant particles. $p = -1$ gives the ubiquitous anti-k_T algorithm, which clusters particles around hard particles and results in clusters that are centred around the hard particle [19]. This makes this algorithm suitable for experiments due to its robustness to noise and pileup. Some jets are larger than others depending upon the value of R used in the clustering algorithm. These larger jets are called fatjets, and they often have substructure since they come from partons that decay into smaller partons before hadronizing. For example, the W boson can decay into a quark-antiquark pair, which forms a two-pronged fatjet (W-jet), and a top quark can decay into a b quark and a W boson, which forms a three-pronged jet (top jet). We can study the internal substructure of these jets to identify their origin and tag them appropriately.

Apart from these, there are neutrinos which are neutral leptons that interact very weakly with matter and do not leave traces in the detector. They escape undetected and carry away some of the energy and momentum of the collision.

3.5 Computational Tools

To simulate and analyze collision events, we need to use various tools that can handle different stages of event generation and reconstruction, Fig. 3.3. In this section, we will briefly describe some of the tools that we use in our project and their roles.

MadGraph [20]: MadGraph is a tool that can calculate the matrix elements for the hard scattering processes. MadGraph uses Monte Carlo techniques to sample the phase space and generate

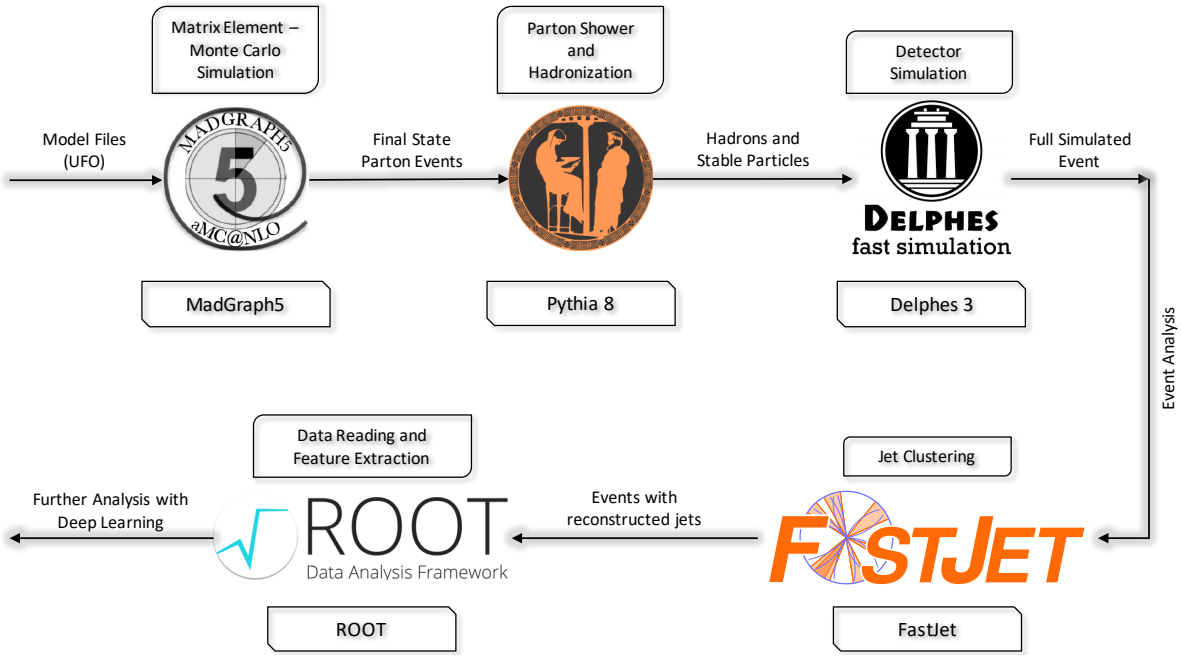


Figure 3.3: The pipeline of Computational Tools

events according to the cross-sections and kinematics of the processes. MadGraph can handle a wide range of processes, both standard model and beyond standard model processes and can interface with other tools for further steps.

Pythia [21]: Pythia is a tool to simulate the parton shower and hadronization stages of the event generation. Pythia uses classical approximations to model how the partons from the hard scattering emit gluons and quarks to form a stream of low-energy partons. Pythia also models how the partons form colourless hadrons, i.e., mesons and baryons. Pythia can also simulate some aspects of the underlying event and multiple interactions.

Delphes [12]: Delphes is a software package to replicate the detector response to the stable particles obtained in the event. Delphes is not a full detector simulation but rather a fast simulation that mimics the performance of realistic detectors using simpler probabilistic methods. Delphes can account for the detector geometry, material, resolution, and efficiency, as well as introduce noise and smearing effects. Delphes can also perform some basic reconstruction tasks, such as track finding.

FastJet [22]: FastJet is a tool that performs jet clustering. FastJet provides several techniques for jet clustering, such as the k_T algorithms and SISCone. These algorithms are based on different criteria for defining and merging jets, such as distance or momentum. FastJet also provides some tools for jet analysis, such as jet substructure and grooming.

There are other tools that can perform similar or complementary tasks for simulating collision events, such as Geant4 [11], Herwig [23], or Sherpa [24]. Some of these tools have more features

or accuracy than the ones we use, but they also require more computational resources. The tools that we have chosen are the standard for phenomenological studies.

3.6 Kinematics at the Collider

We move on to describe the kinematics of the collision since these are relevant to the features we can obtain from the reconstructed objects in the previous section. Special relativity dictates that collision events are invariant under the transformations in the Lorentz Group. The energy E and the 3-momentum of the $\mathbf{p}$ of a particle of mass m form a 4-vector $p = (E, \mathbf{p})$, whose square $p^2 = E^2 - |\mathbf{p}|^2 = m^2$. The particle's velocity is $\beta = \mathbf{p}/E$ is a Lorentz scalar. All transformations that preserve p^2 (or Lorentz scalars) are Lorentz-Invariant.

3.6.1 Mandelstam Variables

In two-body body collisions, two particles of momentum p_1, p_2 and masses m_1, m_2 respectively often interact to give rise to two particles with momentum p_3, p_4 and masses m_3, m_4 respectively. The Mandelstam variables are Lorentz-invariant quantities that characterize the kinematics of the reaction:

$$\begin{aligned} s &= (p_1 + p_2)^2 = (p_3 + p_4)^2 \\ &= m_1^2 + 2E_1E_2 - 2\mathbf{p_1} \cdot \mathbf{p_2} + m_2^2 \end{aligned} \tag{3.3}$$

$$\begin{aligned} t &= (p_1 - p_3)^2 = (p_2 - p_4)^2 \\ &= m_1^2 - 2E_1E_3 + 2\mathbf{p_1} \cdot \mathbf{p_3} + m_3^2 \end{aligned} \tag{3.4}$$

$$\begin{aligned} u &= (p_1 - p_4)^2 = (p_2 - p_3)^2 \\ &= m_1^2 - 2E_1E_4 + 2\mathbf{p_1} \cdot \mathbf{p_4} + m_4^2 \end{aligned} \tag{3.5}$$

These variables are crucial for calculating various properties of the interaction, such as cross-sections. Specifically, $\sqrt{s}$ is the E_{cm} of the collision.

3.6.2 Pseudorapidity

Rapidity y is used to define the direction of the trajectory along the beam direction (i.e., this becomes the direction for the z-axis). Rapidity is then defined by:

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right) \tag{3.6}$$

$$= \ln \left(\frac{E + p_z}{m_T} \right) = \tanh^{-1} \left(\frac{p_z}{E} \right), \tag{3.7}$$

where m_T is the transverse mass defined as $m_T^2 = m^2 + p_x^2 + p_y^2$, which differs from the definition used by experimentalists, E is the energy of the particle, and p_z is the momentum along the z axis.

Importantly, the shape of the rapidity distribution is invariant to a Lorentz boost in the z-direction, i.e., rapidity differences are invariant to Lorentz boosts along the beam direction. This makes rapidity crucial in accelerator physics.

For $p >> m$, the rapidity may be expanded to obtain

$$y = \frac{1}{2} \ln \frac{\cos^2(\theta/2) + m^2/4p^2 + \dots}{\sin^2(\theta/2) + m^2/4p^2 + \dots} \tag{3.8}$$

$$\approx -\ln \tan(\theta/2) \equiv \eta \tag{3.9}$$

Therefore the pseudorapidity η is approximately equal to rapidity y for $p >> m$. Pseudorapidity is more commonly used than rapidity since it is generally difficult to measure the rapidity for highly relativistic particles.

3.6.3 Transverse Variables

In collider experiments, a significant and unknown amount of energy of the incoming protons is lost along the beam pipe. Momentum in the direction transverse to the beam pipe is defined as the transverse momentum $\mathbf{p_T}$. If an invisible particle is created (such as a neutrino), we can only measure and constrain it in the axis transverse to the beam pipe, as an unknown amount of energy is lost along the pipe. We define missing transverse energy as:

$$E_T^{\text{miss}} = -\sum_i \mathbf{p_{T_i}} \tag{3.10}$$

If events have more than 1 invisible particle in the final state, it recovers the vector sum of the 3-momentum of the invisible particle.

For the decay of a mother particle (of mass M) into two particles, of which one is invisible, we can constrain the mass of the mother particle using a quantity "transverse-mass".

$$M_T^2 = \left[E_T(1) + E_T(2)\right]^2 - \left[\mathbf{p_T}(1) + \mathbf{p_T}(2)\right]^2 \tag{3.11}$$

$$= m_1^2 + m_2^2 + 2\left[E_T(1)E_T(2) - \mathbf{p_T}(1)\mathbf{p_T}(2)\right] \tag{3.12}$$

where p_T for the invisible particle is defined as $p_T = E_T^{\text{miss}}$

3.7 Statistics at the LHC

When analyzing the data from LHC for a BSM particle, one can find either one of two outcomes: we observe the BSM signal, or we do not. If we do not find evidence for the presence of the signal

process, the result is utilized to provide limits for one or more parameters of the BSM model. This is a simple example of a "limit-setting" experiment.

One of the simplest kinds of experiments that can be run to test for the presence of a signal process is a counting experiment. In this experiment, one counts the number of events that pass certain selection criteria and compares it with the expected number of events from known processes. There are two hypotheses:

H_0: This is the “background-only” (null) hypothesis, which supposes that there is no beyond standard model physics. It suggests that the observed result could be a statistical fluctuation of the background processes, $N^B = \sigma^B \times \epsilon^B \times \int \mathcal{L}(t)dt$, where σ^B is the background processes' cross-section, and ϵ^b is the efficiency of the selection criteria.

H_1: This hypothesis considers the existence of the signal process. In the nominal case, we would expect additional events, $N^S = \sigma^S \times \epsilon^S \times \int \mathcal{L}(t)dt$, over N^b.

Suppose for each event in the signal sample, one measures a variable x and uses these values to create a histogram $\mathbf{N} = (N_1, N_2, \ldots, N_k)$. The expected value for each N_i is written as,

$$\mathbb{E}[N_i] = \mu N_{S_i} + N_{B_i}, \tag{3.13}$$

where N_{S_i} and N_{B_i} is the mean number of entries in the ith bin from signal and background. The signal process' strength is determined by the parameter μ, where $\mu = 0$ corresponds to the "background-only" hypothesis, and $\mu = 1$ means the nominal signal hypothesis. The likelihood function is given by the product of Poisson probabilities for each bin:

$$L(\mu) = \prod_{j=1}^{k} \frac{(\mu N_{S_j} + N_{B_j})^{N_j}}{N_j!} e^{-(\mu N_{S_j} + N_{B_j})}. \tag{3.14}$$

For the remainder of the section, we will take the case of only 1 bin, i.e., $k = 1$. However, one can expand the following results for more than one bin to achieve better results.

To test a hypothesized value of μ, we define the profile likelihood-ratio test as,

$$\lambda(\mu) = \frac{L(\mu)}{L(\hat{\mu})} \tag{3.15}$$

where $\hat{\mu}$ is the parameter independently optimized to maximize the Poisson likelihood function L. The profile likelihood ratio measures how likely it is for the observations to have come from the model with a given μ. The lower the value, the less likely the hypothesis model explains the data. We have not included any nuisance parameters θ in this analysis, as they will complicate the derivations and are not essential for the final results.

We define two test statistics in order to study the presence of a signal. The first test statistic is for testing for discovery and tests whether the background-only hypothesis could fluctuate

upwards to explain the observations. Generally, we do not consider downward fluctuations and assign those to experimental errors or uncertainties:

$$q_0 = \begin{cases} -2\ln\lambda(0) & \hat{\mu} \geq 0 \\ 0 & \hat{\mu} < 0 \end{cases} \tag{3.16}$$

The second test statistic is used to put an upper bound on the signal strength parameter and constrain or exclude the signal+background hypothesis. The test statistic tests whether the signal + background hypothesis could fluctuate downwards to explain the observations. Here we do not test for upwards fluctuations as they could indicate a different BSM signal or model,

$$q_\mu = \begin{cases} -2\ln\lambda(\mu) & \hat{\mu} \leq \mu \\ 0 & \hat{\mu} > \mu \end{cases} \tag{3.17}$$

Supposed we test a value μ for the strength parameter, and suppose we collect data with a true value of strength parameter μ', we would obtain $\hat{\mu}$ follows a Gaussian distribution with mean μ' and standard deviation σ. Then using Wilk's theorem [25] and Wald's approximation [26], we obtain that:

$$-2\ln\lambda(\mu) = \frac{(\mu-\hat{\mu})^2}{\sigma^2} + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) \tag{3.18}$$

where N represents the sample size. In the limit of a large sample size, the approximation becomes accurate.

We now define an "Asimov dataset" as a hypothetical data set such that it gives back our true parameters when we use it to evaluate the estimators for all parameters [3]. In our case, if we use an Asimov data set to estimate $\hat{\mu}$, we would get $\hat{\mu} = \mu'$.

Using the approximation in equation (3.18), one obtains the connection between the Z score test statistic (referred to as Z score henceforth) and q_0, q_μ. The Z score is preferred as that is more commonly used.

$$Z_0 = \sqrt{q_0} \tag{3.19}$$

$$Z_\mu = \sqrt{q_\mu} \tag{3.20}$$

3.7.1 Experimental Sensitivity

To identify the sensitivity of an experiment, we are interested in the expected (median) significance with which we can reject various values of μ. For example, in the case of discovery sensitivity, we would like to know the median assuming the nominal signal model ($\mu = 1$), with which we can reject the background-only ($\mu = 0$) hypothesis. In the case of exclusion limits, we define the sensitivity as the median significance, assuming data is generated using the background only ($\mu = 0$) hypothesis that can reject a nonzero value of μ (generally, $\mu = 1$).

The median values of q_0, q_μ can easily be obtained using an Asimov data set, and they lead to simple expressions for the median significance. Since Z is a monotonic function of q, as shown in equations (3.19) and (3.20), its median is given by the corresponding function of the Asimov value (median) of q. For discovery using q_0, one wants the median discovery significance assuming a strength parameter μ', and for upper bounds using q_μ, the median exclusion is calculated taking a strength parameter μ assuming data is from $\mu' = 0$. Therefore we get

$$\text{Discovery:} \qquad \text{med}[Z_0|\mu'] = \sqrt{q_{0,A}} \tag{3.21}$$

$$\text{Exclusion:} \qquad \text{med}[Z_\mu|0] = \sqrt{q_{\mu,A}} \tag{3.22}$$

An important case is a counting experiment with a known value of mean background N_B. Now the data collected is only the count N, and thus the likelihood function is:

$$L(\mu) = \frac{(\mu N_S + N_B)^N}{N!} e^{-(\mu N_S + N_B)} \tag{3.23}$$

The test statistic for discovery is q_0, and we obtain the optimized parameter $\hat{\mu} = (N - N_B)/N_S$ that maximizes $L(\hat{\mu})$. Substituting this into q_0, and only considering the case of $\hat{\mu} \geq 0$, we get:

$$Z_0 = \sqrt{q_0} = \sqrt{2N \ln\left(\frac{N}{N_B}\right) + 2N_B - 2N} \tag{3.24}$$

We are interested in the median significance, assuming the data is from the nominal signal hypothesis ($\mu' = 1$). Then we see that for an Asimov data set, $\hat{\mu} = \mu' = 1$, which will give $N = N_S + N_B$. Therefore, we get [3]

$$\text{med}[Z_0|1] = \sqrt{q_{0,A}} = \sqrt{2(N_S + N_B) \ln\left(\frac{N_S + N_B}{N_B}\right) - 2N_S} \tag{3.25}$$

Similarly, if we are interested in the test statistic for exclusion q_μ with $\mu = 1$, we see upon substituting,

$$Z_\mu = \sqrt{q_\mu} = \sqrt{2(N_S + N_B - N) - 2N \ln\left(\frac{N_S + N_B}{N}\right)} \tag{3.26}$$

Since we are interested in the median significance, assuming that the data is from the $\mu' = 0$ hypothesis, we set $\hat{\mu} = \mu' = 0$, which gives us $N = N_B$. Substituting that, we get the asymptotic exclusion limit as,

$$\text{med}[Z_1|0] = \sqrt{q_{\mu,A}} = \sqrt{2N_S - 2N_B \ln\left(\frac{N_S + N_B}{N_B}\right)} \tag{3.27}$$

In the case of $N_S \ll N_B$, which is generally the case with a search for new physics, we see that both equation (3.25) and (3.27) approximate to:

$$\text{med}[Z_0|1] \approx \text{med}[Z_1|0] \approx \frac{N_S}{\sqrt{N_B}} \tag{3.28}$$

3.7.2 Experimental Tests

Analyses at Tevatron and LHC use the likelihood ratio test statistic [27, 28, 29], defined by,

$$\Lambda(N^d) = \frac{L(N^d; N^S + N^B)}{L(N^d; N^B)} \tag{3.29}$$

to test between the null hypothesis $\mu = 0$ (background only), and the nominal signal hypothesis $\mu = 1$ (signal+background). Here N^d is the number of events in the data collected, and $L(a; b)$ means a Poisson likelihood of mean a evaluated at b. Alternatively, we define:

$$q = -2 \ln \Lambda(N^d) \tag{3.30}$$

The probability distribution function for this test statistic is estimated by performing a series of pseudo-experiments for both the "background-only" and "signal+background" hypotheses. For each pseudo-experiment, we sample from a Poisson distribution with mean N_B ($N_S + N_B$) for the background only (signal + background) hypothesis to generate the number of pseudo-events observed.

The compatibility of the data with the two hypotheses is assessed by comparing the test statistic value observed in actual data q_{observed} with the probability distribution functions calculated from the pseudo-experiments. Confidence levels (CL) can then be defined as:

$$CL_B = \int_{q_{\text{observed}}}^{\infty} p_B \left(q \left(N^d \right) \right) \mathrm{d} \left(q \left(N^d \right) \right) \tag{3.31}$$

$$CL_{S+B} = \int_{q_{\text{observed}}}^{\infty} p_{S+B} \left(q \left(N^d \right) \right) \mathrm{d} \left(q \left(N^d \right) \right) \tag{3.32}$$

We use CL_b to estimate the probability that the null hypothesis (background) would fluctuate upwards enough to explain the observed excess if an excess over the background predictions is observed. The background-only hypothesis may be rejected if this probability is small. Conversely, when we do not observe a significant excess, we calculate the probability that the alternate hypothesis (signal+background) would fluctuate downwards to "mask" the existence of the signal. We can exclude the signal+background hypothesis if this probability is small. However, CL_{s+b} is not suitable for this as it can lead to "spurious exclusions", where we may exclude a model even if the experiment had no sensitivity for the particular model. The modified-frequentist method fixes this issue by considering

$$CL_s = \frac{CL_{s+b}}{CL_b}. \tag{3.33}$$

CL_s gives more conservative and appropriate exclusions compared to CL_{s+b}.

q_μ and q_0 lead to equivalent tests to q in the limit of Wald's Approximation. They are, thus, optimal in the Neyman-Pearson lemma sense [3].

Chapter *4*

Machine Learning

4.1 Introduction

Machine Learning models are designed to learn to perform a specific task without being explicitly programmed to do so. Machine learning has become a rapidly expanding field in the last decade, employing various machine learning techniques to teach computers to perform tasks. In this chapter, I briefly introduce machine learning methods, specifically those which are commonly used in physics analyses. For a detailed explanation of the concepts mentioned here, see Refs. [1, 30]. For a comprehensive review of how these ML models have been used in various areas in particle physics in particular, see Ref. [31].

4.1.1 A Representative Example

Let us take the problem of jet tagging as a representative example of machine learning and how it may be used in particle physics. In jet tagging, the goal is to identify the parton (quark/gluon/boson/BSM particles) which initiated the clustered jet of particles observed at the detector. The detector, in principle, can measure various properties of the particles in the jet. But, for the simplicity of this example, we will only consider some high-level features that are reconstructed from the clustered jet, such as jet p_T, jet mass, Nsubjetiness, etc. (although one could also use low-level features and let the machine learning algorithm reconstruct the relevant high-level features). These features form a vector input $x \in \mathbb{R}^d$ to the machine learning algorithm.

We can perform Monte Carlo simulation and obtain pairs of samples (x, y), where x is the feature vector of the jet and y is the label specifying the kind of jet (top, quark, etc.). We want to learn a parameterised function $f_\theta : x \to y$ that can accurately predict the class of the jet. This can be made concrete by specifying a loss function $\mathcal{L}(f_\theta(x), y)$, which is an indicator of how poor the model performs at the prediction task.

Ideally, we would want to minimise the total risk: $\mathcal{R}(f_\theta) = \int_{x,y} \mathcal{L}(f_\theta(x), y) * p(x, y) d\mu$, which is the expected loss over the true distribution of jets. However, in most cases, we do not have access to the true distribution as it is intractable or unknown. Instead, we have finite samples $(x, y) \overset{i.i.d}{\sim} p(x, y)$, which form our training dataset. Therefore we instead minimize the empirical risk $\mathcal{R}_{\text{emp}}(f_\theta) = \sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i)$. The empirical risk formulation assumes that the true distribution can be approximated as the sum of the delta distribution of the samples within the training dataset. One can now train the network using an optimiser (which is generally gradient-based), such as gradient descent.

Typically, we observe that the empirical risk evaluated on a separate test dataset is higher than on the training dataset. Significant differences in the risk indicate *overfitting*, and the model does not perform well on unseen data. The ability of the model to accurately perform the task on unseen data is referred to as *generalisation*, and the empirical risk evaluated on the test dataset gives a measure of the *generalisation error*. If the model performs poorly on both the training dataset as well as the test dataset, then it indicates *underfitting*.

Formally, we are trying to find a function $f \in \mathcal{F}$, where $\mathcal{F}$ is the family of functions that our model can represent, such that:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathcal{R}_{\text{emp}}[f] \tag{4.1}$$

The expressivity of a model characterises its ability to minimise empirical risk. It depends upon the specific implementation of the model and the regularisation techniques employed.

One can classify machine learning based on the kind of problem that it solves.

4.2 Supervised Learning

In supervised learning, one has access to the i.i.d input-output tuples $\{x_i, y_i\}_{i=1,2,\ldots,N}$ in the training dataset. Supervised learning can be further classified into two broad categories: regression and classification.

4.2.1 Regression

In regression, the output $y \in \mathcal{Y}$ is a real-valued scalar, but the input x may be a real-valued vector or more unstructured input such as images, point clouds, sentences, etc. In statistical terms, the output y is often referred to as the dependent variable and the input x is known as the independent variable. In classical statistics, generally, we assume that there is normally distributed additive noise around the target such that,

$$y_i = f_\phi(x_i) + e_i \tag{4.2}$$

where e_i is the additive noise and is independent of x. This assumption leads to the least squares method in the case of linear regression. However, one can relax this assumption and assume an arbitrary joint distribution $p(x, y)$.

4.2.1.1 Mean Squared Error

Mean Squared Error (MSE) is a commonly used metric for evaluating the performance of regression models. It uses the *Squared Error* loss function,

$$\mathcal{L}(y, f(x)) = (y - f(x))^2 \tag{4.3}$$

to measure the discrepancy between the predicted values and the actual values. MSE has the advantage that it is always positive, and a value of 0 indicates a perfect fit. Also, contrary to initial expectations, MSE is not limited to cases where the conditional distribution $p(y|x)$ is normally distributed. Regardless of the underlying distribution, using the calculus of variations, we can show that the optimal regressor for the MSE is,

$$f^*_{\text{MSE}}(x) = \mathbb{E}_{p(y|x)}\left[y\right], \tag{4.4}$$

i.e., the conditional expectation of y given x.

However, one issue with squared error as the loss function is that it is sensitive to outliers. Alternatively, as the loss function, one can use *absolute-error*, $|y - f(x)|$.

4.2.2 Classification

In classification, the output $y \in \mathcal{Y}$ is a discrete value (generally with no continuous connection or ordering between the values). The output y can be interpreted as a class label. A particular case of classification is when the label y takes on only two values (such as "signal" and "background"), known as binary classification.

4.2.2.1 Cross Entropy Loss

Cross entropy loss is a loss function commonly used in classification. It quantifies the difference between the actual probability distribution of classes and the predicted probability distribution. It can be interpreted as the average number of bits needed to identify an event drawn from a sample of classes if a coding scheme is based on the predicted probability distribution rather than the true distribution.

The cross-entropy loss function for binary classification is defined as,

$$\mathcal{L}(y, f(x)) = -y \log(f(x)) - (1 - y) \log(1 - f(x)) \tag{4.5}$$

where y is the true label (0 or 1), and $f(x)$ is the predicted probability for the positive class. This loss penalises differences in predictions such that larger differences are penalised more heavily than smaller differences.

Cross entropy may be generalised for multi-class classification as,

$$\mathcal{L}(y, f(x)) = -\sum_{i=1}^{C} y_i \log(f_i(x)) \tag{4.6}$$

where C is the number of classes, y_i is the true label for class i (0 or 1), and $f_i(x)$ is the predicted probability of class i.

Maximising the (Multinoulli) likelihood with respect to the parameters of the model f can be seen as minimising the cross-entropy loss. Furthermore, cross-entropy minimisation is also related to KL divergence ($D_{KL}(p||q)$) minimisation between the true distribution p and the predicted distribution q. If f is a linear function, then the cross-entropy loss is the same as logistic regression, another classification loss.

4.3 Unsupervised Learning

In unsupervised learning, one only has access to i.i.d samples $\{x_i\}_{i=1,2,...,N}$ in the dataset. Unsupervised learning methods have gained significant attention in recent years due to the difficulty of obtaining ground truth labels for many problems.

4.3.1 Representation Learning

Unsupervised representation learning aims to transform the data without using any label information. By discovering the structure of the data, these methods can provide a transformation that can represent data in a more meaningful way.

Traditional examples of representation learning include clustering methods (such as KMeans, DBSCAN), dimensionality reduction (such as PCA, LLE) and matrix factorisation techniques (such as NMF), among others. Clustering methods aim to group similar data points together based on some similarity measure, such as Euclidean distance. Dimensionality reduction methods aim to project high-dimensional data onto a lower-dimensional space while preserving some properties of the data, such as variance or local neighbourhood. Matrix factorisation techniques aim to decompose a matrix into simpler matrices that can reveal hidden factors or patterns in the data.

Modern deep-learning techniques rely on auxiliary tasks to learn good embeddings for the data. Some examples include autoencoders and self-supervised learning methods such as contrastive learning and inpainting. Autoencoders are neural networks that try to reconstruct the input data from a compressed representation, which can force the network to learn meaningful features of the data. Contrastive learning is a technique that tries to learn embeddings that are similar for

positive pairs (such as two transformations of the same image) and dissimilar for negative pairs (such as two different images). Inpainting is a technique that tries to fill in missing parts of an image based on the surrounding context, which can encourage the network to learn the structure and semantics of the image. We refer the reader to [cite, cite, cite] for more information.

4.3.2 Generative Modelling

Generative modelling refers to the class of problems where the objective is to learn the underlying probability distribution of a given data set and to sample or generate new data from that distribution.

Generative Adversarial Networks [32] are a kind of generative modelling technique which involves two networks competing with each other. The first network, the generator, has the goal of generating samples that are identical or mimic the real sample, while the second network, the discriminator, has the purpose of discriminating between the generated and real samples. A game theoretic analysis of this zero-sum game shows that at ooptimalityptimality, the discriminator ensures that a particular probability distance measure (which depends upon the specific formulation) is reduced by training the generator. In the original formulation, this turns out to be the Jenson-Shannon divergence. In newer formulations, the generator minimises the Wasserstein distance [33] or the χ^2 divergence [34] instead.

There are other generative models, such as Variational Autoencoders [35] Autoregressive models, Normalizing Flows [36], Diffusion models [37] and Poisson Flow models [38], which are widely used by scientists in different applications [39].

4.4 Boosted Decision Trees

An adaptive basis-function model (ABM) is of the form $f(x) = \omega_0 + \sum_{i=1}^{M} \omega_m \phi_m(x)$, where $\phi_m(x)$ is the m$^{\text{th}}$ basis function, which is learned from the data. This basis function specifies the family of functions that can be optimized in equation (4.1).

Decision Trees, or CART (Classification and Regression Trees), models are built by splitting the input space recursively and defining a local model for each input space segment. This can be done in a hierarchical way, with each leaf node representing a segment. Decision trees construct axis aligned splits to partition the N-dimensional space into regions. In the case of regression, we can now calculate a mean response for each region. Similarly, in the case of classification, we can associate the distribution over the class labels instead of the mean response in each of the regions.

4.4.1 Decision Tree Construction

Finding an optimal partitioning of the space is NP-Complete [40], so it is common to greedily optimize the local maximum likelihood scores. One defines a splitting function to choose the best feature (j^*) and the best value for the feature (t^*) to split the region upon. The splitting function has the form:

$$(j^*, t^*) = \arg \min_{j \in \{1,\dots,D\}} \min_{t \in \mathcal{T}_j} \text{cost}(\{x_i, y_i : x_{ij} \leq t\}) + \text{cost}(\{x_i, y_i : x_{ij} > t\}) \tag{4.7}$$

where the cost function for a given problem and dataset is defined in section [ref], t defines a threshold from the set of thresholds $\mathcal{T}_j$ for a feature j, and x, y are the input and label, respectively. While one can allow splitting the tree into more than two child trees, it may lead to very few samples in some leaves, known as data fragmentation.

The cost function of equation [eq] can be defined differently based on whether the problem is a regression or a classification problem.

4.4.1.1 Regression Construction

In regression, we define the cost as follows:

$$\text{cost}(\mathcal{D}) = \sum_{i \in \mathcal{D}} (y_i - \bar{y})^2, \tag{4.8}$$

where $\bar{y} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} y_i$ is the mean of the response variable of the specified set of data. Alternatively, one may fit a linear regressor for each leaf using inputs chosen along the path to the leaf, i.e., pass the series of splits.

4.4.1.2 Classification Construction

In the case of a classification problem, there are different criteria to measure the quality of a split. We fit a categorical distribution to the distribution of labels within each of the leaves by estimating the class probabilities as,

$$\hat{pi}_c = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c) \tag{4.9}$$

where $\mathcal{D}$ is the data in the leaf.

Entropy cost

$$\mathbb{H}(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi}_c \log \hat{\pi}_c \tag{4.10}$$

The entropy is minimized when the information gain between the class label Y and the test $X_j < t$ is maximized

$$\text{infoGain}(X_j < t, Y) = \mathbb{H}(Y) - \mathbb{H}(Y | X_j < t) \tag{4.11}$$

Gini Index

$$\sum_{c=1}^{C} \hat{\pi}_c(1-\hat{\pi}_c) = \sum_c \hat{\pi}_c - \sum_c \hat{\pi}_c^2 = 1 - \sum_c \hat{\pi}_c^2 \tag{4.12}$$

Gini Index quantifies the expected error rate and assesses the impurity or heterogeneity of a set of data. A Gini Index value of 0 indicates perfect purity, meaning all instances in a subset belong to the same class. Conversely, a Gini Index value of 1 signifies maximum impurity, implying an equal distribution of instances across all classes.

Decision trees have several advantages. They are interpretable and can handle both discrete and continuous data and features. However, they have some disadvantages as well. They do not perform as well as some modern methods and are prone to overfitting.

4.4.2 Ensembling Methods

Ensembling methods are techniques that combine predictions of multiple models to make better and more reliable predictions. These methods have gained popularity due to their ability to enhance prediction quality and deal with complex problems. Ensembling methods generally train a group of different base models and combine their predictions to make a final prediction.

Averaging methods, such as Bagging and Random Forests, build multiple independent models and then average their predictions. These methods introduce randomness by training models on different subsets of the data or using different subsets of features. By taking the average of these base models, the ensembling method reduces the variance and improves generalization by taking the average of these base models.

On the other hand, boosting methods, such as Gradient Boosting and AdaBoost, sequentially build models wherein each base model learns to correct the mistakes made by the previous models. Boosting methods assign higher weights to incorrectly classified data samples to prioritize them in the subsequent models. This iterative process focuses on difficult instances and improves the ensemble's performance.

Gradient Boosting is a popular boosting algorithm combining boosting with gradient-based optimization. The method minimises a loss function by iteratively adding weak learners to the ensemble. Each weak model is fitted to the steepest descent of the loss function. Boosted decision trees, a variant of Gradient Boosting, utilize decision trees as the weak models. These decision trees are generally shallow trees, and the combination of multiple shallow decision trees through boosting allows them to capture complex interactions and model non-linear relationships. The final prediction of the ensemble is obtained by summing the predictions of all the decision trees, weighted by a learning rate that controls the contribution of each tree.

In summary, ensembling methods combine the predictions of multiple models to improve the accuracy and robustness of predictions.

4.5 Neural Network

Neural networks are a class of machine learning models inspired by the neuron architecture of the brain. A simple neural network can be formed by stacking a series of perceptron blocks, each followed by a non-linear activation. The following equation describes the perceptron block:

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} \tag{4.13}$$

where $\mathbf{W}$ is the parameter learned from the data.

Neural networks are proven to be universal function approximators with just 2 layers of neurons with sufficiently large layer widths [41] (i.e., number of neurons per layer). However, this proof requires an exponentially large number of neurons as the complexity of the problem rises. Scaling the neural network to add more layers (i.e., make it deeper) has shown to perform better even with fewer nodes on each layer. Therefore, there is a significant effort to develop deeper networks. Designing bigger and deeper networks requires considerable scientific and engineering effort due to larger networks' greater instability and computational complexity.

4.5.1 Activation Functions

The non-linear activation function takes inspiration from the dendrites that communicate between different neurons by releasing activation chemicals. If the activation is not a non-linear function, it is evident that the network becomes an affine transformation. No one activation function works best across all problems, and the choice of the activation function is empirical. However, it has been found that activation functions that do not lead to vanishing gradients tend to provide faster convergence. The general recommendation for designing deep neural networks is to use a `ReLU` [42] activation function given by: $a(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$, where the $\max$ is taken element-wise of the vectors. However, `ReLU` has been shown to perform poorly when the input values take on negative values as the activation becomes zero. In order to mitigate this issue, newer activation functions have been proposed, such as `Swish` [43] and `Mish` [44]. We refer the reader to review [45] for more information.

4.5.2 Backpropogation and Gradient-Based Optimization

Due to the graph structure of all neural networks, it is easy for one to calculate the gradient of the output with respect to each of the internal neurons using the chain rule of differentiation. This process of calculating the gradients by using the graph structure of the internal connections is called backpropagation (named so as the algorithm for calculating the gradients first calculates the gradient of the output based on the last layer and then uses the chain rule to compute the gradients for the preceding layers, thus *propogating back* through the network).

A host of gradient-based optimization methods have been developed that can be used to optimize the parameters of the network using the gradients. The simplest of them is the standard stochastic gradient descent which updates the parameters as,

$$\theta = \theta - \eta \mathrm{d}\theta, \tag{4.14}$$

where θ are the parameters and η is a learning rate that is a hyperparameter. While the gradient descent method has been shown to provide optimal results within polynomial time complexity for convex optimization, neural network optimization is highly non-convex and therefore, vanilla gradient descent fails to perform effectively. Some of the commonly used variants are:

- **SGD with momentum:** It incorporates a momentum term to accelerate convergence. It accumulates the weighted average of past gradients and uses the weighted average to update the parameters.

$$v = \beta v - (1 - \beta)\mathrm{d}\theta \tag{4.15}$$

$$\theta = \theta + \eta v \tag{4.16}$$

  v is the velocity term, and β is the momentum coefficient which controls the averaging.

- **RMSProp:** It uses the sum of squares of the previous gradients to reduce the variance in the update term. Specifically, it maintains an exponential moving average of squared gradients, which gives more importance to recent gradients compared to past gradients.

$$c = \beta c + (1 - \beta)(\mathrm{d}\theta)^2 \tag{4.17}$$

$$\theta = \theta - \eta \frac{\mathrm{d}\theta}{\sqrt{c} + \epsilon} \tag{4.18}$$

  c is the cache term and maintains the sum of squares, β is the coefficient and controls the averaging.

- **Adam [46]:** It combines the concepts of both RMSProp and momentum. It maintains a running average of both the gradients and the squared gradients, allowing it to adaptively adjust the learning rate while incorporating momentum.

$$v = \beta_1 v + (1 - \beta_1)\mathrm{d}\theta \tag{4.19}$$

$$c = \beta_2 c + (1 - \beta_2)(\mathrm{d}\theta)^2 \tag{4.20}$$

$$\hat{v} = v/(1 - \beta_1^t) \tag{4.21}$$

$$\hat{c} = c/(1 - \beta_2^t) \tag{4.22}$$

$$\theta = \theta - \eta \frac{\hat{v}}{\sqrt{c} + \epsilon} \tag{4.23}$$

  Equation (4.21) and eq. (4.22) correct for the bias present in the initial stages of exponential averaging.

4.5.3 Regularization

Regularization is essential to prevent neural networks from overfitting. Regularization encourages the neural network to learn more robust and generalizable patterns by introducing additional constraints or penalties to the learning. A few common methods are explained here.

- **Weight Decay (L2 Regularization):** Adds a penalty to the loss function proportional to the squared values of the model's weight. It encourages smaller weight values, preventing the model from assigning excessively large weights to singular features. Moreover, it encourages smoother decision boundaries and improves numerical stability.
- **Dropout [47]:** Randomly drops a fraction of the neurons while training. Neurons in the neural network can become highly co-adaptive, i.e., they rely on the presence of other specific neurons to produce meaningful outputs. This regularization forces the network to learn redundant representations and prevents individual neurons from relying too heavily on specific input features, encouraging learning more generalizable features. Some works also interpret dropout as training multiple models in parallel. By dropping neurons, the network effectively samples different architectures during each training iteration. At test time, the network approximates the behaviour of an ensemble of these models.
- **Batch Normalization [48]:** Normalizes each layer's inputs by subtracting the batch mean and dividing by the batch standard deviation. This technique helps stabilize the learning process, reduces the internal covariate shift, and allows the network to learn efficiently. It also introduces some noise and makes it more robust to small perturbations in the input, which encourages smoother outputs.

4.6 Neural Network on Unstructured Data

Unstructured data refers to data that does not have a predefined format or structure, such as text, images, video, point clouds, etc. Raw unstructured data often contains abundant and complex information but is also noisy and heterogeneous. Therefore, processing and analyzing raw unstructured data poses many challenges for traditional data tools and methods.

4.6.1 Jets as Particle Clouds

Jets have typically been represented as detector images, p_T ranked sequences of particles or decay trees. However, a more natural representation for such data would be an unordered, permutation-invariant *set of particles*. We refer to this representation as a "particle cloud", analogous to the point cloud representation of 3D shapes used in computer vision. The two representations are very similar in that they are unordered sets of objects distributed irregularly

in space (4D vs 3D), and in both cases, the elements of the cloud are not unrelated but belong to a higher-level object. Moreover, particle cloud representation provides us with similar flexibility to other representations, specifically the ability to include arbitrary features for each particle [49].

Developing neural networks for point cloud data poses a few problems, namely:

- Point Clouds are irregular. Therefore, we cannot apply conventional convolutional neural networks (CNNs) or recurrent neural networks (RNNs) that rely on regular grids or sequences.
- Point Clouds are permutation-invariant. Therefore, any function of such data should also be permutation invariant. Conventional neural networks are not permutation-invariant, and to train them to learn this invariance, we would require data scaling proportional to the factorial of the size of the point cloud.
- Point Clouds are sparse and noisy. This makes it difficult for conventional methods to extract robust and discriminative features from point clouds.

4.6.2 Graph Neural Networks

Graph Neural Networks are a class of neural networks that operate on graphs. One can adapt GNNs to work on point cloud data by observing the relation that point clouds can be interpreted as graphs with (1) no edges between the nodes, (2) locally connected nodes in (spectral) space, or (3) complete graph with connections between all the nodes. There may be other ways to construct graphs from point clouds, but most approaches adopt GNNs for point clouds through either of the three constructions mentioned [50].

Graph Neural Networks provide a solution to the problems posed by point cloud data:

- GNNs handle the irregular nature of point clouds using graph convolutions that do not rely on fixed grids or sequences.
- GNNs are designed to be permutation-invariant since graph data is also permutation-invariant.
- GNNs can handle sparse data through the use of graph pooling and graph attention, allowing it to aggregate information from neighbouring nodes.

The family of functions learned by Graph Neural Networks can be described using the Message Passing framework [51, 52], whereby one node passes its information to the neighbouring nodes, and functions are learned using the passed information. With $x_i^{(k-1)} \in \mathbb{R}^F$ denoting the node feature of node i in layer $(k-1)$ and $e_{ij} \in \mathbb{R}^D$ denoting (optional) edge features from node j to node i, message passing graph neural networks can be described as:

$$x_i^{(k)} = \gamma^{(k)} \left(x_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)} \left(x_i^{(k-1)}, x_j^{(k-1)}, e_{ij} \right) \right) \tag{4.24}$$

where $\bigoplus$ denotes a differentiable, permutation invariant function, e.g., sum, mean, or max, and γ and ϕ denote differentiable functions such as MLPs.

Once we learn representations for each node, we can obtain representations for the entire point cloud (graph) through a global pooling operation applied over all the nodes. Similarly, if we want to obtain representations for the edges, we can do so by transforming the features of the edge's vertices. Graph Neural Networks have found great use in high-energy physics. We refer the reader to Ref. [53] for a review of the application of GNNs in high-energy physics.

Part II

Research Work

Chapter *5*

Application of ML in BSM searches at the LHC: A case study with heavy-quark signals

BSM search results in the conventional channels from the LHC have constrained new physics phenomena to a high mass scale ($\gtrsim 2$ TeV). Observing new physics would involve effectively isolating signals with extremely low cross sections from SM processes with huge cross sections. The task becomes exceptionally challenging with the proposed new colliders and upgrades to LHC (i.e. HE-LHC and HL-LHC). Machine learning models have been shown to perform well for such classification tasks [31]. In this chapter, I discuss how we can use a simple deep-learning model to improve the prospects of a typical BSM signature at the HL-LHC. The signal we pick is the production of a pair of heavy B quarks and its subsequent decay in a well-motivated phenomenological extension of the SM. I discuss the kinematic features that can be used to separate such a signal from similar backgrounds and compare the performance of different machine learning models like BDT and DNNs. The better-performing DNN model is discussed in more detail. To close the chapter, I briefly discuss how to interpret the DNN model predictions using integrated gradients.

5.1 Vectorlike Quarks, in brief

Vectorlike quarks (VLQs) are non-chiral fermions which are present in many extensions of the SM. Unlike the SM quarks, their left- and right-hand representations transform identically under the weak $SU(2)$ group. They can have singlet, doublet and triplet representations under weak group [54]. Detectors at the LHC, ATLAS and CMS have an active search program for these quark partners [55, 56, 57, 58, 59]. These searches generally assume that VLQs decay into an SM quark and a gauge or an SM Higgs boson due to their mixing with the SM quarks. Direct limits on their masses, which come from searches for scattering processes producing a pair of these quarks, vary from $1.4 - 1.8$ TeV depending on various representations and branching ratios

of the final states searched for. Their expected mass scale is far larger than that of the SM quark masses (top quark mass at 172 GeV); hence, we also refer to them as heavy quarks.

Collider searches have not found these quark partners while searching for them in the conventional channels (yet, I should add). This prompts us to check for 'gaps' in these searches. Recent papers have explored scenarios where a sub-TeV singlet state Φ, i.e. a scalar or a pseudoscalar, coupling to VLQs in the context of larger extensions of the SM [60, 61, 62, 63, 64, 65]. This can lead to many exotic final states at the LHC. Still, collider searches are yet to look for VLQs in such channels. Minimally, we assume that such a singlet only couples with the VLQs. Then, its decay gives a di-jet signature consisting of either two quarks or gluon jets [66]. Such signatures with mostly hadronic objects are challenging to isolate at the collider from the huge SM backgrounds. We can make use of ML models to isolate such challenging signatures. As a concrete example,

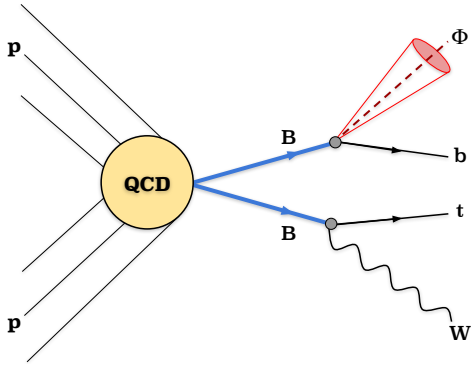


Figure 5.1: An illustrative diagram of the signal topology for the pair-produced B scattering process, where one B decays to a BSM singlet and a b-quark and other decays to a top quark and a W-boson.

this chapter studies the production of a pair of singlet B VLQs, also called the bottom (b-quark) partners. Like the SM b-quark, it comes in three colour varieties and has an electric charge $-1e/3$, where e is the charge of an electron. When it also couples with Φ, there are four total decays possible for a singlet B quark:

$$B \to \begin{cases} tW \\ bH \\ bZ \\ b\Phi \end{cases} . \tag{5.1}$$

Our goal is to estimate the LHC reach of future searches for pair-produced singlet B in the presence of $B \to b\Phi$ decay mode. Therefore, we demand that one of the pairs produced B decay gives a bottom quark and Φ. We also require that the other B decays to a top quark and a W-boson to maximise the signal yield and to add more well-defined objects that can be reconstructed at the LHC. We demand that either the top quark or the W-boson decay to a lepton and a neutrino. Leptons have dedicated calorimeters at the LHC (ECAL and the Muon Chamber),

Background Processes		σ (pb)
V + jets [67, 68]	W + jets	1.95×10^5
	Z + jets	6.33×10^4
tt [69]	tt + jets	988.57
Single t [70]	tW	83.10
	tb	248.00
	t + jets	12.35
VV + jets [71]	WW + jets	124.31
	WZ + jets	51.82
ttV [72, 73]	ttZ	1.05
	ttW	0.65
	ttH	0.61

Table 5.1: Higher-order cross sections of the SM backgrounds considered in our analysis. The cross-sections (σ) are in units of picobarn.

therefore giving cleaner signatures. An illustrative Feynman diagram for the signal process is shown in Fig. 5.1.

We need to isolate such a signal from the known SM processes. All the background processes that can yield the same final states as a signal must be considered. Specifically, we demand that there should be at least one lepton (from the decay of the top quark or W-boson coming from heavy B) in the final state. The appropriate background processes (which gives one or more leptons) and their cross-sections at $\sqrt{s} = 14$TeV are shown in Table 5.1.

5.2 Search Setup

5.2.1 Process Generation

We apply FeynRules [74] to generate the model files (UFO) [75] for the Vectorlike B scenario. We use MadGraph5 [20] to simulate the hard scattering process at the leading order, Pythia8 [21] for the showering and hadronisation, and Delphes3 [12] to mimic the generic LHC detector environment. The events are generated at $\sqrt{s} = 14$ TeV. To account for the boosted kinematics of the final state objects, we adjust the `DeltaRMax` parameter (the radius of an electron cone centred around an identified track), within the lepton isolation criteria, from 0.5 to 0.2, following Ref. [76]. The b-tagging efficiency and the mis-tag rate for lighter quarks were updated to reflect the medium working point of the DeepCSV tagger from Ref. [77].

5.2.2 Reconstructed Objects, Kinematic Cuts

To analyse an event, we need to reconstruct certain objects/particles as done by the experiments at the LHC. At the LHC, this is done using dedicated tracking modules and measurements at the calorimeters. We use DELPHES3 to mimic the detector environment and the efficiencies of various particles at the LHC. Broadly, the more objects we choose to reconstruct, the more control we have over isolating a signal event from backgrounds. However, reconstructing an object means that we have to deal with the efficiencies associated with identifying it. Even though we generate events (both signal and background) with the exact topologies we require, the events might not have the desired final state objects after parton showering and factoring in the efficiencies of isolating that particular particle at the LHC. We also know that kinematically final state particles coming from the signal events must have a boosted profile compared to the ones from the backgrounds. This means that after identifying them, we can put demands on the reconstructed objects' kinematic features to cut background contributions. This ensures that the ML model performs the classification in a harder phase space, where the background events mimic the signals more closely. We refer to each such demand of ensuring a particular final state particle or constraining its kinematic quantities as a cut on the phase space. The cuts chosen for this particular study are listed below (for more detail, we refer the reader to the paper [78]):

$\mathfrak{C}_1$: *Exactly 1 lepton* ($\ell \in \{e, \mu\}$).
The lepton is required to have a $p_T > 100$ GeV, $|\eta| < 2.5$ and must obey the updated isolation criteria mentioned earlier.

$\mathfrak{C}_2$: $H_T > 900$ GeV, where H_T is the scalar sum of the transverse momenta of all hadronic objects in an event.

$\mathfrak{C}_3$: *At least 3 AK-4 jets with* $p_T > 60$ GeV.
The leading jet must have $p_T > 120$ GeV.

$\mathfrak{C}_4$: *At least 1 b-tagged jet with* $p_T > 60$ GeV.
At least one of the AK-4 jets must be identified as a b jet.

$\mathfrak{C}_5$: *At least 1 fatjet (J) with* $R = 1.2$ *and* $p_T > 500$ GeV.
The fatjet is clustered using the anti-k_T algorithm, and the parameters have been optimised to tag a Φ fatjet. We also demand the invariant mass of the fatjet, M_J, to be more than 250 GeV.

$\mathfrak{C}_6$: $\Delta R_{bJ} > 1.2$.
We demand that at least one of the identified b jets is well separated from the leading fatjet passing $\mathfrak{C}_5$, i.e., the b jet lies outside the fatjet cone.

Feature Type	Description	Features
Kinematic Variables	For each of the reconstructed objects, we consider the transverse momentum. Furthermore, we also take the scalar sum of transverse momentum H_T, and the missing energy E_T.	$\{p_{T_{j_i}}\}_{i=1,2,3}$, p_{T_J}, H_T, E_T, ...
Jet Substructure	We take the N-subjetiness variables for the fatjet. This is to study the prongness of the fatjet.	τ_{21}^{β}, τ_{32}^{β} $\forall\ \beta = 1, 2$
Invariant Masses	We take the invariant mass of the fatjet, as we expect the signal process to show a peak around the mass of the Φ scalar, but a noisy spread for the background processes. We also further take the invariant mass of the combinations of the various jets.	m_J, $\{m_{j_m}\}_{m=1,2,3,b}$, $\{m_{j_m j_n}\}_{m,n=1,2,3,b,J}$, $\{m_{j_m j_n j_o}\}_{m,n,o=1,2,3,b,J}$, $\{m_{\ell j_m}\}_{m=1,2,3,b,J}$, ...
Girth of Jets	The girth is the p_T weighted average distance of the constituents to the jet axis. If one thinks of this as the first order central moment of the distribution of energy with distance to the jet axis, one may generalise to higher-order central moments.	ω_J, $\{\omega_{j_a}\}_{a=1,2,3,b}$, Var $[j_a]_{a=1,2,3,b,J}$, Skew $[j_a]_{a=1,2,3,b,J}$, Kurt $[j_a]_{a=1,2,3,b,J}$

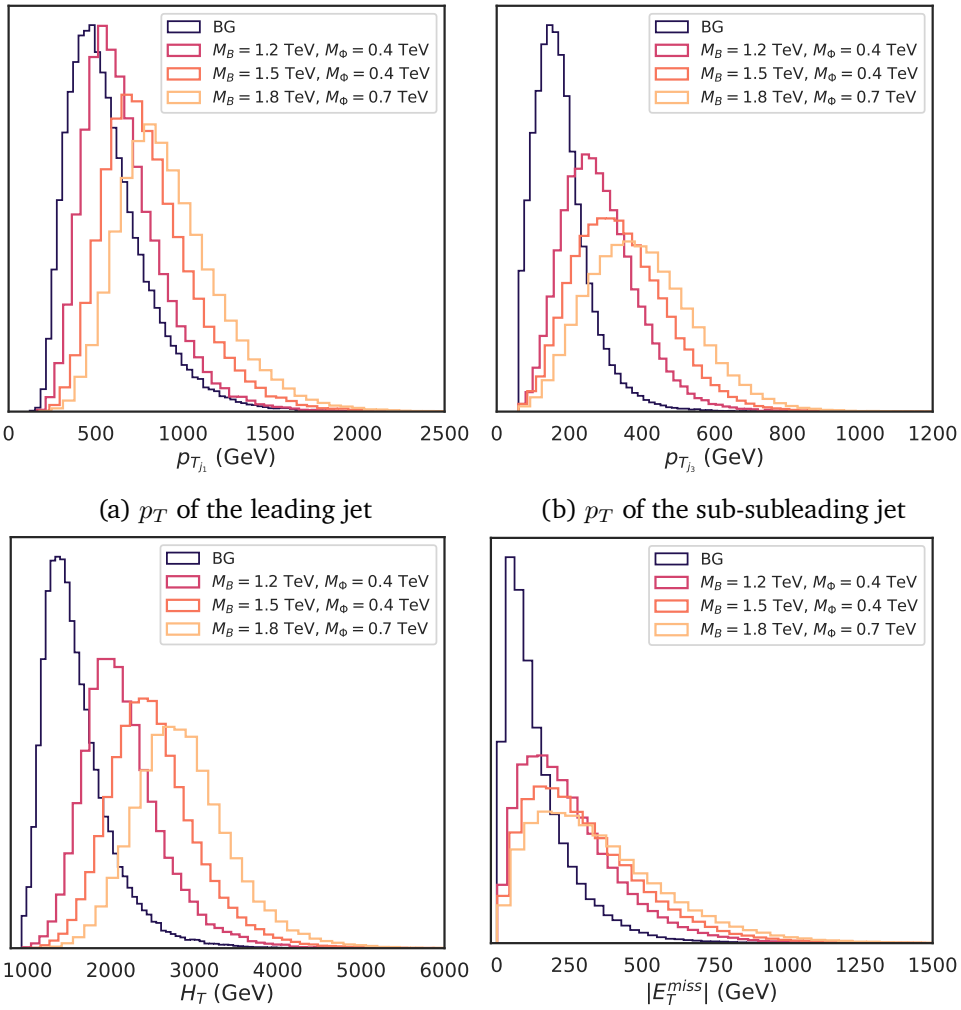
Table 5.2: Categorization of reconstructed features and their descriptions

5.2.3 Feature Selection

We use a comprehensive set of kinematic variables reconstructed from the event as input to these learning-based methods.

Kinematic Features: The boost of the final state jets are significantly higher in the signal compared to the background processes. While this may not be that apparent in leading jet p_T, Fig. 5.2a, (as the signal has higher H_T and more jets to distribute the total p_T compared to fewer hard jets in the dominant background), it is clear in the sub-subleading jet p_T, Fig. 5.2b. The sub-subleading jet has significantly higher p_T in the case of the signal process than the background process. Signal events also usually have higher scalar H_T than all the background processes, Fig. 5.2c. This is due to the higher mass (energy) of the pair produced B VLQs, which decay to give a higher transverse boost. The same trend as H_T can be seen in the missing E_T distribution for the signal process in Fig. 5.2d.

Jet Substructure Features: We expect the signal fatjet to be 2-pronged and the largest background to have a 3-pronged fatjet. In Fig. 5.3c, we can see from the Nsubjetiness $\tau_{21}^{\beta=2}$ ratio that the selected fatjet in the signal events is *more* more consistent with two prongs than the selected fatjets in background events. This is expected, as the scalar ϕ in the signal event decays to two hadronic jets and is reconstructed as the fatjet. In background events, top jets are predominantly



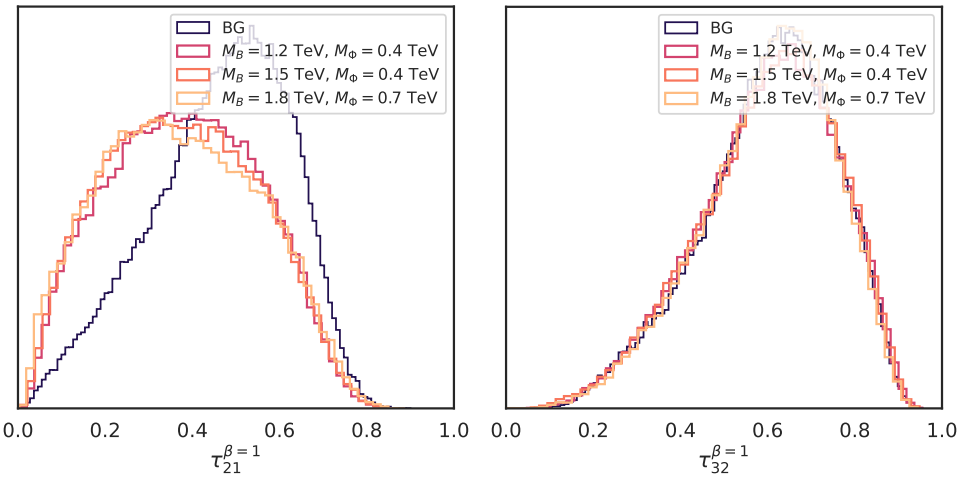
(a) p_T of the leading jet

(b) p_T of the sub-subleading jet

(c) Scalar H_T

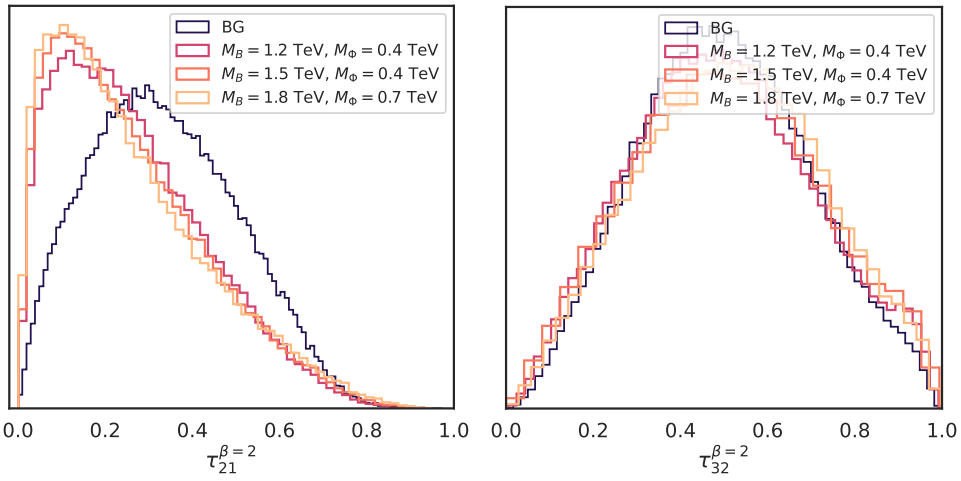
(d) Missing E_T

Figure 5.2: A few kinematic variable distributions



(a) $\tau_{21}^{\beta=1}$ of the fatjet

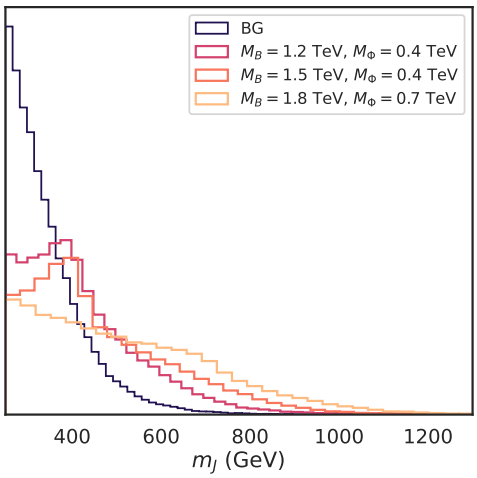
(b) $\tau_{32}^{\beta=1}$ of the fatjet



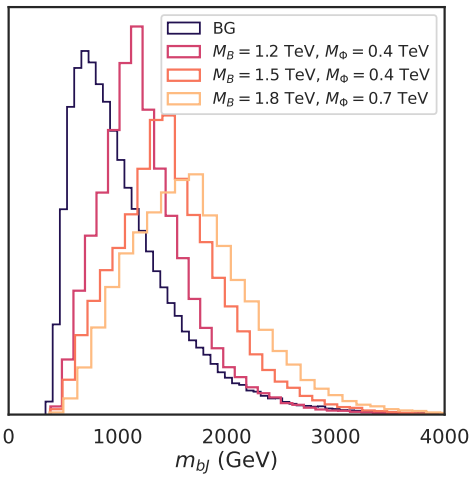
(c) $\tau_{21}^{\beta=2}$ of the fatjet

(d) $\tau_{32}^{\beta=2}$ of the fatjet

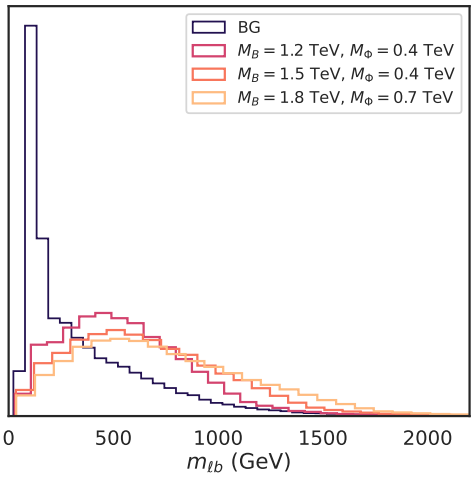
Figure 5.3: Jet Substructure Variables for the selected fatjet



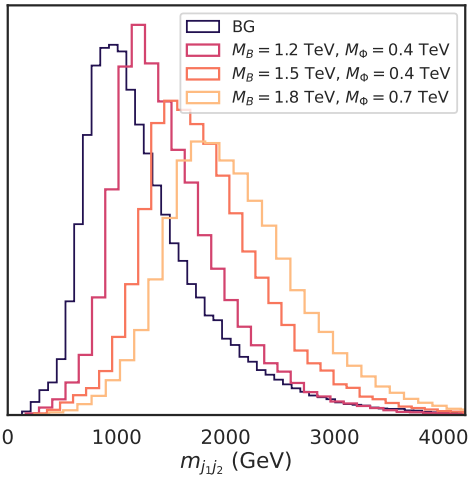
(a) Mass of the fatjet



(b) Invariant mass $m_{b,J}$ of b jet and fatjet



(c) Invariant mass ($m_{\ell b}$) of lepton and identified b jet



(d) Invariant mass $m_{j_1 j_2 j_3}$ of leading jet, subleading jet and sub-subleading jet

Figure 5.4: Select invariant mass reconstructions

reconstructed, which decay into 3 hadronic jets. Clearly, from Figs. 5.3b, 5.3d, there is no discriminating power in τ_{32}^{β}.

Mass Variables: In general, the reconstructed invariant masses are higher in the case of the signal process than in the background processes. This is due to the higher mass (energy) of the B VLQ than the mass of t or W found in the dominant background processes. From Fig. 5.4a, we can see that m_J peaks around the mass of the ϕ scalar in the case of the signal process, indicating that it is well clustered. As mentioned in [cite], we do not perform any jet filtering; therefore, in some instances, noisy top jets also appear. In the case of background processes, this reconstruction peaks around the mass of t quark in the case of processes involving t jets and noisy spread around the mass of $W/Z/H$ boson in the case of processes involving these bosons.

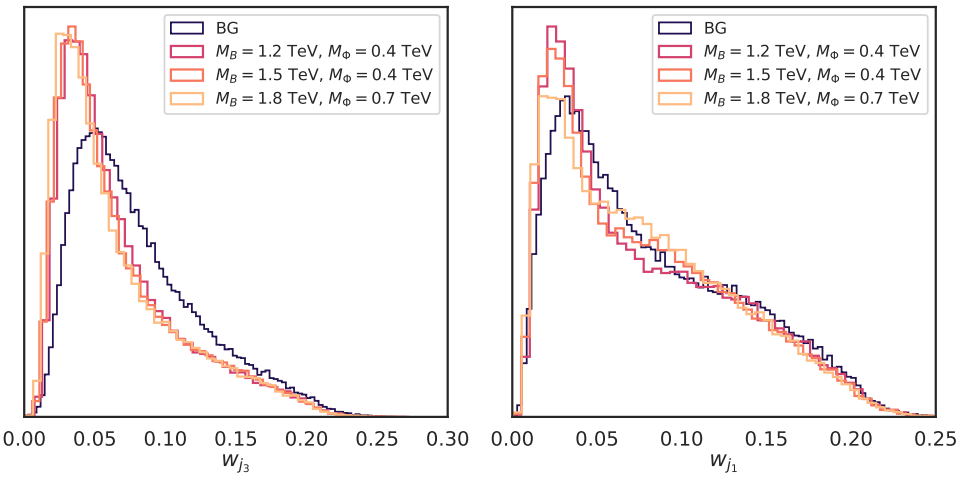
We show the distribution of m_{bJ} in Fig. 5.4b. We expect that m_{bJ} should reconstruct the invariant mass of the B VLQ, and from the figure, we can see that the reconstruction peaks close to the benchmark points. The spread around the benchmark point is largely due to two reasons, namely: (1) poor reconstruction of the ϕ fatjet and/or the b jet; and (2) the mismatch between the fatjet and b jet from different sides, which do not exactly reconstruct the B mass.

From Fig. 5.4c, we can see that for the background processes, $m_{\ell b}$ reconstructs the mass of the t jet (when the t jet decays leptonically) or reconstructs the mass of the W boson (in the case of W+jets background process). However, in the case of the signal process, there is only a small *shoulder* around the mass of t-jet, and it peaks at a much higher value.

The general trend of $m_{j_p j_q}$ and $m_{j_p j_q j_m}$ (Fig. 5.4d), where p, q, m are indices of the clustered jets, is that they peak at higher values in the signal process compared to the background processes. This can again be attributed to the higher mass (energy) of the parent particle in the signal process.

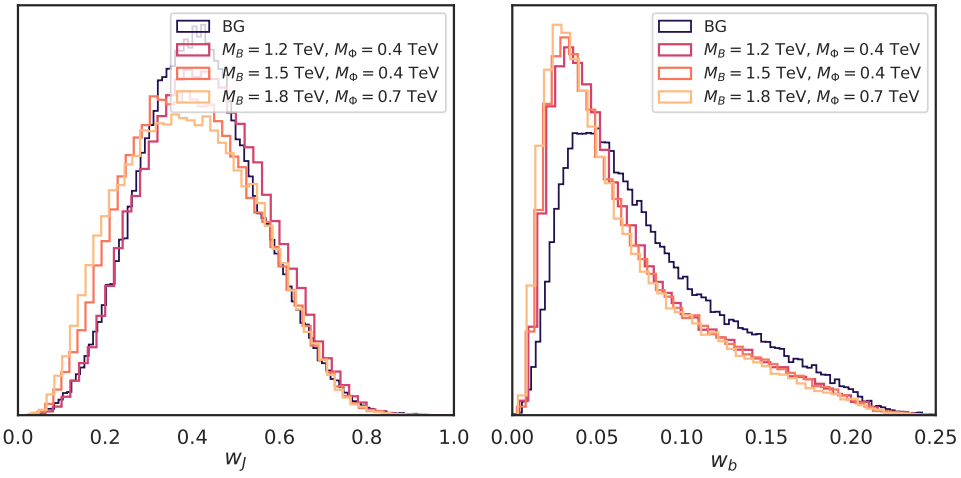
Girth/Width of jets: In Fig. 5.5a, we show the girth of the sub-subleading jet. As anticipated, since the sub-subleading jet has a higher boost in the signal process, it is more collimated and therefore exhibits lower girth as high p_T particles are closer to the jet axis. This trend is broadly true for all hadronic objects (Fig. 5.5b) in the signal process because of their higher boosts. Fig. 5.5c reveals no significant difference in the girth of the fatjets, although as the mass of B VLQ rises, we see that the signal fatjet's girth reduces since the fatjet becomes more collimated. Fig. 5.5d shows the girth of the b jet, which follows a very similar trend to that of the sub-subleading jet.

Distances in $\eta - \phi$ plane: We see a varying trend between the signal and background processes. Most of these can be explained by appreciating the different final state topologies of the signal and background processes. Fig. 5.6a demonstrates how boosted the leptonic W is for the signal process than for the background processes. Additionally, we see from the ΔR_{bJ} distribution (Fig. 5.6b) that the tagged b jet is opposite to the fatjet in background processes, but in the case of signal processes, the b jet from the same side of the fatjet may also be reconstructed. We see a distinct bimodal distribution for the background processes from Fig. 5.6c of $\Delta R_{\ell b}$. The left peak likely originates from top processes with the b jet and ℓ coming from the same leptonically decaying top jet. In this case, the other jet is reconstructed as the fatjet, and the ΔR_{bJ} requirement ensures that b is not from that fatjet. The right peak likely arises from other processes involving additional W jets or purely W jets. In the case of the signal process, most of the b jets are distant from the ℓ. We see a similar bimodal distribution in the case of $\Delta R_{j_2 j_3}$ (Fig. 5.6d) for background processes, which indicates that the two non-leading jets are clustered from the same side or the opposite side. In the case of opposite jets, this could occur in $t\bar{t}$ events when the b jet from the leptonically decaying top jet is selected as either non-leading jet. The signal distribution mainly reflects the higher number of hard jets in the final state and does not vary significantly across different benchmark points. This suggests that this feature depends more on the final state topology than kinematics.



(a) Girth of sub-subleading jet

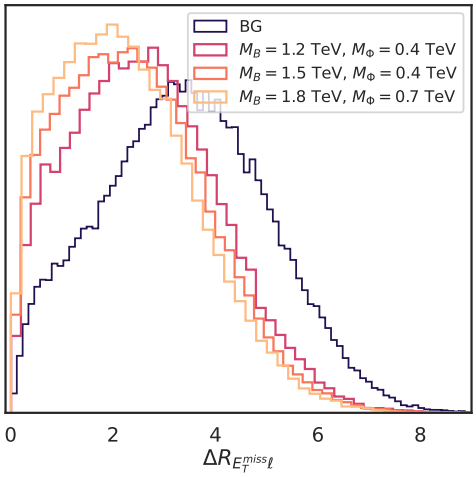
(b) Girth of the leading jet



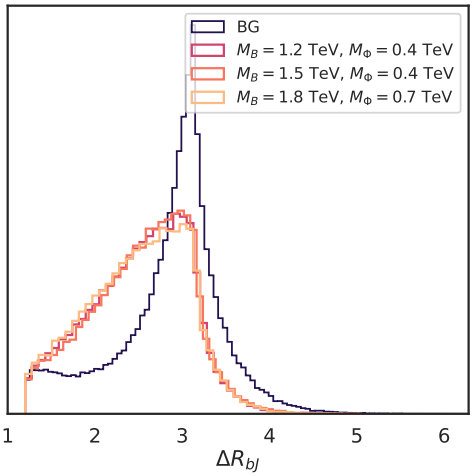
(c) Girth of the fatjet

(d) Girth of b jet

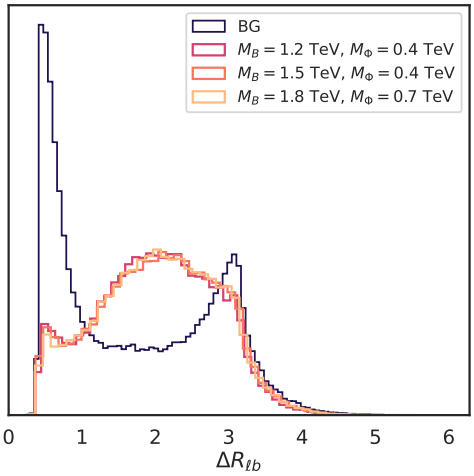
Figure 5.5: Girth of select reconstructed hadronic objects



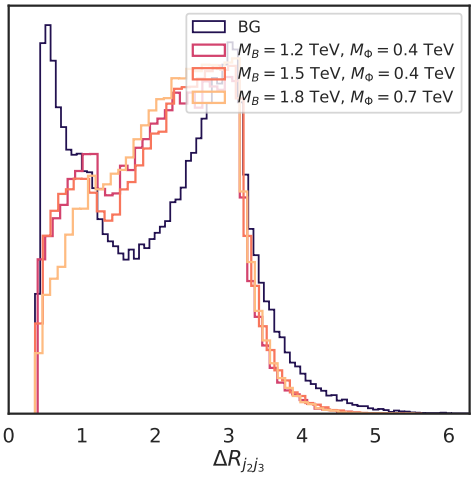
(a) Distance between MET and ℓ



(b) Distance between b jet and the fatjet

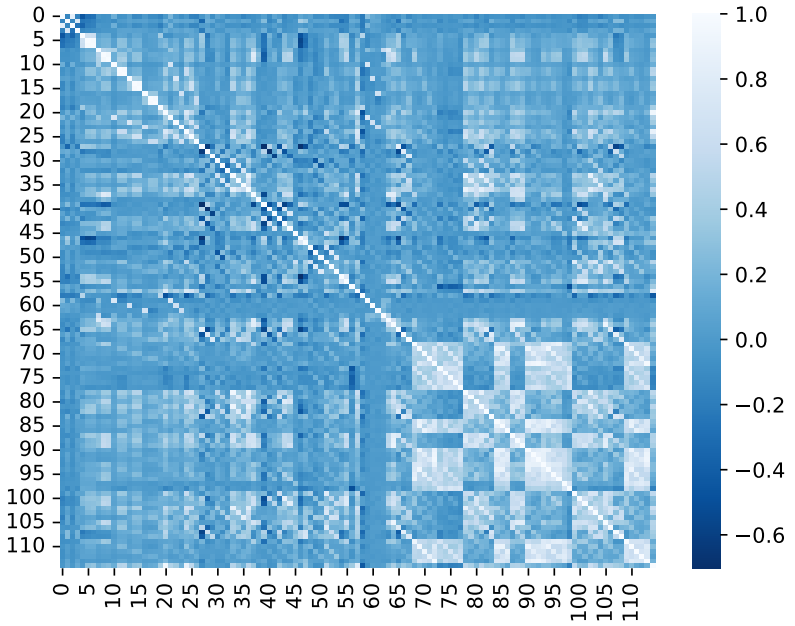


(c) Distance between the lepton and the b jet

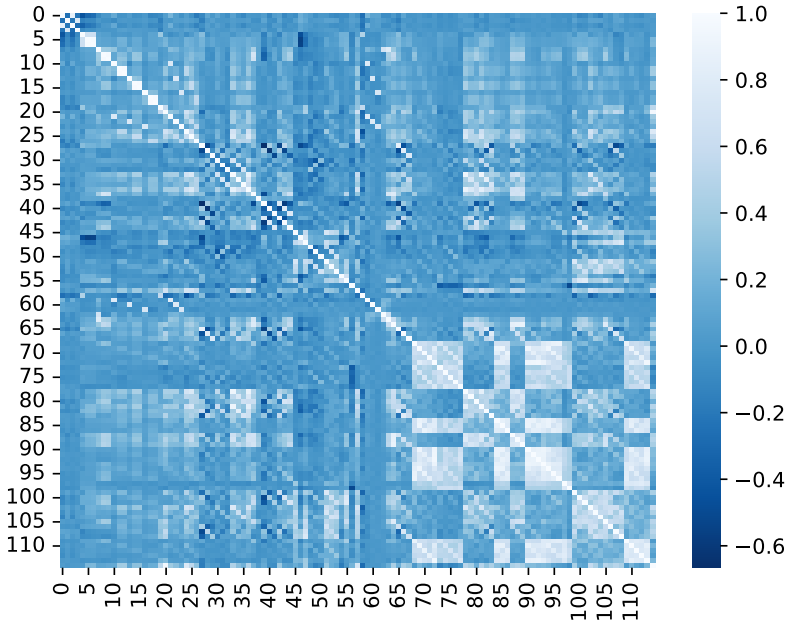


(d) Distance between the subleading jet and the sub-subleading jet

Figure 5.6: Few plots of the distance (Δ_R) between reconstructed objects



(a) Weighted Correlation plot between the reconstructed variables for the background processes. $0 - 114$ are the identifiers for each of the variables.



(b) Correlation plot between the reconstructed variables for the signal process. $0-114$ are the identifiers for each of the variables.

Figure 5.7: Correlation between the features for signal and background processes

We also show the correlation between the features in Fig. 5.7. There is a strong correlation between the Nsubjetiness features $(0-4)$ and also between the mass features $(70-110)$. However, most of the correlation values are close to 0, except some that deviate above 0.5 or below -0.5. These trends are true for both the signal and background processes. This correlation matrix is based on the Pearson coefficient, which only measures the linear relationship between pairs of features. It does not capture the complete redundancy among the features. To do so, one needs to examine the effect of each feature in a nonlinear setting (assuming further analysis is nonlinear), taking into account the effects of all features and not just pairwise interactions. In case one would like to replicate the results with a smaller number of variables (for example, to increase robustness to noise), one can refer to the following section § 5.5 on interpretability and feature importance.

5.2.4 Dataset Curation

The dataset was split into 3 parts: Training, Validation and Test with a split of 1:1:1 (i.e., $\sim 33\%$ for each part). The models were trained on the training dataset, and hyperparameter tuning was performed on the validation dataset. Finally, we calculate the results of the test dataset.

5.3 Method

In order to improve the reach, we employ a neural network to separate the signal and background processes. We benchmark the neural network's performance against the baseline of XGBoost, a modern gradient-boosted decision tree algorithm.

We perform the hyperparameter optimization for both the models at a representative with $M_B = 1.5$ TeV and $M_\phi = 0.4$ TeV. We then apply the same hyperparameters for the rest of the search points.

Note: One might argue why the first series of analysis cuts were even applied since they are far more inefficient than a neural network or a BDT. The rationale for such cuts is that they are motivated by physics considerations and ensure we target the required event topology. Furthermore, these cuts are the most explainable, interpretable and reproducible by the broader physics community.

5.3.1 Boosted Decision Tree

We use the `XGBoost` [79] library in Python to train the XGBoost classifier on the dataset comprising these events. The hyperparameter for XGBoost was chosen using `GridSearchCV`, which searches over all the possible combinations of hyperparameters. We select the hyperparameter that yields the highest significance on the cross-validation dataset. For models with

comparable performance, we prefer the simpler BDT in order to prevent overfitting. The chosen BDT has a depth of 2, with 200 estimators and a regularization weight $\lambda = 0.4$. We observe that larger BDTs give roughly the same performance. Interestingly, for models with larger depth, the hyperparameter scan preferred a larger regularization value ($\lambda = 0.6$ for a depth of 3, $\lambda = 0.8$ for a depth of 4). While parameter-heavy models with larger regularization have been shown to give better generalization performance in other domains [80], investigating such effects is beyond the scope of this work. We use Gini Index to train the Boosted Decision Tree.

5.3.2 Neural Network

We use a standard DNN architecture composed of 2 Linear layers (with layer width 128) with `Mish` [44] Activation and `BatchNorm` [48]. Additionally, we apply `Dropout` [47] with dropout probability 0.2 and L2 weight decay with $\lambda = 10^{-4}$ to regularize the training. The optimization was performed using the AdamW optimizer. We determine the hyperparameters through a grid search such that the chosen hyperparameter achieves the highest significance on the validation dataset. Similar to the BDT design, among two comparably performing networks, we choose the simpler one.

Instead of training the neural network using the standard cross-entropy loss, we find that using a weighted cross-entropy loss provides better robustness to the choice of final threshold. Since we are dealing with multiple classes of processes, we weigh the loss function to impose a higher penalty for misclassifying a sample from a process with a higher cross-section. Similarly, we re-weigh the samples to be unbiased to the explicit count of samples generated for each process. The final weight for a sample of process p_i is given by:

$$\omega_{p_i} = \sqrt{\frac{\sigma_{p_i} L}{\mathcal{N}_{p_i}}} \tag{5.2}$$

Here, σ_{p_i} denotes the cross-section of the process p_i, $\mathcal{N}_i$ is the number of events present in the training dataset and $L = 3\text{ab}^{-1}$ is the experimental luminosity. The exact functional form of the weight was determined experimentally, as the weights performed slightly better with the root than without it.

For both approaches, we scan the final response output for a threshold that maximizes the $\mathcal{Z}$ score on the validation dataset. Here we exclude those threshold points in the final response that are unstable (high variance around that point) as they do not generalize well to the test dataset. Fig. 5.8 shows that as we increase the neural network response threshold, i.e., demand a more stringent classification, N_S decreases smoothly, but N_B drops drastically.

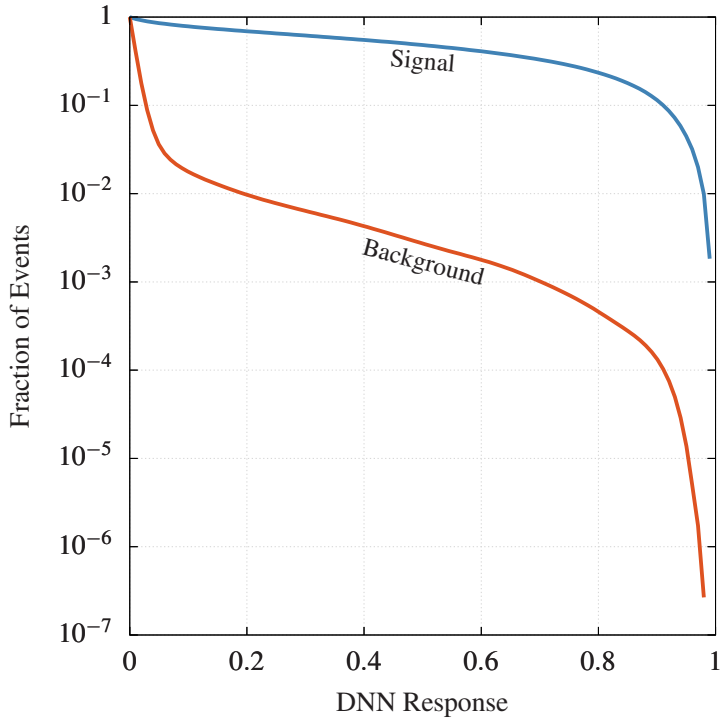


Figure 5.8: N_S, N_B curve as threshold choices for neural network

5.4 Results

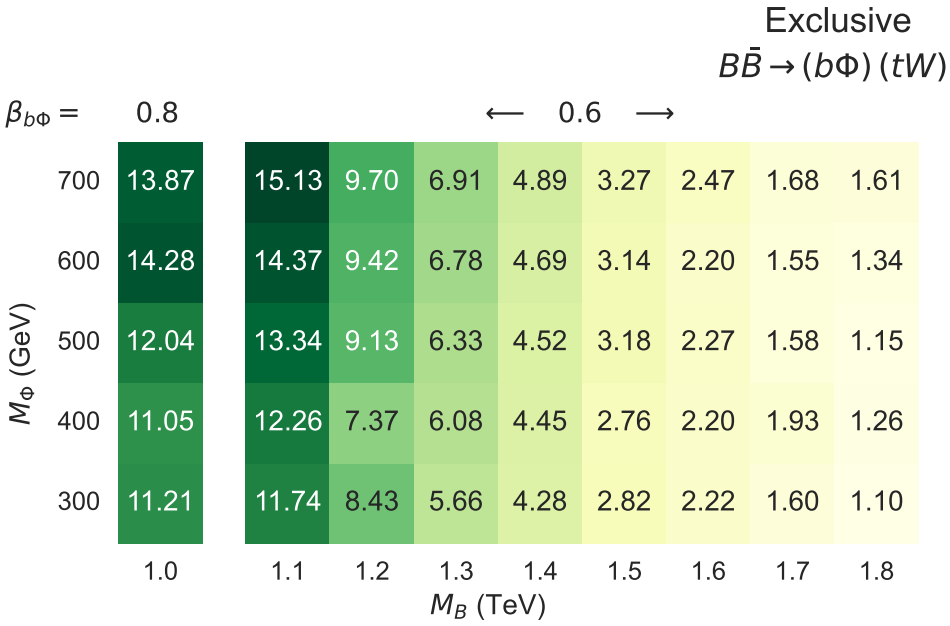
We are interested in calculating the sensitivity of the two methods. As described in section§ 3.7.1, one calculates the discovery sensitivity score using equation (3.25):

$$Z = \sqrt{2\left(N_S + N_B\right)\log\left(\frac{N_S + N_B}{N_B}\right) - 2N_S} \tag{5.3}$$

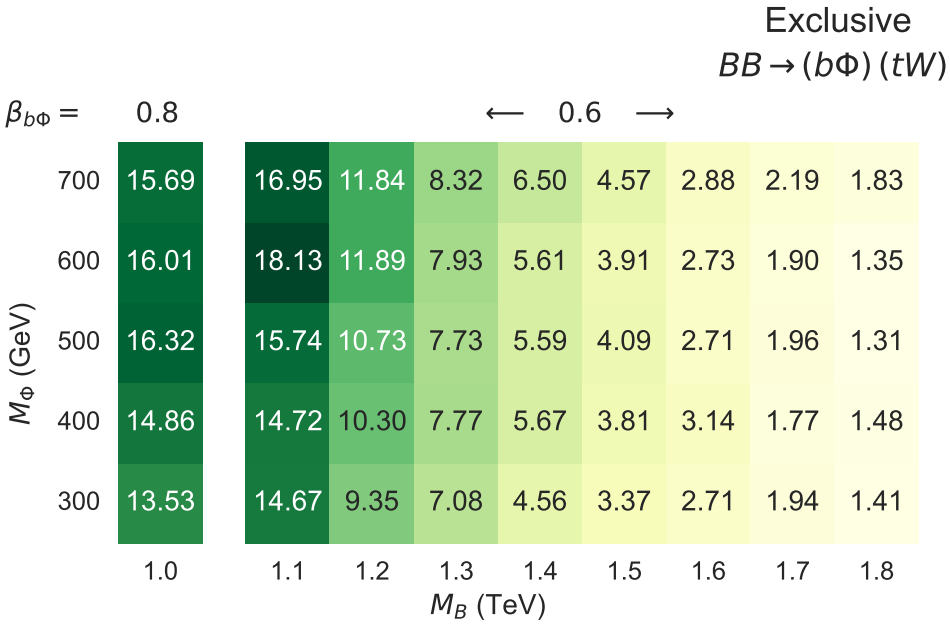
After the analysis cuts, for the benchmark point $M_B = 1.2$ TeV, $M_\Phi = 0.4$ TeV, we have $N_S = 223$, and $N_B = 335946$, resulting in an experimental discovery sensitivity of roughly 0.38σ. Therefore, we need data-driven models to make more complex cuts in order to improve the experimental sensitivity significantly.

From Fig 5.9, one can see that the neural network outperforms the BDT across all points in the $M_B - M_\phi$ scan. The absolute difference in performance decreases as we approach higher masses of M_B, which is also due to the lower pair production cross-section of the BB process.

The performance ranges from $\sim 20\%$ to $\sim 40\%$ better for the neural network than the BDT. Moreover, an interesting trend in the case of the neural network is that its performance increases as the mass of the ϕ increases, while the performance is relatively constant for the BDT.



(a) Mass scan over M_B and M_ϕ for sensitivity scores obtained by the BDT



(b) Mass scan over M_B and M_ϕ for sensitivity scores obtained by the NN

Figure 5.9: Results for BDT and NN

In this specific study, we can project the reach for such a decay process using the above methodology. The 5σ reach is up to 1.7 TeV for B mass for this specific channel. We refer the readers to [78] for more results and a comprehensive scan.

5.5 Interpretability

A key issue with neural networks is that they lack interpretability due to their highly nonlinear hidden units. This poses an issue in understanding the outcome of the network and gaining insights into the predictions. Specifically, we would like to identify input features critical in distinguishing the signal process from the background ones.

A few methods have been proposed to provide some interpretability to neural networks. Among these, the method of interest is Integrated Gradients.

5.5.1 Integrated Gradients

The idea behind Integrated Gradients [81] is that the contribution of each input feature to the model's prediction can be estimated by integrating the gradients of the model's output with respect to the input feature along a straight line path from a baseline to the input.

Integrated Gradients have the following desirable properties:

1. **Sensitivity:** If the input and the baseline have different predictions and only differ in one feature, the feature that differs should have a non-zero attribution. Also, in this case, the features that are the same are given zero attribution.

2. **Implementation Invariance:** Two networks are *functionally equivalent* if they produce identical outputs for all inputs, regardless of how they are implemented. IG satisfies *implementation invariance*, i.e., the attributions are always the same for two functionally equivalent networks.

3. **Completeness:** Integrated Gradients also satisfy a property called completeness, that the sum of the attributions is equal to the difference between the function's output at the input x and the baseline x'.

Mathematically, if we have an ML model with input $\mathbf{x}$ and output $\mathbf{y}$, the contribution of the i^{th} input feature to the model's prediction can be approximately expressed as

$$\mathfrak{G}_i^{IG}(f, \mathbf{x}, \mathbf{x}') = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\delta f\left[\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}')\right]}{\delta x_i} d\alpha \tag{5.4}$$

where $\mathbf{x}$ and $\mathbf{x}'$ represent the features of the input and the baseline input, respectively. The integral is taken from $\alpha = 0$ to $\alpha = 1$, and $(\mathbf{x} - \mathbf{x}')$ represents the vector from the baseline to the input, $\delta f\left[\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}')\right]/\delta x_i$ is the gradient of the model's output with respect to the i^{th} input feature.

Integrated Gradients is a local interpretability model i.e., it provides attributions to each feature for an individual input. However, if one fixes the baselines and averages the attributions over all the inputs in the dataset, one can obtain global attribution scores that estimate the

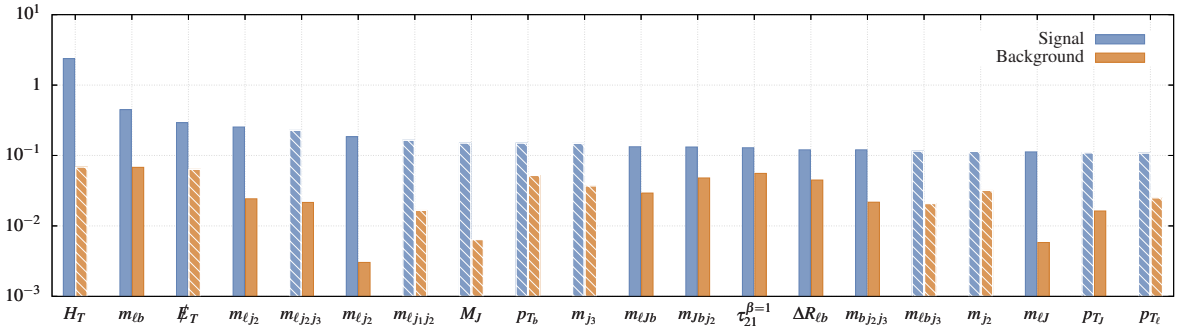


Figure 5.10: Integrated Gradients Feature Importances

relative importance of each input feature for the problem. The sign of the integrated gradient for an input feature indicates whether the feature had a positive or negative impact on the model's prediction. The magnitude indicates the relative scale of impact the feature had on the prediction compared to other features.

5.5.2 The choice of baselines and Averaged Gradients

The attribution score provided for each input is sensitive to the values that are similar or the same between the baseline and the input. While in the case of images, one can take a black image as the baseline, it is not evident what an appropriate choice of baseline would be in the case of particle physics, as the baseline needs to be physically valid in the specific problem setting.

Refs. [82, 83] propose an extension of Integrated Gradients that computes the Integrated Gradients over many baselines and averages the result. When we average over multiple baselines from the same distribution $\mathcal{D}$, we use the distribution as the baseline. So now we have changed the problem from choosing a baseline x' to choosing a distribution $\mathcal{D}$.

$$\mathfrak{G}_i(f, \mathbf{x}) = \int_{\mathbf{x}'} \mathfrak{G}_i^{IG}(f, \mathbf{x}, \mathbf{x}') \times p_D(\mathbf{x}')\, d\mathbf{x}' \tag{5.5}$$

where the baseline x' is integrated over a probability distribution on baselines D.

In our problem setting, we want to gain insights into features that are crucial in separating signal events from background events. Therefore, we take the distribution of the background events (in proportion to the cross sections of the originating processes) as the distribution of the baselines. Eq. (5.5) can be seen as computing the expectation over the set of baselines, which can be approximated from a few samples using the Monte Carlo. This method provides more robust and better attribution for each of the features.

5.5.3 Results

Fig. 5.10 shows that H_T has the highest importance in pushing the model to predict the signal class. This is expected since there is a considerable separation in H_T (Fig. 5.2c). Furthermore, $\not{E}_T$ and $m_{\ell b}$ also seem essential to the network for predicting the signal class. We also see that from the left towards the right, implying that the first few variables are enough to provide good separation. While there are some negative importance scores, they are close to 0; we can think of them as tiny correction terms to the final classifier output rather than essential contributors. The feature attribution falls only slightly for the ones following the top 10 due to the weight-decay regularisation used while training the network. For example, the 40^{th} most-important variable, $m_{\not{E}_T j_3}$, has a feature attribution of ≈ 0.056.

Chapter 6

Loss Functions for Deep Learning at the LHC

In this chapter, we investigate how to align a DNN model prediction to physics goals using targeted loss functions while training the network. We explore two avenues. First, we tweak the binary cross entropy loss with cross sections of the scattering processes. Second, we develop a loss function to maximise the sensitivity metric of a projection study, namely the Z score, using Lovàsz's extension [84]. We also compare these two classes of losses with the baseline (binary cross entropy loss) using different metrics to quantify the improvement and their robustness.

6.1 Introduction

Modern particle physics experiments at the accelerators like the LHC rely heavily on multivariate classifiers to isolate signatures of interesting processes (the signal) from well-understood ones (the background). These investigations are generally of two types: 1) measurements of known processes/properties with improving precision to check for anomalies, i.e., departures from SM and 2) looking for new processes—like looking for a hypothetical particle predicted in some new-physics theory. The main challenge comes from the fact that the signal is (extremely) rare compared to the background in most cases. Generally, a classifier has to use kinematic features to identify signal or background events. Regardless of the nature of the classifier, one generally characterises an experiment's sensitivity with a significance score. The signal plus background hypothesis (H_1) is tested against the null or background-only hypothesis (H_0), and the disagreement between them is expressed in terms of a p value. An equivalent interpretation of the p value is the significance score, Z, defined such that a Gaussian distributed variable found Z standard deviations away from its mean has a tail-distribution probability equal to p [3], as mentioned in Chapter 3. In binary classification tasks with neural networks, a widely-used metric is the BCE loss.

Here, we investigate whether other loss functions are more suitable for improving experimental sensitivity in particle physics experiments. We start with two observations. First, not all event

rates are equal; some scattering processes have higher probabilities than others. Given a set of experimental conditions, the cross section with which a scattering process occurs can be calculated using QFT. A basic BCE loss that treats all events equally might be ill-suited to such classification problems, as misclassifying events of some types could be more detrimental to the classifier performance than the other. Second, the BCE loss may not actually maximise the Z score. Instead of relying on generic classification performance metrics, can we derive a better-suited loss for any classifier which maximises the test statistic (in this case, the Z score) directly? Motivated by these observations, we investigate the following classes of losses:

1. *Process weighted Cross Entropy*:

   We assign appropriate weights to the loss of each data sample based on the effective cross-section of the corresponding process. By doing so, we make the training aware of the risk of misclassifying that particular sample, i.e., there is a greater error (risk) in misclassifying a process that is more frequent at the LHC than one that is less frequent.

2. *Surrogate Loss to maximize the Z score derived using Lovasz Extension*:

   We derive a surrogate loss in order to directly maximize the med$[Z]$ score. This extends from the insight that the Z score described in equation (3.28) is effectively a set function, and certain sets will have higher Z scores than others. Therefore, we define a loss using the Lovász extension to maximize it directly.

6.2 Process Weighted Cross Entropy Loss

Since we define the risk of misclassifying a particular kind of event, this becomes an empirical task, as the risk can depend upon the specific contribution of the sample to the task. By making the network aware of the risk of misclassifying a sample, we ensure that the classes with the highest risk are assigned values close to the desired label. This guarantees that as we scale the threshold, the classes with the highest risk are correctly classified first. While the unweighted loss might perform similarly at the extreme threshold values ($t \to 1.0$), we observe that since our method classifies the highest risk background (i.e., background event with the largest cross-section) closest to 0, there is a large range over which the model achieves a sensitivity score close to the maximum.

We can estimate the cross-sections from theory and predict the data distribution with a high degree of accuracy, so we can quantify the "risk" as a function of the rate of occurrence at the LHC. We can simply derive a simple weighing scheme to match the risk of a sample to its rate of production at LHC. However, as this remains an empirical task, we experiment with different weights to see which performs the best. In cases where we do not know the exact cross-sections or

where there is a large disparity between the risks, one may prefer to train an unbiased classifier due to its better convergence properties.

6.2.1 Different Weighting Schemes

We examine the performance of different weighting schemes, which can be categorized as follows:

WC1 (Choices related to the dataset size): Generally, some processes are much easier to generate compared to others and therefore are present in larger numbers in the training dataset, creating large class imbalances. These choices are used to reduce the bias on the exact number of samples from each process in the dataset and address the class imbalances. The weighing schemes take the form:

$$\omega_1 \in \left\{ \frac{1}{\mathcal{N}_i}, \sqrt[n]{\frac{1}{\mathcal{N}_i}}, \frac{1}{\ln \mathcal{N}_i} \right\}, \qquad n \in \mathbb{R} \tag{6.1}$$

WC2 (Choices related to the cross-section of the process):] By providing the cross-section information, we would inform the training about the risk of misclassifying a sample as a function of the production rate of that process. Here the weighting scheme takes the form:

$$\omega_2 \in \{\sigma_i, \sqrt[n]{\sigma_i}, \ln(\sigma_i), \mathbb{U}(\sigma_i)\}, \qquad n \in \mathbb{R} \tag{6.2}$$

where $\mathbb{U}$ is described as,

$$\mathbb{U}(\sigma_i) = \begin{cases} \sigma_i \times 10^{\left\lfloor \log_{10}\left(\frac{\max_{j\in B}\{\sigma_j\}}{\sigma_i}\right)\right\rfloor}, & \text{if } i \in S \\ \sigma_i, & \text{otherwise} \end{cases}$$

where S, B denote the signal and background processes, respectively.

We try not only individual choices from **WC1** and **WC2** but also their combinations. We note a few specific cases, $\omega = \frac{1}{\mathcal{N}_i}$ is the weighting scheme used for developing unbiased estimators, and $\omega = \frac{\sigma_i}{\mathcal{N}_i}$ weighs the sample of a process such that it matches its rate of production at LHC.

6.3 Surrogate med$[Z]$ score loss

One wants to maximize the med$[Z]$-score (referred to as Z-score for the remainder of this chapter) given in equation (3.25), which corresponds to the sensitivity of the experiment. In analyses involving searches for new particles, we typically have $N_s << N_b$ and the sensitivity is optimized by maximizing $N_s/\sqrt{N_b}$. Although an excellent binary cross entropy-based classifier

might give a good sensitivity, it is not guaranteed optimal. Therefore, we wonder whether we can train a classifier to directly maximize the relevant metric, i.e., the sensitivity score of the method.

There are two hurdles that one will have to resolve in order for the approach to work with gradient-based methods such as machine learning:

1. The metric Z-score is a non-differentiable function as it depends upon discrete quantities. Therefore we need to develop some kind of continuous interpolation for the function.

2. The metric operates on sets of data instead of individual samples; specifically, it operates on the count data. Therefore, one must either develop a method to directly optimize the set function or assign contributions to specific samples within the set to optimize.

6.3.1 Submodular Functions and Lovasz Extension

A hint towards the solution comes from the domain of discrete optimization, specifically submodular functions. A submodular function is a function that captures the concept of diminishing returns. It is defined on sets and has a property similar to concavity. Formally one defines a submodular function as:

Definition 6.3.1. Submodularity: A set function $\Delta : 2^N \rightarrow \mathbb{R}$, where N is a finite set, is submodular if it satisfies the following condition: For $A, B \subseteq N$, if $A \subseteq B$, and $x \notin B$, then

$$\Delta(A \cup \{x\}) - \Delta(A) \geq \Delta(B \cup \{x\}) - \Delta(B) \tag{6.3}$$

or equivalently, for all sets $A, B \subseteq N$,

$$\Delta(A) + \Delta(B) \geq \Delta(A \cup B) + \Delta(A \cap B) \tag{6.4}$$

where N is the Universe set, and 2^N refers to the power set.

Due to this property of submodular functions, they can be optimized using greedy optimization techniques, and optimal solutions may be reached in polynomial time. However, these discrete optimization techniques cannot be used directly due to the lack of gradients for descent-based methods.

For the next section on continuous relaxation, we alternatively represent the powerset as $\{0, 1\}^p$, where p is the number of elements in the Universe and the value 0 or 1 for an element indicates its inclusion in a specific subset.

Lovász Extension allows us to associate a continuous, convex function with any submodular function. The Lovász extension for any submodular function is defined as

Definition 6.3.2. Lovász Extension: For a set function $\Delta : \{0,1\}^p \to \mathbb{R}$, the Lovász extension $\bar{\Delta} : [0,1]^p \to \mathbb{R}$ is defined as

$$\bar{\Delta} : \boldsymbol{m} \in \mathbb{R}^p \mapsto \sum_{i=1}^{p} m_i \; g_i(\boldsymbol{m}) \tag{6.5}$$

where $g_i(\boldsymbol{m}) = \Delta(\{\pi_1, \ldots, \pi_i\}) - \Delta(\{\pi_i, \ldots, \pi_{i-1}\})$ and $\boldsymbol{\pi}$ is a permuation ordering the components of $\boldsymbol{m}$ in decreasing order, i.e., $x_{\pi_1} \geq x_{\pi_2} \geq \cdots \geq x_{\pi_p}$ [85]. Usually, $\mathbf{m}$ is the vector of errors that, i.e., $\mathbf{m_i}$ is the deviation for element i from the label (0 or 1).

Additionally, the Lovasz extension of a submodular function preserves submodularity, i.e., the extension evaluated at the points of the hypercube still follows submodularity. Using the Lovász extension, we can directly compute the tight convex closure of a submodular function within polynomial time [$\mathcal{O}(p \log(p))$ time complexity]. This convex extension is amenable to a host of efficient optimization methods, especially gradient-based approaches.

6.3.2 Z-score as a submodular function

To view the Z score as a set function, we define some sets. Let V_i be the set of events of a process i where $i \in S \cup B$, where S, B are the sets of signal and background processes, respectively. Let y be the ground truth labels of a set of events and $\tilde{y}$ be the labels predicted by the method for the set of events. We define P_y to be the set of positive labels, i.e., $y = 1$, and $P_{\tilde{y}}$ be the set of positive predictions, i.e., $\tilde{y} = 1$. We now have the Z score,

$$Z \approx \frac{N_S}{\sqrt{N_B}} = \frac{\sum_{i \in S} \frac{|V_i \cap P_{\tilde{y}}|}{|V_i|} \sigma_i \mathcal{L}}{\sqrt{\sum_{i \in B} \frac{|V_i \cap P_{\tilde{y}}|}{|V_i|} \sigma_i \mathcal{L}}} \tag{6.6}$$

Now we may rewrite the above equation in terms of a set of misclassifications. Note that v_i is the number of events of each process, n_i is the number of false negatives for process $i \in S$, and p_i is the number of false positives for process $i \in B$. Then for a set of misclassifications $(\mathbf{n}, \mathbf{p})$, we have:

$$Z(y, \tilde{y}) = \frac{\sum_{i \in S} \frac{v_i - n_i}{v_i} \sigma_i \mathcal{L}}{\sqrt{\sum_{i \in B} \frac{p_i}{v_i} \sigma_i \mathcal{L}}} \tag{6.7}$$

Now we make three changes in order to adapt the above term for gradient minimization:

1. First, we introduce a small ϵ term in the denominator to ensure numerical stability.
2. We take the negative of the Z score so that we can minimize it using inbuilt gradient descent algorithms (§ 4.5.2).
3. We additionally demand that the modified function evaluates to 0 for an empty set.

Based on these changes, we define a modified function Δ_Z based on the Z-score.

$$\Delta_Z(y, \tilde{y}) = \sum_{i \in S} \frac{\sigma_i \mathcal{L}}{\sqrt{\epsilon}} - \frac{\sum_{i \in S} \frac{v_i - n_i}{v_i} \sigma_i \mathcal{L}}{\sqrt{\epsilon + \sum_{i \in B} \frac{p_i}{v_i} \sigma_i \mathcal{L}}} \tag{6.8}$$

Theorem 6.3.3. *Δ_Z is submodular on the set of misclassifications $(\mathbf{n}, \mathbf{p})$, where $\mathbf{n}$ is the vector for number of false negatives (n_i), and $\mathbf{p}$ is the vector of number of false positives (p_i).*

Proof:

For the proof, we take the scenario with a single signal process ($|S| = 1$, $\mathbf{n} = n_1 = n$) and a single background process ($|P| = 1$, $\mathbf{p} = p_1 = p$) to simplify the expressions. But the result can be easily extended to incorporate multiple signal and background processes due to the linearity of additional signal processes and background processes. We will also drop the luminosity term as that will not affect the core derivation.

For the proof, let us assume that we have two sets of misclassifications C (n_C, p_C) and D (n_D, p_D), such that $D \subseteq C$, i.e.,

$$D \subseteq C, \quad n_D \leq n_C, \quad p_D \leq p_C \tag{6.9}$$

The total number of events remains the same between C and D, i.e., v_S for signal and v_B for background, and only the misclassifications on the total set change.

To establish the proof, we need to show that the diminishing return property of submodularity holds under the addition of a new element $i \notin C$.

Case I: Adding false negatives $i \notin C$

We prove that Δ_Z is submodular under the addition of false negatives:

$$\Delta_Z(C \cup \{i\}) = \Delta_Z(n_C + 1, p_C) \tag{6.10}$$

$$= \frac{\sigma_S}{\sqrt{\epsilon}} - \frac{\frac{v_S - n_C - 1}{v_S} \sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B} \sigma_B}} \tag{6.11}$$

$$= \left(\frac{\sigma_S}{\sqrt{\epsilon}} - \frac{\frac{v_S - n_C}{v_S} \sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B} \sigma_B}} \right) + \frac{\frac{1}{v_S} \sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B} \sigma_B}} \tag{6.12}$$

$$= \Delta_Z(C) + \frac{\frac{1}{v_S} \sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B} \sigma_B}} \tag{6.13}$$

$$\Delta_Z(C \cup \{i\}) - \Delta_Z(C) = \frac{\frac{1}{v_S} \sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B} \sigma_B}} \tag{6.14}$$

Now since $D \subseteq C$, i.e., $p_D \leq p_C$, we see from equation (6.14),

$$\Delta_Z(C \cup \{i\}) - \Delta_Z(C) \leq \Delta_Z(D \cup \{i\}) - \Delta_Z(D), \quad i \text{ is a false negative} \tag{6.15}$$

Case II: Adding a false positive $i \notin C$

We prove that Δ_Z is submodular under the addition of false positives:

$$\Delta_Z(C \cup \{i\}) = \Delta_Z(n_C, p_C + 1) \tag{6.16}$$

$$= \frac{\sigma_S}{\sqrt{\epsilon}} - \frac{\frac{v_S - n_C}{v_S}\sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B}\sigma_B + \frac{1}{v_B}\sigma_B}} \tag{6.17}$$

$$\tag{6.18}$$

Now we have,

$$\Delta_Z(C \cup \{i\}) - \Delta_Z(C) = \frac{\frac{v_S - n_C}{v_S}\sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B}\sigma_B}} - \frac{\frac{v_S - n_C}{v_S}\sigma_S}{\sqrt{\epsilon + \frac{p_C}{v_B}\sigma_B + \frac{1}{v_B}\sigma_B}} \tag{6.19}$$

$$= \underbrace{\left(\frac{v_S - n_C}{v_S}\sigma_S\right)}_{T_1} \underbrace{\left[\frac{1}{\sqrt{\epsilon + \frac{p_C}{v_B}\sigma_B}} - \frac{1}{\sqrt{\epsilon + \frac{p_C}{v_B}\sigma_B + \frac{1}{v_B}\sigma_B}}\right]}_{T_2} \tag{6.20}$$

Therefore it decomposes into a product of two terms. If we show that independently both of these terms are independently smaller for C than for D, we will have our result.

First consider T_1, we have

$$n_C \geq n_D \tag{6.21}$$

$$= -n_C \leq -n_D \tag{6.22}$$

$$= \frac{v_S - n_C}{v_S} \leq \frac{v_S - n_D}{v_S} \tag{6.23}$$

Therefore, T_1 is indeed larger for D compared to C.

In order to check for term T_2, we first simplify the expression and write $H_C = \epsilon + \frac{p_C}{v_B}\sigma_B$, ($H_C \geq H_D$). Now we can write term two as:

$$\frac{1}{\sqrt{H_C}} - \frac{1}{\sqrt{H_C + \frac{1}{v_B}\sigma_B}} \tag{6.24}$$

We move to a continuous relaxation of the term such that:

$$f(x) = \frac{1}{\sqrt{x}} - \frac{1}{\sqrt{x + \frac{1}{v_B}\sigma_B}} \tag{6.25}$$

$$f(H_C) = \frac{1}{\sqrt{H_C}} - \frac{1}{\sqrt{H_C + \frac{1}{v_B}\sigma_B}} \tag{6.26}$$

which is the same as equation (6.24). Now differentiating equation (6.25) with respect to x, we get:

$$\frac{\mathrm{d}}{\mathrm{d}x} f = \frac{1}{2}\left(\frac{1}{\left(x + \frac{1}{v_B}\sigma_B\right)^{\frac{3}{2}}} - \frac{1}{(x)^{\frac{3}{2}}}\right) \tag{6.27}$$

which will always be less than zero for $x > 0$. Thus since $\frac{\mathrm{d}}{\mathrm{d}x} f < 0$, we have that T_2 will be greater for D compared to C.

Now since both T_1 and T_2 is greater for D compared to C, we have

$$\Delta_Z(C \cup \{i\}) - \Delta_Z(C) \leq \Delta_Z(D \cup \{i\}) - \Delta_Z(D), \quad i \text{ is a false positive} \tag{6.28}$$

Therefore from **Case I** and **Case II**, we have shown that Δ_Z is submodular for all the possible cases and therefore is submodular for the set of misclassifications $(\mathbf{n}, \mathbf{p})$. □

We posit the task of maximizing the Z score as minimizing the above submodular term Δ_Z. Using the submodular property of the term, we can create a Lovasz extension ($\bar{\Delta_Z}$) that can be used to train a deep neural network through gradient-based optimization.

6.3.3 Error Functions

We require a loss function to handle any vector of errors $\mathbf{m} \in \mathbb{R}_+^p$ since we are working with continuous predictions, not only to discrete vectors of misclassifications in $\{0, 1\}^p$. We consider four cases for defining the vector of errors $\mathbf{m}$ to construct the surrogate losses using the Lovasz extension.

1. **Hinge (Max Margin) Loss:** Following Ref. [86], we implement a hinge loss to compute the error in the prediction. The labels are considered signed ($y_i \in \{-1, 1\}$). The model outputs a score $F_i(x)$ for each sample x. The error is given by the hinge loss,

$$m_i = \max(1 - F_i(x) y_i, 0), \qquad y_i \in \{-1, 1\}. \tag{6.29}$$

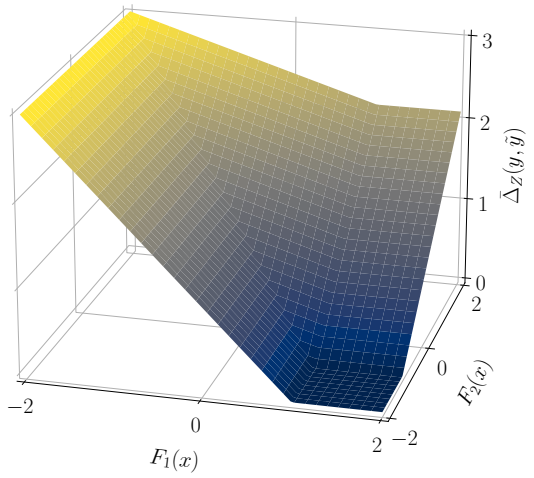
2. **Sigmoid Error**: Similar to Ref. [85], we also consider the sigmoid error. The model outputs a probability $F_i(x)$ for the sample x to be in the signal class. The error is given by

$$m_i = \begin{cases} 1 - F_i(x), & \text{if } y_i = 1, \\ F_i(x), & \text{otherwise.} \end{cases} \tag{6.30}$$

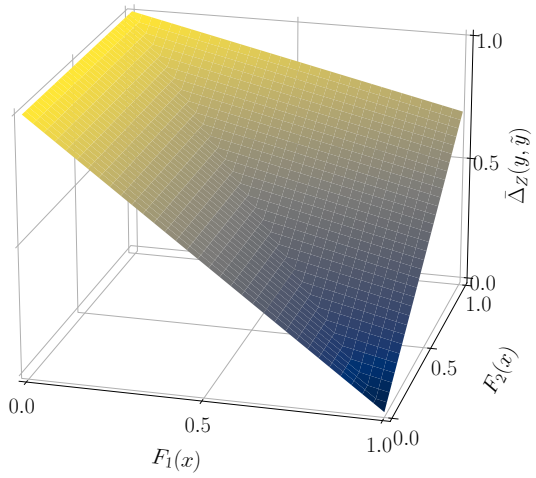
3. **Cross Entropy Error**: We also experiment with the BCE loss to measure the error m_i. This is similar to taking the logarithm of the error calculated in the Sigmoid Error. One could also interpret this as a form of weighted cross entropy where the weights are calculated based on the specific composition of the batch of events and misclassifications on that batch.

4. **Focal Loss Error**: We also consider Focal Loss [87] as the measure for the error m_i. This loss function drives the network to focus on hard misclassified events.

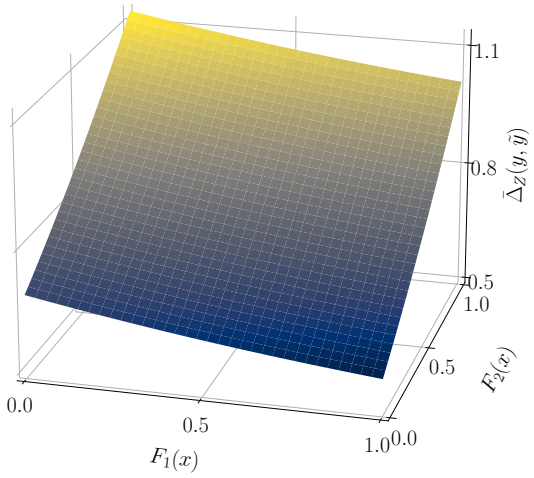
Algorithm 1 provides a simple pseudocode to calculate the gradient $g(\boldsymbol{m})$ from Eq. (6.5) using Eq. (6.8) as the loss.



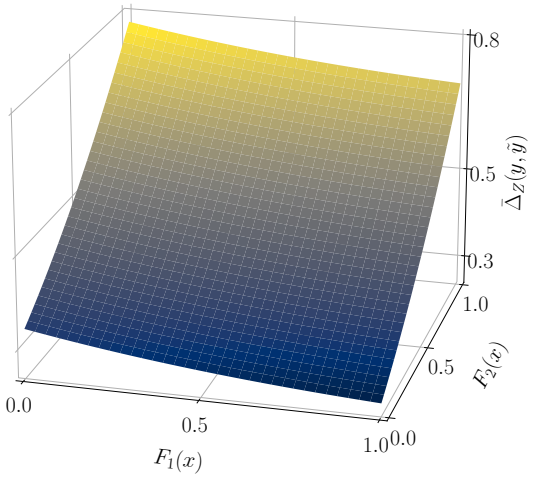
(a) Hinge Error



(b) Sigmoid Error



(c) Cross Entropy Error



(d) Focal Loss Error

Figure 6.1: Loss landscapes for the four error measures $\boldsymbol{m}$ in Sec.§ 6.3.3. The Z score loss is plotted with ground truth, $GT = [1, 0]$, $\sigma = [1, 10]$. The x, y axes denote the classifier output $(F_1(x), F_2(x))$. For the Hinge Error, the GT labels are converted to their signed equivalent.

Algorithm 1 Gradient of Lovász Z loss $\bar{\Delta}_Z$

Require: vector of errors $\boldsymbol{m} \in \mathbb{R}^p_+$, ground truth labels $\boldsymbol{\delta}$, sample weights $\boldsymbol{w} = \{w_1, w_2, \ldots, w_p\}$ calculated from σ_i and counts.
Ensure: $g(\boldsymbol{m})$ gradient of $\bar{\Delta}_Z$ from Equation (6.5)

$\boldsymbol{\pi} \leftarrow$ decreasing sort permutation for $\boldsymbol{m}$
$\boldsymbol{\delta_\pi} \leftarrow (\delta_{\pi_i})_{i \in [1,p]}$
numerator $\leftarrow 1$ - **cumulative_sum**$(\boldsymbol{\delta_\pi})\ \boldsymbol{w}$
denominator $\leftarrow 1$ + **cumulative_sum**$(\mathbf{1} - \boldsymbol{\delta_\pi})\ \boldsymbol{w}$
$\boldsymbol{g} \leftarrow \sigma -$ **numerator**$/\sqrt{\textbf{denominator}}$
if $p > 1$ **then**
    $\boldsymbol{g}[2:p] \leftarrow \boldsymbol{g}[2:p] - \boldsymbol{g}[1:p-1]$
end if
return $\boldsymbol{g_\pi}$

We also investigate deriving a surrogate loss from the med$[Z]$-score given in equation (3.25). Similar to equation (6.8), we redefine the equation for gradient minimization, which we refer to as Δ_ζ. For the case of a single signal and single background process, i.e, $|S| = |B| = 1$ and $\mathbf{n} = n_1 = n$, $\mathbf{p} = p_1 = p$, the redefined equation is

$$\Delta_\zeta(y, \tilde{y}) = C - \sqrt{2\left(\epsilon + \frac{v_S - n}{v_S}\sigma_S + \frac{p}{v_B}\sigma_B\right)\ln\left(1 + \frac{\frac{v_S - n}{v_S}\sigma_S}{\epsilon + \frac{p}{v_B}\sigma_B}\right) - 2\left(\frac{v_S - n}{v_S}\sigma_S\right)} \tag{6.31}$$

$$C: \text{ is a constant such that } \Delta_\zeta(\emptyset) = 0$$

Lemma 6.3.4. *The Δ_ζ loss function is not submodular on the set of misclassifications $(\mathbf{n}, \mathbf{p})$, where similar to equation (6.8), $\mathbf{n}$ is the set of false negatives and $\mathbf{p}$ is the set of false positives.*

Proof: For the proof, we take the scenario with a single signal process ($|S| = 1$, $\mathbf{n} = n_1 = n$) and a single background process ($|P| = 1$, $\mathbf{p} = p_1 = p$) in order to simplify the expressions. Let us assume that we have two sets of misclassifications C (n_C, p_C) and D (n_D, p_D), such that $D \subseteq C$, i.e.,

$$D \subseteq C, \quad n_D \leq n_C, \quad p_D \leq p_C \tag{6.32}$$

The total number of events remains the same between C and D, i.e., v_S for signal and v_B for background, and only the misclassifications on the total set change.

Now, we observe the case of adding a false negative to Δ_ζ, and checking for diminishing returns,

$$\Delta_\zeta(C \cup \{i\}) - \Delta_\zeta(C) = \Delta_\zeta(n_C + 1, p_C) - \Delta_\zeta(n_C, p_C) \tag{6.33}$$

For brevity, we do not expand the above terms, but one can now take derivative with respect to n and p and plot it over the entire domain of n, and p, i.e. v_S and v_B, respectively. We plot

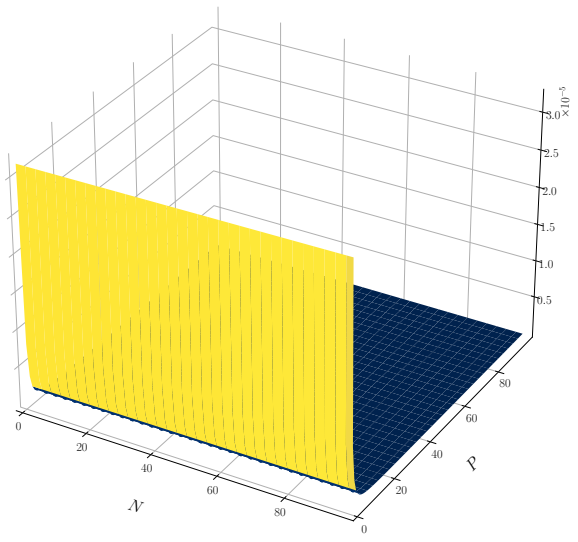


Figure 6.2: The plot of $\frac{\partial}{\partial p}\frac{\partial}{\partial n}\left(\Delta_\zeta(n_C+1,p_C)-\Delta_\zeta(n_C,p_C)\right)$. The value is positive, indicating that the diminishing return property does not hold. But the values are very close to zero.

the same in Fig. 6.2, where we see that the derivative is positive, showing that the diminishing returns property does not hold. However, we see that the value is very close to zero near the central regions indicating that it may still be viable to employ this as a potential loss function.

Additionally, Δ_ζ is submodular under the addition of a false positive sample. One can still define a continuous extension using the Lovasz extension and optimize it. We test it for one case and find the performance is very similar, if not identical, (Fig. 6.5) between Δ_ζ and Δ_Z. □

6.4 Setup and Evaluation

6.4.1 Dataset Construction

We choose a typical particle physics experiment searching for a new phenomenon to generate the dataset. We consider the task of isolating events from inelastic scattering processes producing two vectorlike quarks (VLQs) from the background processes with similar final states. Particularly we study the $pp \to BB$ process (introduced in § 5.1). The reason for choosing this particular example is that its cross-section is expected to be small compared to the dominant background processes. Since the focus of this study is to evaluate the performance of the different losses and not perform a thorough physics study, we consider only a few dominant background processes — namely, we consider the semileptonic decay of $pp \to tt$ process (BG1), and the monoleptonic decay of $pp \to ttW$ process as the backgrounds (BG2), (see Table 6.1)

Process	Initial σ (fb)	Event Sel. σ (fb)	Dataset Size	TTG (Server)
$pp \to BB$ (Signal)	0.2171	0.0739	420k	$\sim$ 24 hours
$pp \to tt$ (BG1)	2.9×10^5	91.23	430k	$\sim$ 7 days
$pp \to ttW$ (BG2)	1.92×10^2	0.0231	420k	$\sim$ 3 days

Table 6.1: Details of the various processes used for the current experiments: their initial cross sections (in fb), their cross sections after event selection criteria are applied (in fb), the number of samples in the dataset, and the time taken to generate (TTG) these samples on a server grade PC running Xeon processors.

The data generation and preprocessing pipeline remain the same as in § 5.2.1. We employ `MADGRAPH5` [20] to generate events for the study. We employ `PYTHIA8` [21] for parton showering and hadronisation. We use a detector simulator `DELPHES3` [12] to replicate LHC detectors where all the final state objects are detected. We also apply the same analysis cuts specified in § 5.2.2 on the processes to extract "signal-like" events.

6.4.2 Deep Learning Model Construction

Since we want to explore the effect of the loss function, we have chosen a simple DNN architecture with 2 layers (each with 128 neurons). After each hidden layer, the activation function is `Mish` [44] activation. We use `BatchNorm` [48] to stabilise the training process. We also use `Dropout` [47] with probability 0.2, and weight decay with $\lambda = 10^{-4}$ to regularize the training. We train the network using `AdamW` [88] optimizer for 50 epochs, after which we observe no change in the result. Significant hyperparameter tuning was not performed as that is not relevant to the motivation of the solution.

6.4.3 Evaluation Metrics

An ideal classifier would result in the maximum med$[Z]$ (sensitivity) score ((3.25)) for the broadest range of the classifier response. With this in mind, we define the following metrics to evaluate the performance of the proposed loss functions.

- Maximum Z score $[\max(Z)]$: We scan over the various classifier thresholds and report the maximum Z score the classifier achieves. However, this metric does not capture the complete nature of the Z score vs classifier response curve — it is susceptible to sharp peaks in the response curve, which are undesirable.

- Multiple working points (WPs): We pick various WPs of classifier thresholds to report the Z score. The classisifer thresholds for these WPs are 0.3 (WP1), 0.5 (WP2), 0.8 (WP3), 0.95 (WP4). These choices correspond to increasing levels of strictness for signal classification.

One can also get an idea of the flatness of the curve by observing the trends between all the working points.

- Area under the curve (AUC): We calculate the area under the Z-score vs classifier response curve. For an ideal classifier, the area under the curve would equal the maximum Z score (i.e., the maximum Z score is achieved at a low level of classifier thresholds and remains the same for the entire range of classifier responses).

- Region within 10% of $\max(Z)$ [$R_{10\%}$]: Using this metric, one can identify models that perform close to their peak performance over a wide region of threshold choice. A classifier with large regions within 10% of the best performance will essentially eliminate the need to pick a good threshold.

Moreover, when calculating the Z score, we demand that the signal and background classes contain a minimum number of events to be statistically significant. We choose this threshold to be 10 events (i.e., $N_S, N_B \geq 10$).

6.4.4 Test Scenarios

We evaluate the performances of the proposed losses on some simple test cases (TCs) for different class-imbalance scenarios and computational constraints.

- *TC1: Full Dataset*

  In this scenario, the number of training events in each class is roughly equal, and there are sufficient training samples in all classes to learn from.

- *TC2: Reduced training samples in the background class*

  This test examines how the losses perform when the classifier has a small number of event samples for processes with higher probability. Only 10% of the training samples in the background class are used to train, while the complete set of signal samples is used. Such a scenario arises when one has limited computing resources to generate a large number of background events, especially ones that resemble the signal.

- *TC3: Reduced training samples in both signal and background class*

  We also study whether the proposed losses can help compensate for a severe lack of samples to learn from. Specifically, we take only 10% of the training samples in signal and background classes to train the network. This test case is an extension of *TC2* and can be mapped to a study performed under extremely stringent computing constraints.

Loss	$\max(Z)$	WP1(0.3)	WP2(0.5)	WP3(0.8)	WP4(0.95)	AUC	$R_{10\%}$
BCE	$6.08_{\pm 0.09}$	$1.57_{\pm 0.01}$	$1.93_{\pm 0.02}$	$2.79_{\pm 0.04}$	$4.28_{\pm 0.09}$	$2.11_{\pm 0.02}$	$0.01_{\pm 0.00}$
Weighted CE loss based on dataset size (ω_1)							
$1/\mathcal{N}_i$	$6.00_{\pm 0.19}$	$1.58_{\pm 0.03}$	$1.93_{\pm 0.03}$	$2.77_{\pm 0.07}$	$4.23_{\pm 0.08}$	$2.11_{\pm 0.04}$	$0.01_{\pm 0.00}$
$1/\sqrt{\mathcal{N}_i}$	$6.09_{\pm 0.06}$	$1.57_{\pm 0.01}$	$1.93_{\pm 0.02}$	$2.78_{\pm 0.04}$	$4.26_{\pm 0.11}$	$2.11_{\pm 0.03}$	$0.01_{\pm 0.00}$
$1/\ln(\mathcal{N}_i)$	$6.09_{\pm 0.12}$	$1.57_{\pm 0.01}$	$1.93_{\pm 0.02}$	$2.79_{\pm 0.05}$	$4.28_{\pm 0.11}$	$2.11_{\pm 0.03}$	$0.01_{\pm 0.00}$
Process-aware CE losses, characterised by $\omega_2/\mathcal{N}_i$							
σ_i	$5.54_{\pm 0.30}$	$5.22_{\pm 0.23}$	$4.89_{\pm 0.40}$	$1.73_{\pm 2.00}$	$0.00_{\pm 0.00}$	$3.87_{\pm 0.41}$	$0.36_{\pm 0.09}$
$\sqrt[2]{\sigma_i}$	$6.48_{\pm 0.30}$	$3.53_{\pm 0.10}$	$4.34_{\pm 0.16}$	$5.20_{\pm 0.56}$	$6.27_{\pm 0.50}$	$4.17_{\pm 0.06}$	$0.13_{\pm 0.07}$
$\sqrt[4]{\sigma_i}$	$6.43_{\pm 0.11}$	$2.33_{\pm 0.02}$	$2.88_{\pm 0.04}$	$4.09_{\pm 0.08}$	$5.66_{\pm 0.06}$	$3.04_{\pm 0.04}$	$0.04_{\pm 0.01}$
$\sqrt[10]{\sigma_i}$	$6.48_{\pm 0.08}$	$1.81_{\pm 0.02}$	$2.23_{\pm 0.01}$	$3.20_{\pm 0.02}$	$4.78_{\pm 0.06}$	$2.41_{\pm 0.01}$	$0.01_{\pm 0.01}$
$\ln \sigma_i$	$6.46_{\pm 0.02}$	$1.86_{\pm 0.02}$	$2.30_{\pm 0.02}$	$3.28_{\pm 0.05}$	$4.84_{\pm 0.08}$	$2.47_{\pm 0.02}$	$0.02_{\pm 0.00}$
$\mathbb{U}(\sigma_i)$	$5.62_{\pm 0.17}$	$1.56_{\pm 0.04}$	$1.81_{\pm 0.19}$	$2.67_{\pm 0.05}$	$3.97_{\pm 0.07}$	$2.04_{\pm 0.04}$	$0.01_{\pm 0.00}$
Surrogate Z score loss, characterised by the $\mathbf{m_i}$ error							
Hinge	$5.58_{\pm 0.31}$	$2.54_{\pm 0.16}$	$3.66_{\pm 0.14}$	$5.49_{\pm 0.23}$	$0.00_{\pm 0.00}$	$2.78_{\pm 0.11}$	$0.15_{\pm 0.05}$
Sigmoid	$3.54_{\pm 0.03}$	$3.21_{\pm 0.04}$	$3.26_{\pm 0.04}$	$3.34_{\pm 0.05}$	$3.43_{\pm 0.03}$	$3.21_{\pm 0.04}$	$0.76_{\pm 0.04}$
CE	$5.91_{\pm 0.09}$	$1.40_{\pm 0.07}$	$3.62_{\pm 0.10}$	$5.80_{\pm 0.13}$	$0.94_{\pm 1.88}$	$2.90_{\pm 0.16}$	$0.18_{\pm 0.03}$
Focal	$5.88_{\pm 0.25}$	$0.71_{\pm 0.03}$	$3.54_{\pm 0.08}$	$4.53_{\pm 0.45}$	$0.00_{\pm 0.00}$	$2.12_{\pm 0.05}$	$0.13_{\pm 0.01}$

Table 6.2: Performance of the proposed losses when the whole dataset is used (TC1) to train the classifier using the metrics described in Sec.§ 6.4.3. The highest score for each metric is highlighted for process-aware losses and theoretical ones.

6.5 Results

We set $\epsilon = \mathcal{L} = 1$, as larger or smaller values lead to unstable training. We suspect that the results will improve if we take $epsilon$ to be a small value and $\mathcal{L} = 3\text{ab}^{-1}$.

Tables[6.2-6.4] present the performances of the proposed losses with the metrics defined in section § 6.4.3. We take the BCE loss as the baseline. We report the results only for the interesting cases.

We see that the BCE loss performs worse than both the process-aware weighted cross entropy and the Z score losses across the metrics for all the test cases considered. Between the proposed losses, we see that the empirically weighted losses generally perform better than the surrogate loss. Among the TCs, TC2 and TC3 metrics for the proposed losses show only a slight variation from each other. We see that BCE performs significantly better for TC3 than for TC2, as expected, because TC3 has roughly the same number of samples for all classes. We see that all the *process-aware* losses give a flatter Z score for a higher range of classifier thresholds.

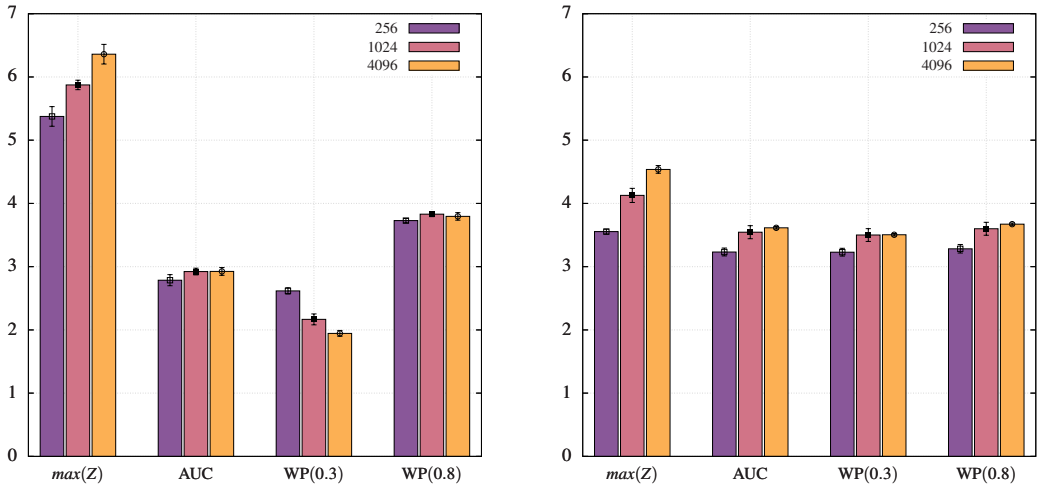
Furthermore, amongst the weighting based on the number of samples of the process within the dataset, we see that $1/\mathcal{N}_i$ consistently performs excellently. However, none of these methods has a flat curve and peaks at a point, like the BCE loss.

Loss	$\max(Z)$	WP1(0.3)	WP2(0.5)	WP3(0.8)	WP4(0.95)	AUC	$R_{10\%}$
BCE	$2.97_{\pm 0.02}$	$0.94_{\pm 0.01}$	$1.10_{\pm 0.02}$	$1.45_{\pm 0.03}$	$2.05_{\pm 0.03}$	$1.16_{\pm 0.02}$	$0.01_{\pm 0.00}$
Weighted CE loss based on dataset size (ω_1)							
$1/\mathcal{N}_i$	$4.33_{\pm 0.07}$	$1.44_{\pm 0.02}$	$1.74_{\pm 0.02}$	$2.40_{\pm 0.03}$	$3.35_{\pm 0.05}$	$1.84_{\pm 0.03}$	$0.02_{\pm 0.01}$
$1/\sqrt{\mathcal{N}_i}$	$3.65_{\pm 0.12}$	$1.15_{\pm 0.02}$	$1.37_{\pm 0.02}$	$1.86_{\pm 0.04}$	$2.62_{\pm 0.06}$	$1.45_{\pm 0.02}$	$0.01_{\pm 0.00}$
$1/\ln(\mathcal{N}_i)$	$3.06_{\pm 0.04}$	$0.97_{\pm 0.00}$	$1.13_{\pm 0.01}$	$1.50_{\pm 0.02}$	$2.11_{\pm 0.03}$	$1.20_{\pm 0.01}$	$0.01_{\pm 0.00}$
Process-aware CE losses, characterised by $g(\omega_2/\mathcal{N}_i)$							
σ_i	$3.94_{\pm 0.28}$	$3.78_{\pm 0.27}$	$3.75_{\pm 0.29}$	$3.67_{\pm 0.44}$	$2.65_{\pm 1.83}$	$3.57_{\pm 0.34}$	$0.87_{\pm 0.01}$
$\sqrt[2]{\sigma_i}$	$4.60_{\pm 0.01}$	$2.91_{\pm 0.05}$	$3.43_{\pm 0.07}$	$4.16_{\pm 0.08}$	$4.53_{\pm 0.12}$	$3.28_{\pm 0.05}$	$0.18_{\pm 0.05}$
$\sqrt[4]{\sigma_i}$	$4.69_{\pm 0.23}$	$2.06_{\pm 0.04}$	$2.49_{\pm 0.03}$	$3.34_{\pm 0.05}$	$4.27_{\pm 0.14}$	$2.54_{\pm 0.02}$	$0.06_{\pm 0.01}$
$\sqrt[10]{\sigma_i}$	$4.65_{\pm 0.19}$	$1.69_{\pm 0.10}$	$2.05_{\pm 0.13}$	$2.83_{\pm 0.24}$	$3.99_{\pm 0.48}$	$2.17_{\pm 0.17}$	$0.02_{\pm 0.01}$
$\ln \sigma_i$	$4.58_{\pm 0.17}$	$1.70_{\pm 0.02}$	$2.06_{\pm 0.02}$	$2.80_{\pm 0.03}$	$3.80_{\pm 0.08}$	$2.14_{\pm 0.02}$	$0.03_{\pm 0.01}$
$\mathbb{U}(\sigma_i)$	$4.27_{\pm 0.15}$	$1.43_{\pm 0.02}$	$1.72_{\pm 0.02}$	$2.34_{\pm 0.01}$	$3.26_{\pm 0.05}$	$1.81_{\pm 0.01}$	$0.02_{\pm 0.01}$
Surrogate Z score loss, characterised by the $\mathbf{m_i}$ error							
Hinge	$4.16_{\pm 0.26}$	$1.81_{\pm 0.07}$	$2.59_{\pm 0.02}$	$3.74_{\pm 0.15}$	$3.49_{\pm 0.70}$	$2.30_{\pm 0.04}$	$0.15_{\pm 0.03}$
Sigmoid	$2.63_{\pm 0.04}$	$2.31_{\pm 0.04}$	$2.36_{\pm 0.04}$	$2.44_{\pm 0.03}$	$2.53_{\pm 0.03}$	$2.33_{\pm 0.04}$	$0.43_{\pm 0.02}$
CE	$4.17_{\pm 0.13}$	$1.78_{\pm 0.06}$	$2.49_{\pm 0.03}$	$3.44_{\pm 0.21}$	$4.02_{\pm 0.24}$	$2.35_{\pm 0.04}$	$0.11_{\pm 0.04}$
Focal	$4.18_{\pm 0.11}$	$1.10_{\pm 0.09}$	$2.50_{\pm 0.02}$	$4.00_{\pm 0.20}$	$2.90_{\pm 0.74}$	$2.17_{\pm 0.04}$	$0.18_{\pm 0.01}$

Table 6.3: Performance of the proposed losses while using a reduced number of background events (TC2) using the metrics described in Sec.§ 6.4.3. The highest score for each metric is highlighted for process-aware losses and theoretical ones.

Loss	$\max(Z)$	WP1(0.3)	WP2(0.5)	WP3(0.8)	WP4(0.95)	AUC	$R_{10\%}$
BCE	$4.41_{\pm 0.09}$	$1.43_{\pm 0.01}$	$1.72_{\pm 0.01}$	$2.33_{\pm 0.02}$	$3.28_{\pm 0.03}$	$1.81_{\pm 0.02}$	$0.02_{\pm 0.01}$
Weighted CE loss based on dataset size (ω_1)							
$1/\mathcal{N}_i$	$\underline{4.49_{\pm 0.08}}$	$1.43_{\pm 0.01}$	$1.71_{\pm 0.01}$	$2.34_{\pm 0.02}$	$\underline{3.30_{\pm 0.03}}$	$1.81_{\pm 0.02}$	$0.01_{\pm 0.00}$
$1/\sqrt{\mathcal{N}_i}$	$4.44_{\pm 0.03}$	$1.43_{\pm 0.01}$	$1.71_{\pm 0.01}$	$2.33_{\pm 0.01}$	$3.28_{\pm 0.01}$	$1.81_{\pm 0.01}$	$0.01_{\pm 0.00}$
$1/\ln(\mathcal{N}_i)$	$4.45_{\pm 0.04}$	$\underline{1.44_{\pm 0.01}}$	$\underline{1.72_{\pm 0.01}}$	$\underline{2.34_{\pm 0.01}}$	$3.29_{\pm 0.01}$	$1.81_{\pm 0.01}$	$\underline{0.02_{\pm 0.00}}$
Process-aware CE losses, characterised by $\omega_2/\mathcal{N}_i$							
σ_i	$3.78_{\pm 0.22}$	$\underline{3.67_{\pm 0.27}}$	$\underline{3.60_{\pm 0.27}}$	$3.47_{\pm 0.41}$	$3.10_{\pm 0.59}$	$\underline{3.46_{\pm 0.27}}$	$\underline{0.75_{\pm 0.09}}$
$\sqrt[2]{\sigma_i}$	$4.35_{\pm 0.08}$	$2.95_{\pm 0.03}$	$3.39_{\pm 0.06}$	$\underline{4.06_{\pm 0.06}}$	$\underline{4.32_{\pm 0.06}}$	$3.25_{\pm 0.04}$	$0.26_{\pm 0.02}$
$\sqrt[4]{\sigma_i}$	$\underline{4.89_{\pm 0.14}}$	$2.01_{\pm 0.02}$	$2.44_{\pm 0.03}$	$3.30_{\pm 0.03}$	$4.29_{\pm 0.09}$	$2.50_{\pm 0.03}$	$0.04_{\pm 0.01}$
$\sqrt[10]{\sigma_i}$	$4.79_{\pm 0.14}$	$1.64_{\pm 0.02}$	$1.98_{\pm 0.03}$	$2.72_{\pm 0.08}$	$3.82_{\pm 0.14}$	$2.08_{\pm 0.04}$	$0.02_{\pm 0.00}$
$\ln(\sigma_i)$	$4.81_{\pm 0.19}$	$1.67_{\pm 0.03}$	$2.02_{\pm 0.05}$	$2.77_{\pm 0.11}$	$3.84_{\pm 0.16}$	$2.12_{\pm 0.06}$	$0.02_{\pm 0.00}$
$\mathbb{U}(\sigma_i)$	$4.44_{\pm 0.14}$	$1.39_{\pm 0.02}$	$1.67_{\pm 0.02}$	$2.27_{\pm 0.03}$	$3.23_{\pm 0.07}$	$1.76_{\pm 0.02}$	$0.01_{\pm 0.01}$
Surrogate Z score loss, characterised by the $\mathbf{m_i}$ error							
Hinge	$\underline{4.54_{\pm 0.34}}$	$1.84_{\pm 0.07}$	$\underline{2.89_{\pm 0.16}}$	$\underline{4.37_{\pm 0.37}}$	$0.93_{\pm 1.87}$	$2.41_{\pm 0.10}$	$0.16_{\pm 0.04}$
Sigm.	$3.45_{\pm 0.58}$	$\underline{2.70_{\pm 0.25}}$	$2.82_{\pm 0.30}$	$3.03_{\pm 0.40}$	$3.24_{\pm 0.50}$	$\underline{2.79_{\pm 0.30}}$	$\underline{0.21_{\pm 0.19}}$
CE	$4.33_{\pm 0.22}$	$1.69_{\pm 0.12}$	$2.80_{\pm 0.23}$	$4.06_{\pm 0.47}$	$\underline{3.43_{\pm 0.81}}$	$2.45_{\pm 0.13}$	$0.18_{\pm 0.06}$
Focal	$4.44_{\pm 0.19}$	$0.94_{\pm 0.19}$	$2.79_{\pm 0.23}$	$4.08_{\pm 0.34}$	$0.86_{\pm 1.73}$	$2.02_{\pm 0.14}$	$0.16_{\pm 0.01}$

Table 6.4: Performance of the proposed losses when the classifier is trained with only 10% of the total data (TC3) using the metrics described in Sec.§ 6.4.3. The highest score for each metric is highlighted for process-aware losses and theoretical ones.



(a) Hinge Error

(b) Sigmoid Error

(c) Cross Entropy Error

(d) Focal Error

Figure 6.3: The effect of batch size scaling to some performance metrics $\max(Z)$, AUC, WP0.3, WP0.8 for the surrogate loss $\bar{\Delta}_Z$. Batch sizes 256, 1024 and 4096 have been considered for the study.

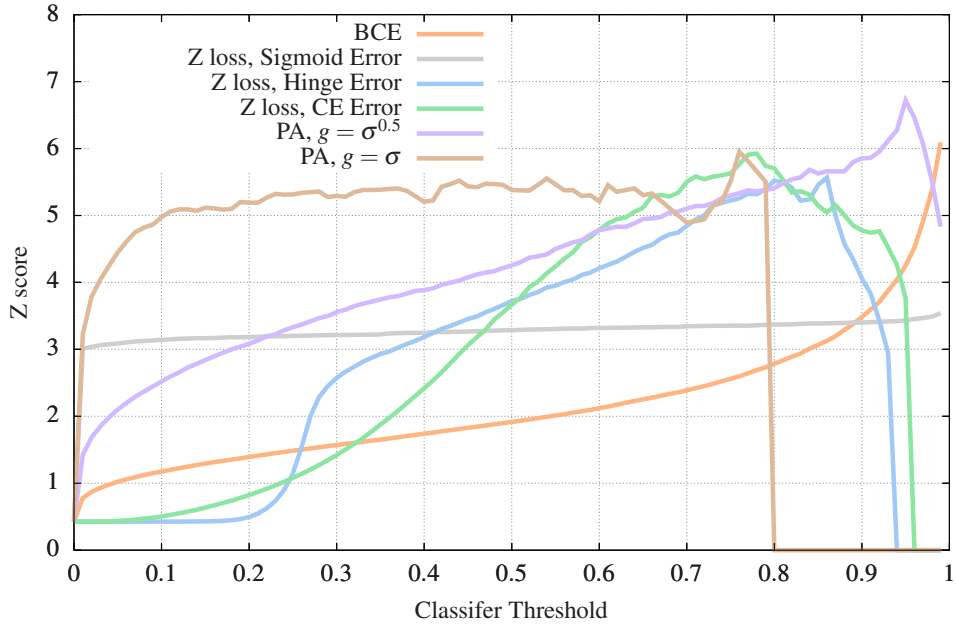


Figure 6.4: The Z vs Classifier Threshold curves for various loss functions. PA stands for the process weighted loss functions

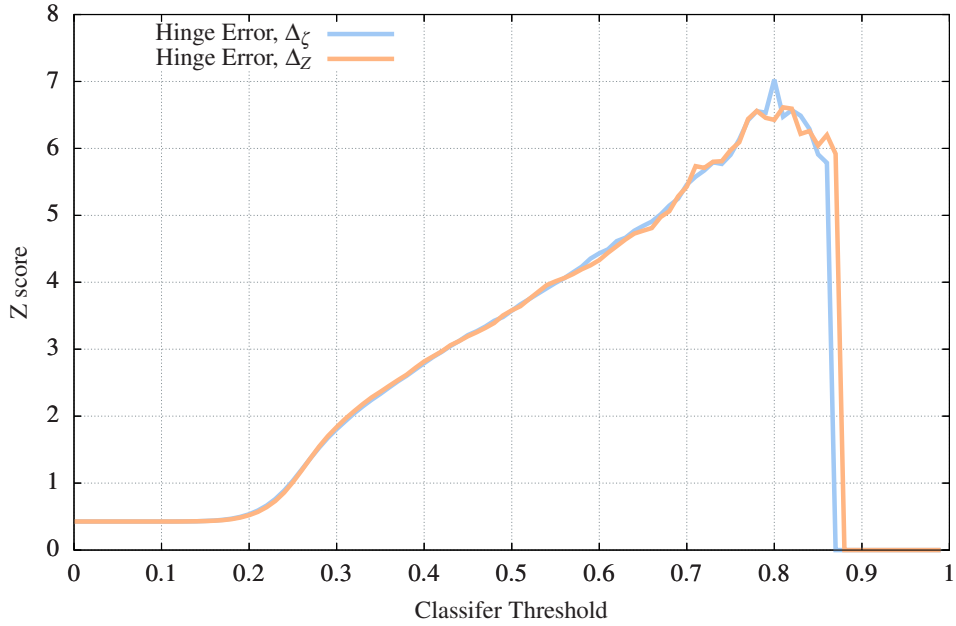


Figure 6.5: The Z vs Classifier Threshold curves for comparing surrogate loss of Δ_Z to that of Δ_ζ

Two losses are particularly noteworthy: (1) weighted cross entropy with $\omega_i = \sigma_i/\mathcal{N}_i$, and (2) surrogate loss function with $\mathbf{m}$ given by the Sigmoid scheme. In both losses, we note that the maximal Z score is achieved very early, and the curve remains very flat as we increase the threshold. Figure 6.4 shows the Z vs NN threshold response curve for both losses.

We also study the effect of various batch sizes on the performance of the surrogate loss function (Fig. 6.3). Overall, we see a slight improvement as the batch size increases, but we note that this improvement is marginal for all practical purposes.

6.6 Related Works

Weighted CE losses have been used to mitigate the effects of dataset imbalances while training a classifier [89, 90]. Biasing them with priors is known to improve model performances—for example, in the case of language models, biasing the loss with character frequencies [91] performs better than a naive Bayes classifier. Earlier, the Lovász extension has been used to derive loss functions in various other fields as well. Our proof of Theorem 6.3.3 is similar to Ref. [86], which defines a surrogate loss called the Lovász Hinge with reduced complexity of gradient computation. In the domain of image segmentation, the authors of Ref. [85] have defined a surrogate loss over the mean Jaccard index.

In particle physics, recent works focus on developing losses for likelihood-free inference by training a neural network to estimate the likelihood ratios directly. Refs. [92, 93] exploit latent information available in Monte Carlo simulators and define losses that utilize this additional information to estimate the likelihood ratio directly. The most closely related work to ours is that of Ref. [94], where the authors devise a surrogate loss to maximize a term proportional to the extended likelihood function. Our work differs from theirs in that we take the asymptotic formulae for likelihood-based tests, derive a continuous relaxation of the sensitivity, and maximize it directly.

Chapter 7

Generative Modelling of Jets at the LHC

Deep learning models rely on large amounts of data to make accurate predictions. We use Monte Carlo event generators to construct the datasets for these models, which is computationally intensive and time-consuming. In this chapter, we investigate the potential of generative models to produce a large number of jet samples. Such models can facilitate searches for new physics by resource-limited groups and accelerate large-scale prototyping. Moreover, such a method can provide a useful prior for anomaly detection or other self-supervised learning approaches.

7.1 Introduction and Motivation

As we pursue rarer signals at the LHC, we require enormous amounts of data to infer information about the problem (and especially to apply deep learning techniques to it). This poses a barrier for resource-limited groups who want to participate in such endeavours. Large, pre-processed datasets might seem like an obvious solution here; however, they have certain caveats. They are cumbersome to handle, difficult to distribute, and do not allow for control by the end user. The natural goal, then, is to develop a faster alternative to current simulation-based frameworks for event generation. Generative models offer a promising solution as they are able to encode information from large datasets. These generative models could easily be distributed between teams, and events may be generated by the end-user with specific generation control that could be incorporated into the model's design. Furthermore, deep learning-based generative models are much faster and can be parallelised over GPUs, unlike current classical Monte Carlo-based generation methods.

However, replacing event generators with deep generative models is a challenging task; it would involve modelling multiple stages of a hard scattering event (§ 3.3). In this chapter, we focus on the simpler task of generating the final state jets of an event. This problem is similar to the event generation problem, and the lessons learned from solving this simpler problem can be

applied to the more complex one. In this chapter, we, therefore, propose a Generative Adversarial Network (GAN) (see section § 4.3.2) to generate the jets.

7.2 Mathematical Setup for Jet Generation

We generate jets as particle clouds (see § 4.6.1). A challenge that we face that is absent in 3D shape (point cloud) generation is the variable sizes of the jets. Therefore, we model the cardinality and the attributes of the constituents as a product of two distinct probabilities. We model the generative process for jets as follows — Let Y be a jet of size n (i.e., it has n constituents) and $y_1, y_2, \ldots, y_n$ be the constituents of the jet. Then we have:

$$p(Y|z) = p(n|z) \cdot p(y_1, y_2, \ldots, y_n|z, x_1, x_2, \ldots, x_n) \tag{7.1}$$

where z is a global latent noise parameter which controls both the cardinality of the jet and the properties of constituents of the jet. Additionally, for the second part of the generation, we have point-wise noise parameters $x_1, x_2, \ldots, x_n$, which are used to distinctly model the particles in the jet.

This mathematical framework translates into a two-step process:

1. Sample the global latent noise $z \sim \mathcal{P}$ from a chosen probability distribution $\mathcal{P}$, and pass it as input into the `Sizes Generator` which would output the cardinality of the jet (n).

2. Samples n points $x_1, x_2, \ldots, x_n \sim P'$ from another probability distribution P' to construct the input set. Append the global latent noise z to each of the points and pass it as input to the `Particles Generator` to transform the input set to the final particles within the jet.

We note that the cardinality of a jet is not an IRC-safe quantity [95], and therefore this generation is not IRC-safe. In order for one to construct an IRC-safe generator, one should consider IRC-safe quantities such as p_T, M_J, but such an exploration is out of the scope of the current work.

7.3 Methodology

The complete schema of the method is shown in Fig 7.1. The generation is performed in a pipeline with two stages. Each stage has a separate generator, namely, the `Sizes Generator` and the `Particle Generator`. Accordingly, we have two discriminators for each stage.

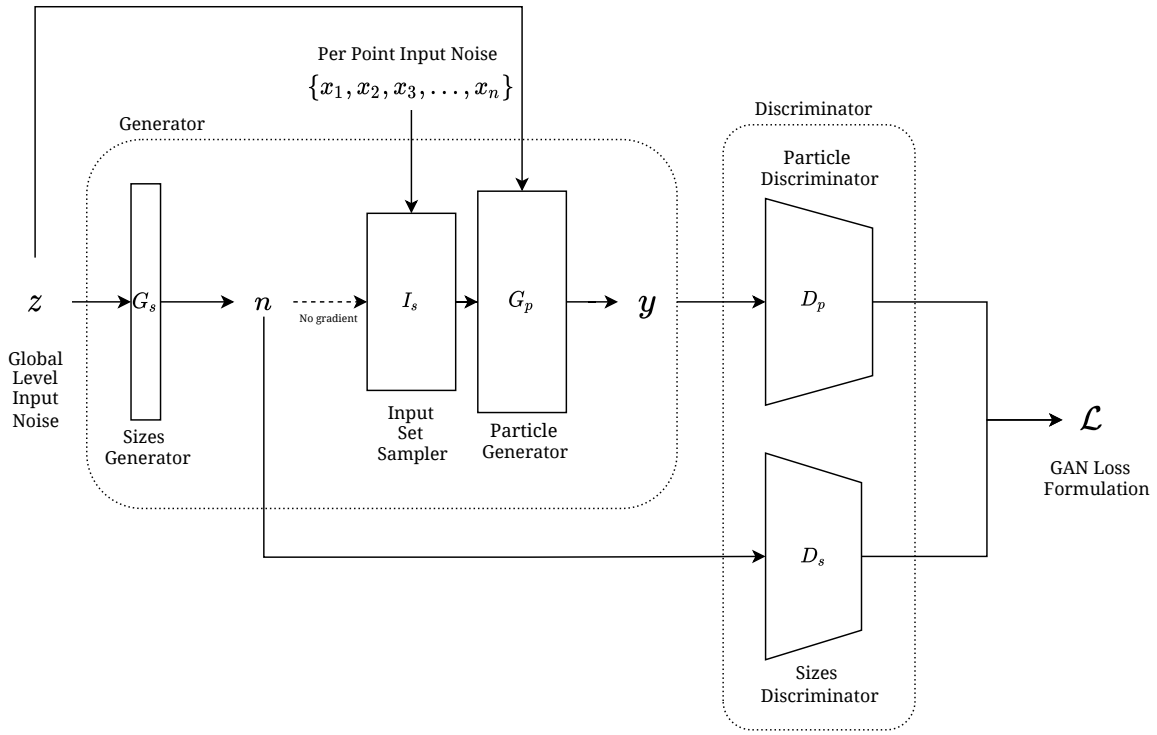


Figure 7.1: The schematic of the GAN setup. z is input to the Sizes Generator, which outputs n. This is passed to an input set sampler that samples n points. These sampled points and z are passed into the Particle Generator, which outputs y. There are 2 discriminators, one - for each of the two generators.

7.3.1 Generating the discrete categorical variable n

Generating discrete variables is a difficult task for neural networks since such variables are not differentiable and do not provide gradients to train the network weights. Discrete data can be generated by sampling from a categorical distribution, which may be constructed using the `softmax` operation on the outputs of the neural network. However, this sampling process is non-differentiable.

$$y_i = \texttt{onehot}(\texttt{argmax}(\pi_1, \pi_2, \ldots, \pi_N)) \tag{7.2}$$

Refs. [96, 97] proposed to reparameterize the problem using a Gumbel Distribution to enable sampling from the underlying probabilities. Furthermore, they sidestepped the problem of non-differentiability by replacing the `argmax` operation with a `softmax` operation with a temperature τ. For a noise sample g_i sampled from Gumbel$(0, 1)$ distribution y_i is given by:

$$y_i = \frac{e^{\frac{log(\pi_i)+g_i}{\tau}}}{\sum_{j=1}^{N} e^{\frac{log(\pi_j)+g_j}{\tau}}} \tag{7.3}$$

As $\tau \to 0$, this approximation becomes closer to the `onehot` representation of the discrete variable, and as $\tau \to \infty$, this approximation becomes closer to a uniform distribution.

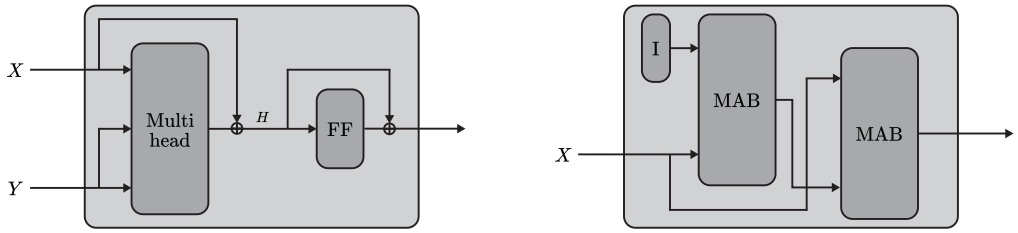
We use this setup to generate the categorical variable n. We have N as the maximum cardinality in the dataset (here 100). The probabilities π_i are the output of `Sizes Generator`, an MLP network, $\pi = f_1(z)$ where z is the global level noise. We additionally use the Straight-Through trick to obtain a onehot representation of y_i.

7.3.2 Generating the Jet

Particle Clouds are inherently sets, and thus, functions on them must respect certain constraints and symmetries to prevent adding in artificial information (§ 4.6.1). We re-iterate these constraints:

- The network should be able to work for variable-sized inputs (since sets, in general, can have variable cardinality, the network needs to be able to adapt to that).
- The network should be permutation invariant (or equivariant in some cases). This symmetry is fundamental as it reduces the search space for the network. A set can have $n!$ arrangements, and thus a network that is invariant to this symmetry would learn faster and with fewer data points.

In order to generate permutation-invariant jets, we reimplement the SetTransformer [98] architecture with a minor modification that allows it to work for variable-sized inputs without masks. The SetTransformer architecture is based on Transformers [99], originally developed for language modelling. Transformers are essentially graph neural networks (§ 4.6.2) [100]



(a) MAB Architecture (b) ISAB Architecture

Figure 7.2: Architectures of the Blocks from Set Transformer. *Figures by original authors*

that assume a complete graph between the nodes and learn functions that can focus (or attend) on interactions between specific subsets of nodes. We direct the reader to [101] for a review on transformers. SetTransformer proposes an attention-based module that models interactions between the elements of the input set. Using schemes inspired by the inducing points method in Gaussian process literature, the authors reduce the self-attention complexity from quadratic to linear in the number of elements in the set. This reduced complexity attention module is dubbed ISAB, shown in Fig 7.2.

$$ISAB_m(X) = MAB(X, H) \in \mathbb{R}^{n \times d} \tag{7.4}$$

$$\textbf{where}, H = MAB(I, X) \in \mathbb{R}^{m \times d} \tag{7.5}$$

Here I (dubbed inducing points) is a learnable parameter of size $m \times d$, and MAB is the `MultiHeadAttention` block (from the Transformer architecture). The complexity of this layer is $\mathcal{O}(nm) \approx \mathcal{O}(n)$ for small constant values of m.

The `Particle Generator` consists of 5 *ISAB* blocks, each with 4 separate attention heads. The first two blocks have an embedding dimension of 128, and the last three blocks have an embedding dimension of 64. `LeakyReLU` [102] is used as the activation function, and `LayerNorm` [103] is used to stabilize the training, following the transformer architecture. This creates a *Set-to-Set* architecture, and the input sampled point cloud is converted to the final particle cloud.

7.3.3 Discriminator Architectures

In this work, we have two discriminators which are trained independently. The first discriminator provides guidance to the `Sizes Generator`, and the second provides guidance to the `Particle Generator`.

Jet Type	# Parameters
Total (Sizes + Particle) Generator	460k
Sizes Discriminator	36.8k
Particle Discriminator	699k

Table 7.1: Model Architecture Details

7.3.3.1 Sizes Discriminator

The role of the `Sizes Discriminator` is to discriminate between the generated onehot samples of n with the true distribution of n. The architecture is a simple MLP composed of a series of Linear layers and `Swish` Activation. The `Sizes Discriminator` has 6 MLP layers with an embedding dimension of 128, followed by 4 layers with an embedding layer 64, and finally, a layer with an embedding dimension 32. `BatchNorm` is also used for stabilizing the training. This task of this discriminator is relatively easy and thus should have sufficient capacity.

7.3.3.2 Particle Discriminator

The task of the `Particle Discriminator` is to discriminate between generated particle clouds (jets) and real particle clouds (jets). This discriminator must also respect the constraints specified in section§ 4.6.1. The `Particle Discriminator` also comprises **ISAB** layers in series followed by a global pooling operation and an MLP. The `Particle Discriminator` consists of 4 `ISAB` blocks in series. Each $ISAB$ block with an embedding dimension of 128 and four separate attention heads. `LeakyReLU` is used as the activation function, and `LayerNorm` is used for stabilizing the training.

The total number of parameters of the models trained are provided in Table 7.1. The discriminator networks are made to be more complex to provide better guidance to the generator networks.

7.4 Dataset

The dataset for training the GAN is based on the Top Tagging Reference Dataset ([104]). This dataset was used as it has sufficiently complex jets, with the cardinalities varying from 7 to 200. We believe that a network that can replicate this complex data would, in principle, replicate a host of other kinds of jets.

The dataset contains 14TeV hadronic Tops as signal and QCD dijets as background. Parton shower and hadronization were performed using Pythia8, and detector simulation was performed using Delphes. The jets were clustered using the anti-k_T algorithm with a ΔR of 0.8 in the p_T

range of $[550, 650]$ GeV. The dataset contains jets with max 200 constituents, but we reduced the maximum size to 100 for our purposes.

We transformed the jets to change their basis from (E, p_x, p_y, p_z) to (E, p_T, η, ϕ). η and ϕ values were normalized with respect to the jet so that they lie between $[-1, 1]$. We worked with $\log E$ and $\log p_T$ as they are easier to train on.

7.5 Training

To train the network, we adopt the LSGAN formulation [34], which uses the least squares loss function instead of the binary cross entropy loss from the original formulation. The LSGAN formulation is defined as follows:

$$\min_D V(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[(D(x)-b)^2\right] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[(D(G(z))-a)^2\right] \tag{7.6}$$

$$\min_G V(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[(D(G(z))-c)^2\right] \tag{7.7}$$

Here we choose $b = c = 1$ and $a = 0$.

Minimizing the LSGAN loss is equivalent to minimizing Pearson's χ^2 divergence between the generated and real dataset. LSGAN encourages the generator to generate samples closer to the decision boundary in the discriminator's space, leading to better quality and diversity of the generated samples. LSGAN also alleviates the issue of vanishing gradient that occurs in the original formulation when the discriminator becomes too powerful.

We also use Spectral Normalization [105] on the weights of the discriminators to stabilize the training process. Spectral Normalization is a technique that constrains the spectral norm (the largest singular value) of the weight matrix to be less than or equal to a certain value. This prevents the discriminators from becoming too dominant over the generator and ensures balanced learning dynamics.

We trained the generator and the two discriminators alternatively, updating the discriminators three times each time for every two generator updates. This unequal frequency allows the discriminator to learn faster and provide more meaningful feedback to the generator.

`Rmsprop` optimizer was used with a learning rate of 0.0002. We trained the models for 600 epochs on 2 GTX1080Tis. The models were implemented in `Pytorch` Deep Learning Framework [106]. `Pytorch Geometric` [107] and `Pytorch Lightning` [108] were used for model creation and training respectively.

7.6 Evaluation

Evaluating generated jets is a difficult task as there is no single metric that is sensitive to the features of the jets. Taking inspiration from the domain of generative modelling in Computer Vision [109] have adapted and tested a few metrics to evaluate the quality of the generation of jets. Among these, they find the following metrics sensitive enough:

1. Frechet Particle Distance **(FPD)** — Frechet Particle Network, inspired by the image-counterpart Frechet Inception Distance [110], measures the difference between the generated images and the true images by finding the W_2 distance between the latent distribution of ParticleNet for the real data and the generated data.

2. Maximum Mean Discrepancy **(MMD)**: Using the Energy Mover's Distance metric [111], designed to compare collider events, we calculate the MMD between the generated and true distributions.

3. Wasserstein Distance on Energy Flow Polynomials $W_{\textbf{EFP}}$ — Energy Flow Polynomials [112] form a linear basis on all the IRC-safe observables that can be constructed from jets. Therefore, one can compare the difference in such IRC-safe observables by comparing the Wasserstein distance on the Energy Flow Polynomials.

7.7 Results

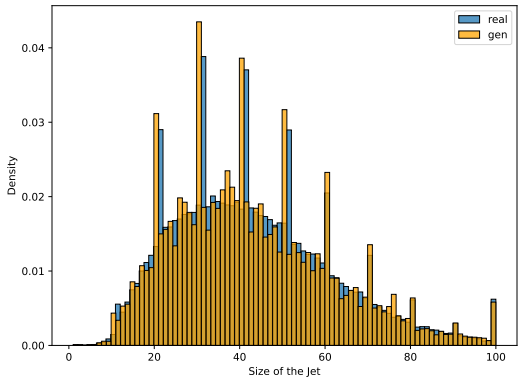
Jet Type	$W_{EFP} \downarrow$	MMD $\downarrow$	FPD Score $\downarrow$
Top Jet	0.0001	0.2300	0.0055
QCD Jet	0.0005	0.3364	0.0060

Table 7.2: Results of the metrics for jets Generated by the Model

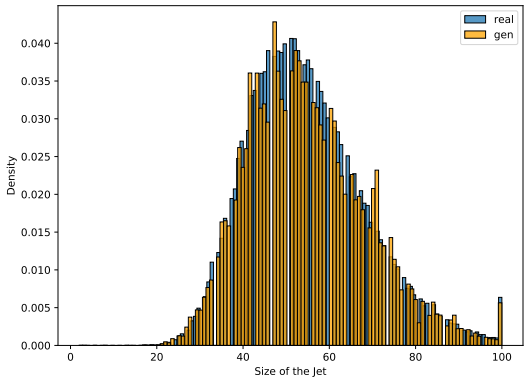
Table 7.2 summarizes our results for Top Jet and QCD Jet generation. The Wasserstein distance between the EFPs of the real and generated data is low, which indicates that the IRC safe observables computed from the generated data are similar to those computed from the real data. Both our FPD score and MMD score are low. However, it shows that there may still be scope for improvement. This we can see from a few cases in Fig. 7.7 to Fig. 7.14.

We also plot the distribution of the 4-vector in Fig 7.4 and Fig 7.5. Visually from the distributions of $\log p_T, \log E_t, \eta$, and ϕ, the generated data appears to follow the same distribution as the real data. We also plot the distribution of the masses of the constituents in Fig 7.6.

Additionally, we plot, in Fig 7.3, the size distributions of the jets in the generated dataset and the true dataset. We can observe that, although there are some discrepancies, the general trend of the learned distribution follows the true distribution.



(a) The distribution of n in Top Jet generation



(b) The distribution of n in QCD Jet generation

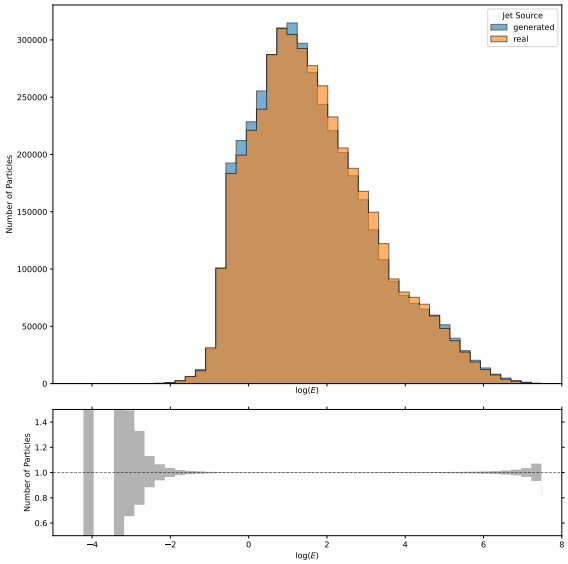
Figure 7.3: The distribution of n learned for the two cases. Plot generated with 100000 samples of real and generated jets. *(real in blue and generated in orange)*

7.7.1 Plots For Individual Jet Sizes

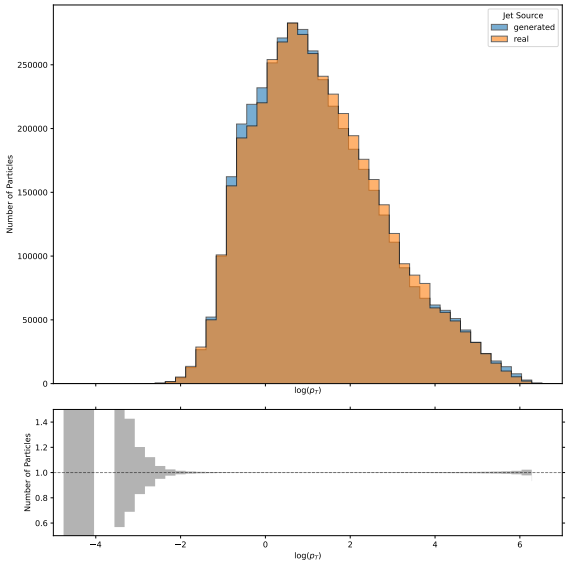
We also plot the distributions of $\log p_T$,$\log E$, η, and ϕ for jets with various sizes in Figs. 7.7 to Fig. 7.14. In most cases, the generated samples are close to the real distribution. The cases where the generated distribution deviates from the actual distribution are when the jet sizes are very small or very large and there is insufficient training data for the model to learn the distribution. Through these plots, we can also see the disparity between the quality of the top jets generated and the QCD jets. In the case of QCD jets, the model has difficulty learning the distribution of jets with > 60 constituents.

7.8 Related Works

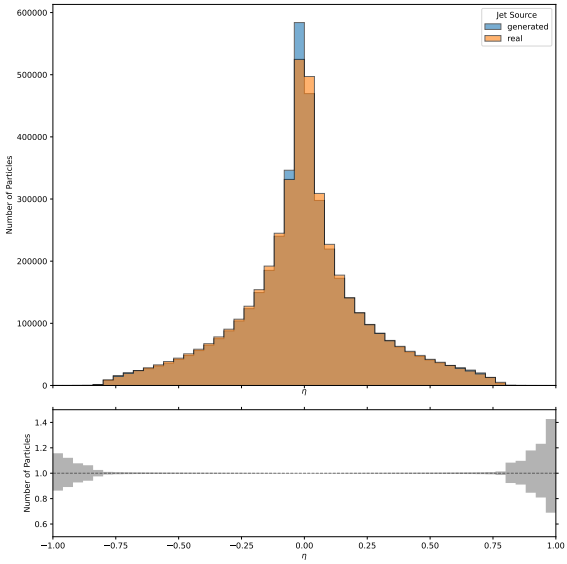
Deep generative models have attracted increasing attention as a tool for generating samples for various stages of the Monte Carlo event generation pipeline Ref. [113] employed conditional GANs with MLPs and trained them on the MMD-GAN objective to generate hard partonic collision samples to be used in later stages of the generation pipeline. They demonstrated that the GAN-generated events satisfied physical constraints such as momentum conservation. Refs. [114, 115] proposed using GANs as an alternative to the detector-level simulations performed by GEANT4, which is the most time-consuming stage of the pipeline and thus can benefit from significant speed-ups. Current phenomenological rely on events simulated using fast approximations for the actual detector simulations. Ref. [116] provided a proof of concept for GANs to generate jets. The works in Refs.[117, 118] develop models for jet generation based on normalizing flows and diffusion models. However, these methods work with smaller jets with little variation in the jet



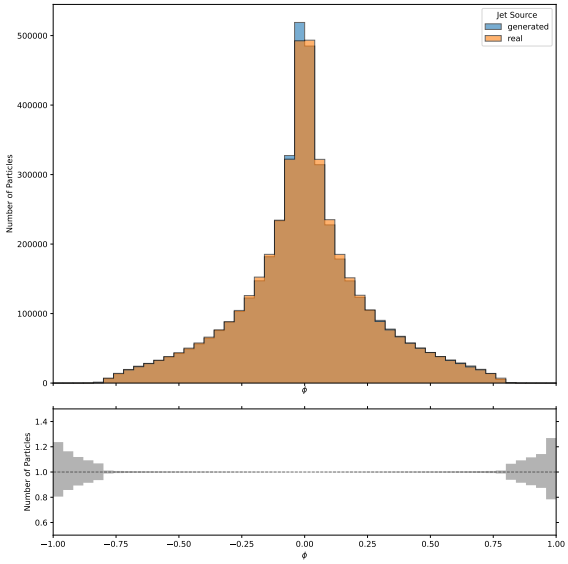
(a) The distribution of $\log E$ in Top Jet generation



(b) The distribution of $\log p_T$ in Top Jet generation

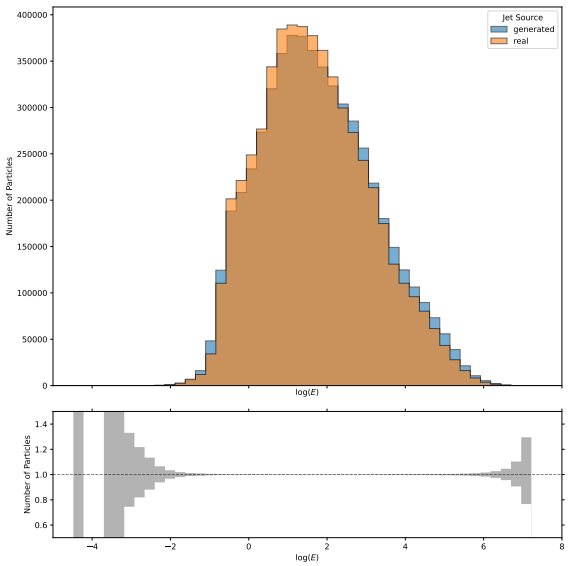


(c) The distribution of η in Top Jet generation

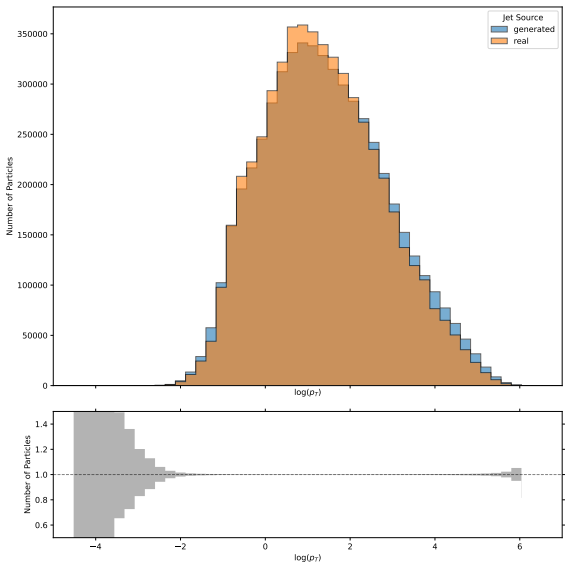


(d) The distribution of ϕ in Top Jet generation

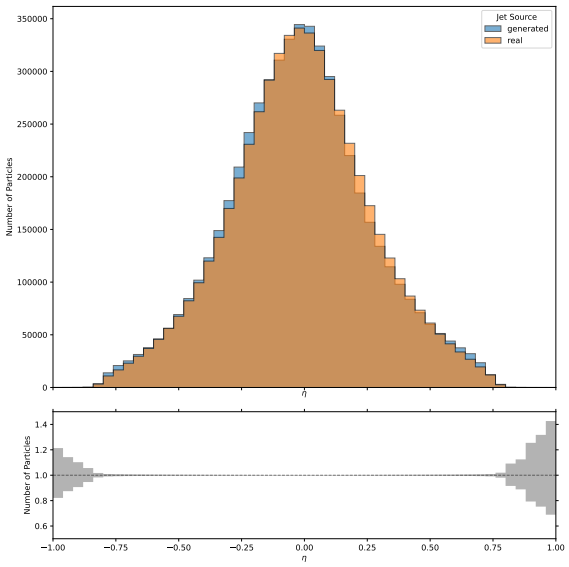
Figure 7.4: The distribution of $\log p_T, \log E, \eta, \phi$ for generated Top jets. Plot generated with 100000 samples of real and generated jets. *(real in orange and generated in blue)*



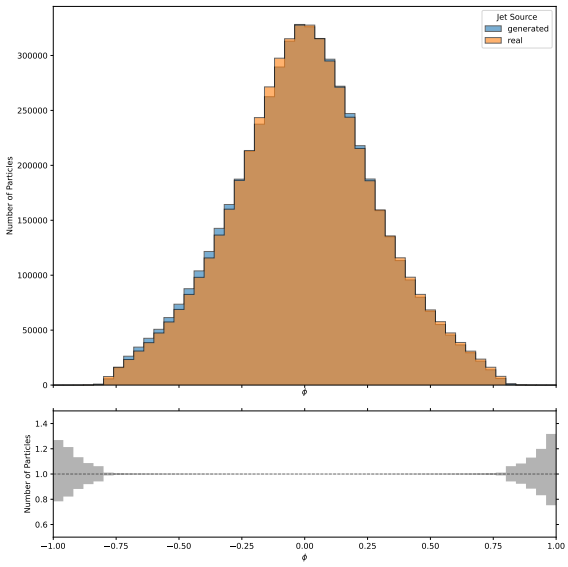
(a) The distribution of $\log E$ in QCD Jet generation



(b) The distribution of $\log p_T$ in QCD Jet generation

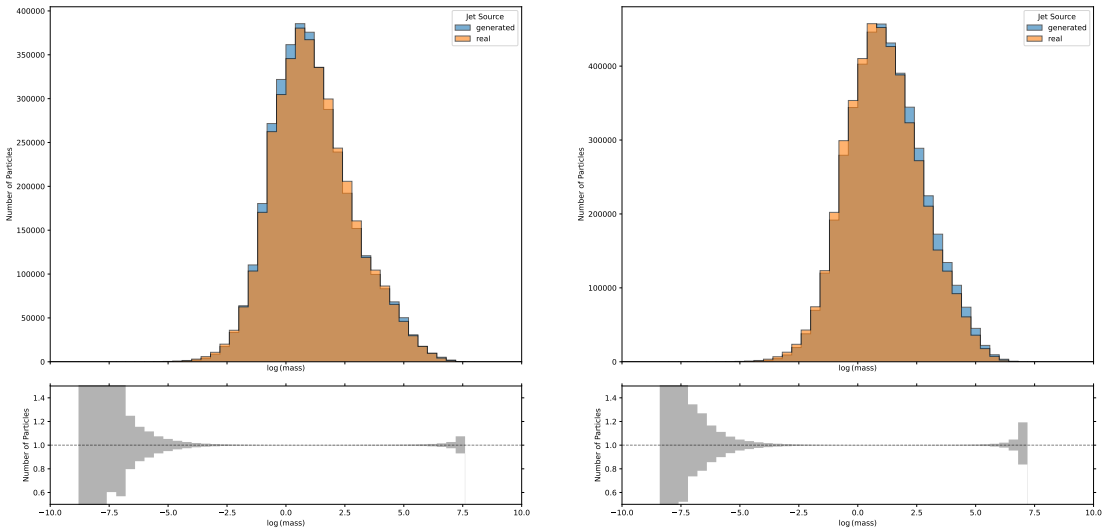


(c) The distribution of η in QCD Jet generation



(d) The distribution of ϕ in QCD Jet generation

Figure 7.5: The distribution of $\log p_T, \log E, \eta, \phi$ for generated QCD jets. Plot generated with 100000 samples of real and generated jets. *(real in orange and generated in blue)*



(a) The distribution of log mass in Top Jet generation

(b) The distribution of log mass in QCD Jet generation

Figure 7.6: The distribution of log mass for generated Top and QCD Jets. Plot generated with 100000 samples of real and generated jets. *(real in orange and generated in blue)*

size. Moreover, they do not incorporate the variability of jet size into the generative modelling but instead introduce extra "masks" that emulate the output. Our work differs from theirs as we not only incorporate an additional generator that can learn the distribution of the jet sizes, but we also use the GAN framework, which leads to faster generation compared to methods like normalizing flows and diffusion models.

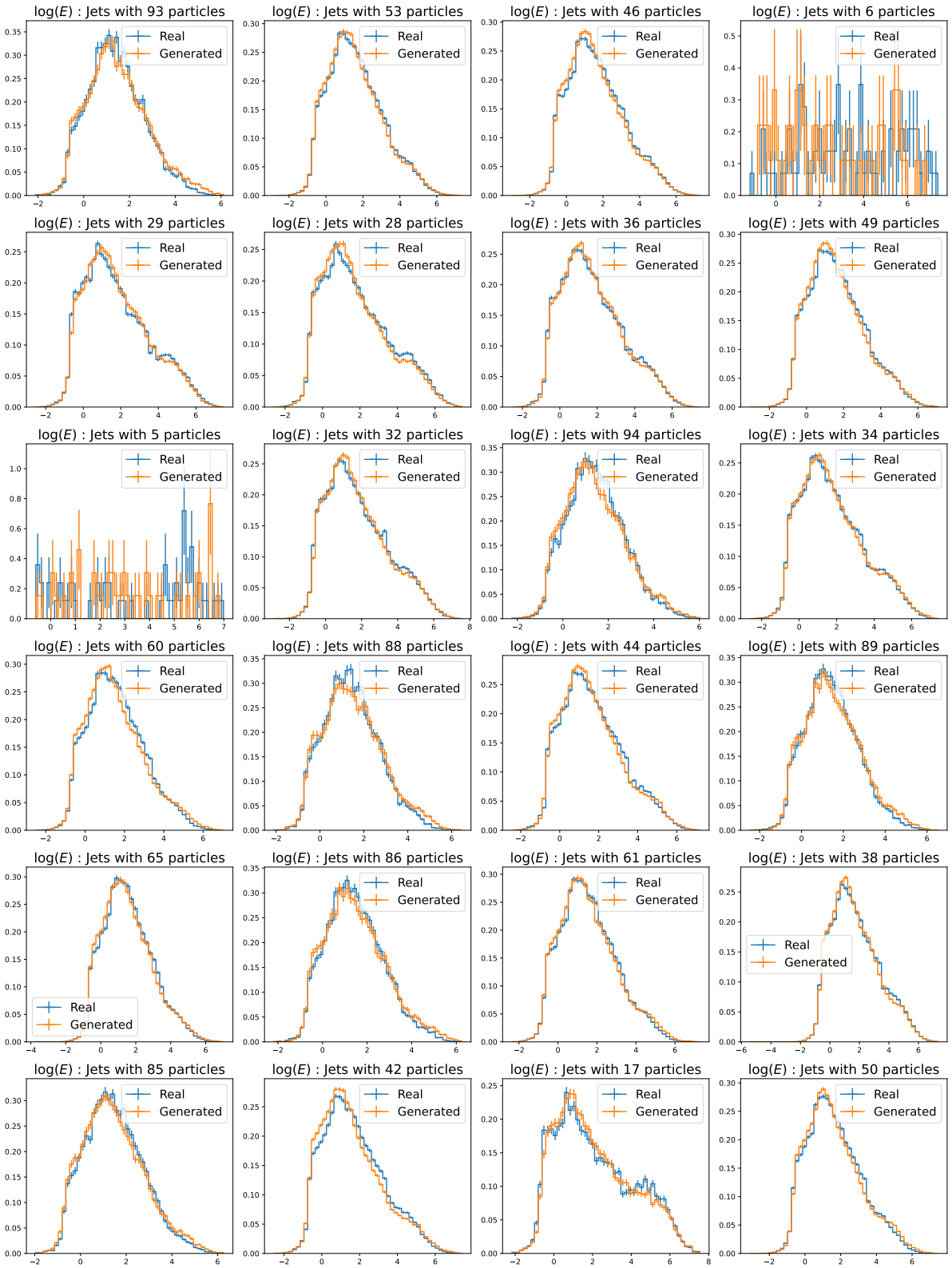


Figure 7.7: $\log E$ distribution for Top jets of various sizes

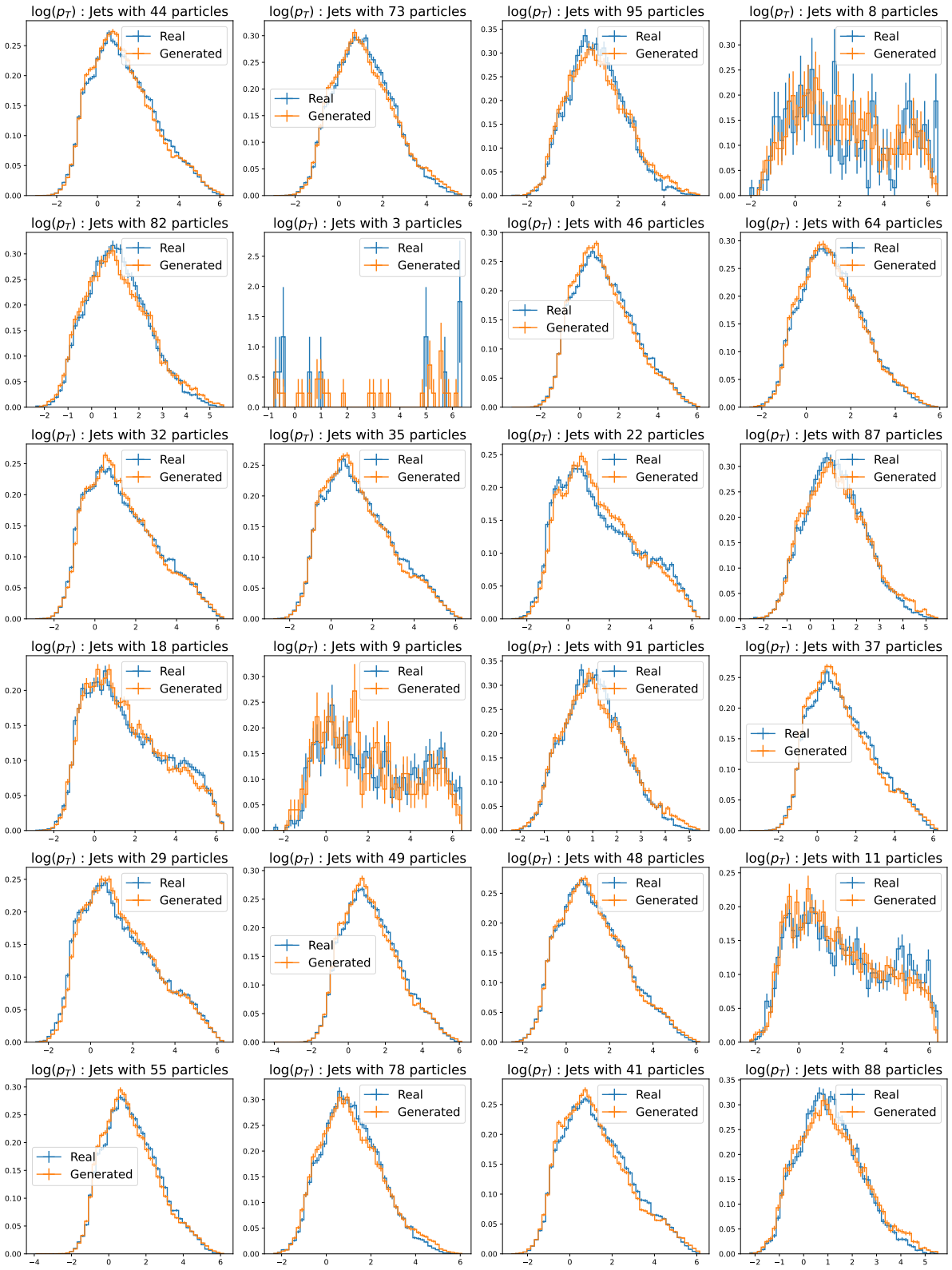


Figure 7.8: $\log p_T$ distribution for Top jets of various sizes

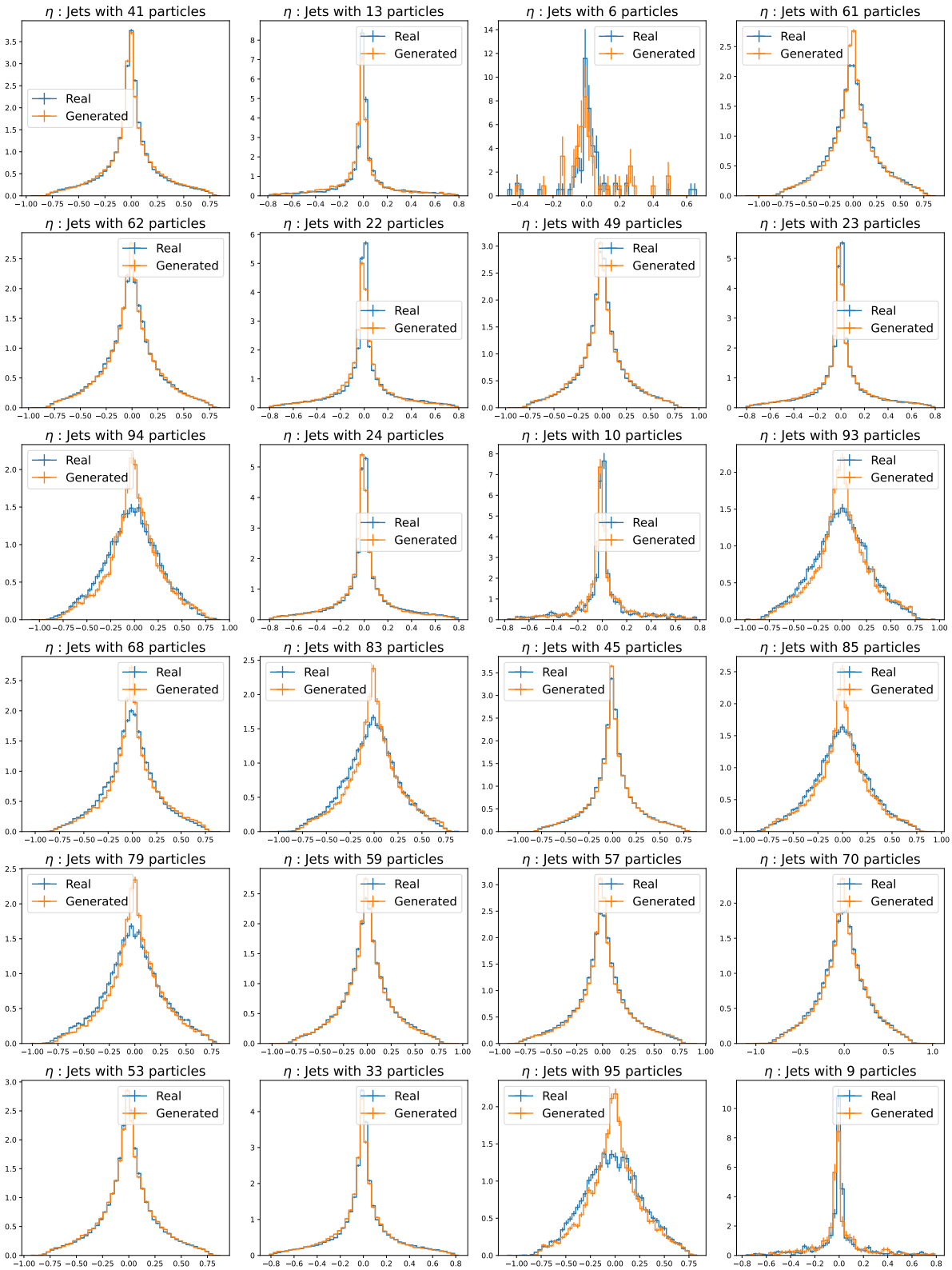


Figure 7.9: η distribution for Top jets of various sizes

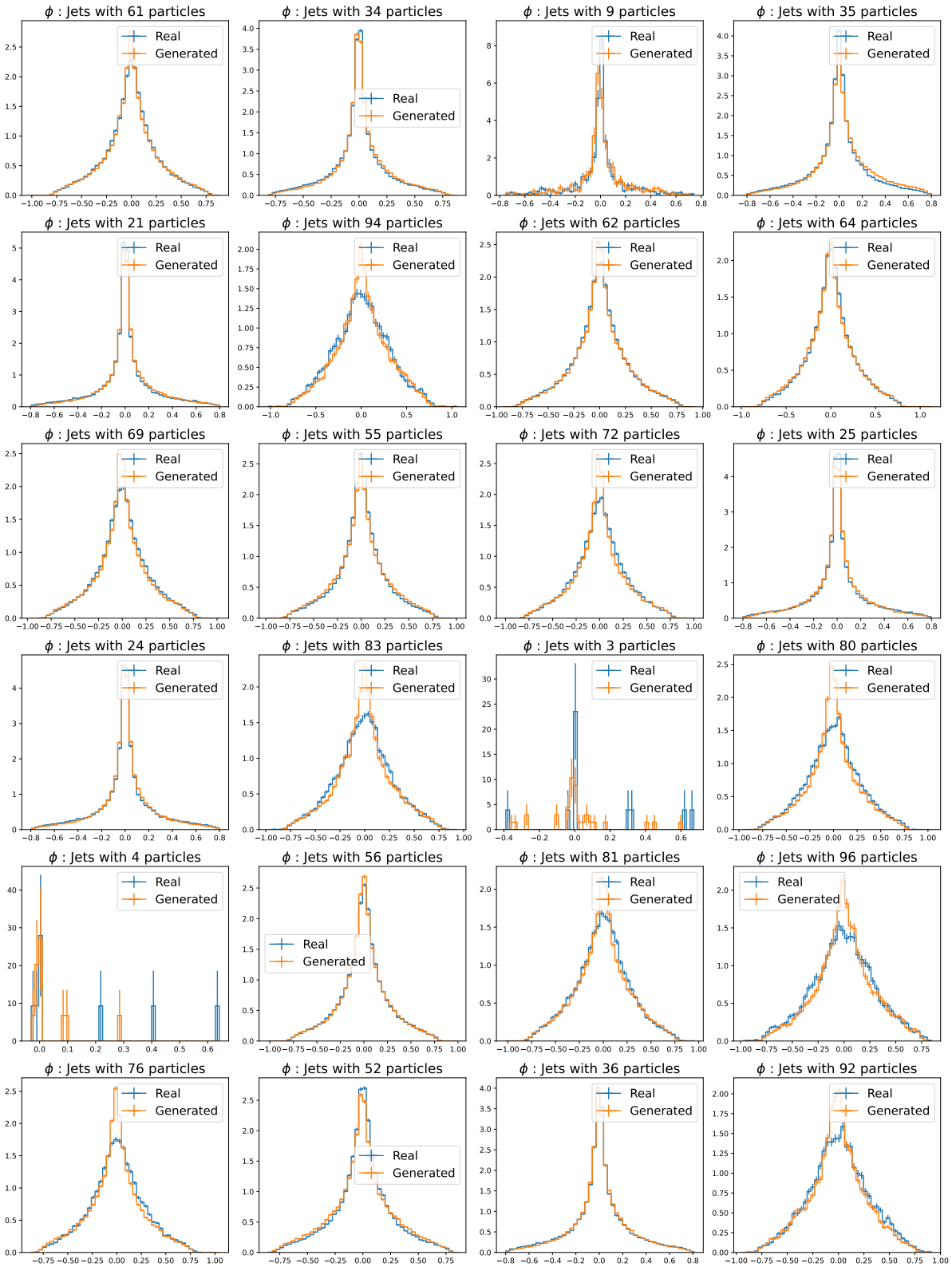


Figure 7.10: ϕ distribution for Top jets of various sizes

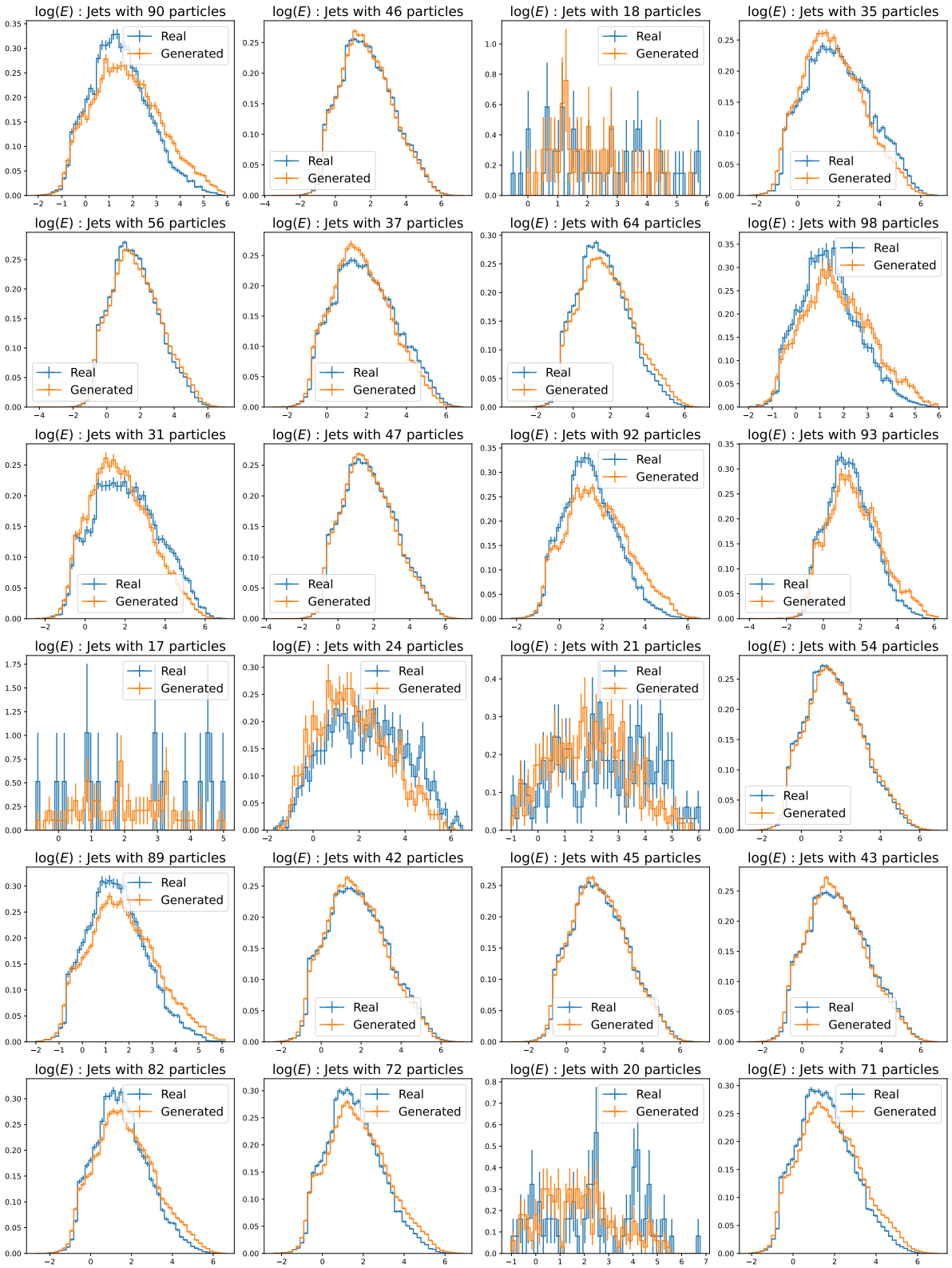


Figure 7.11: $\log E$ distribution for QCD jets of various sizes

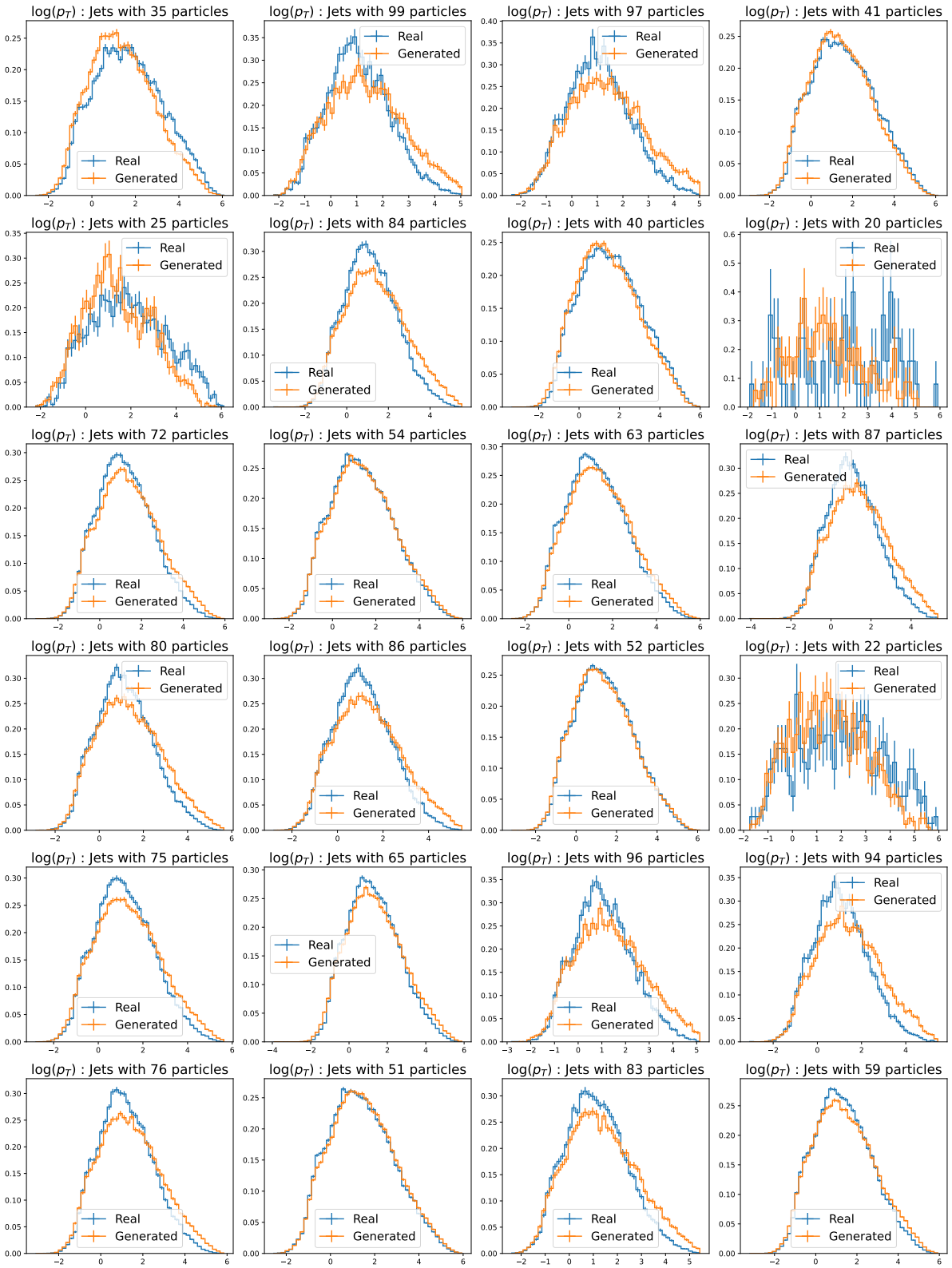


Figure 7.12: $\log p_T$ distribution for QCD jets of various sizes

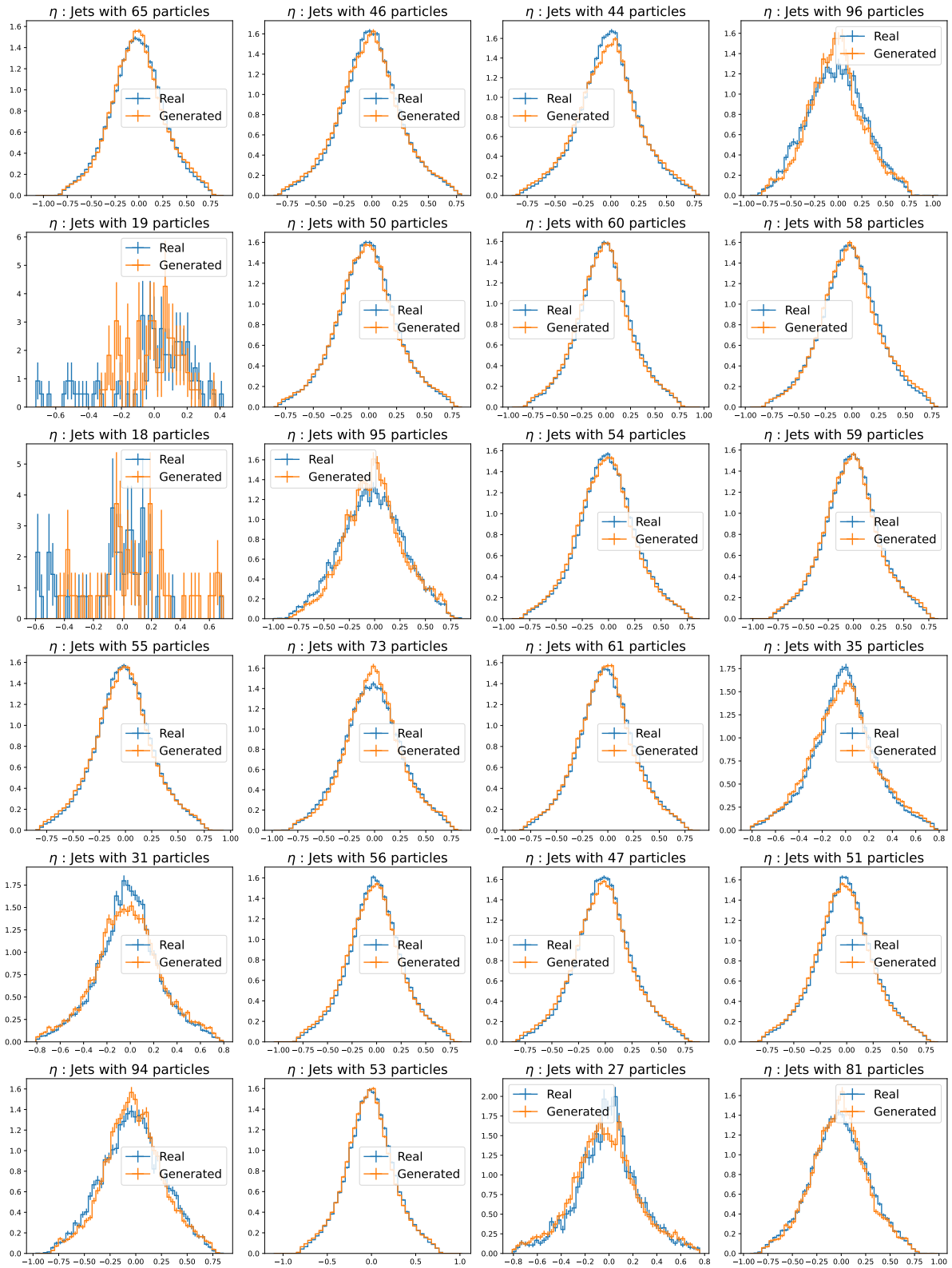


Figure 7.13: η distribution for QCD jets of various sizes

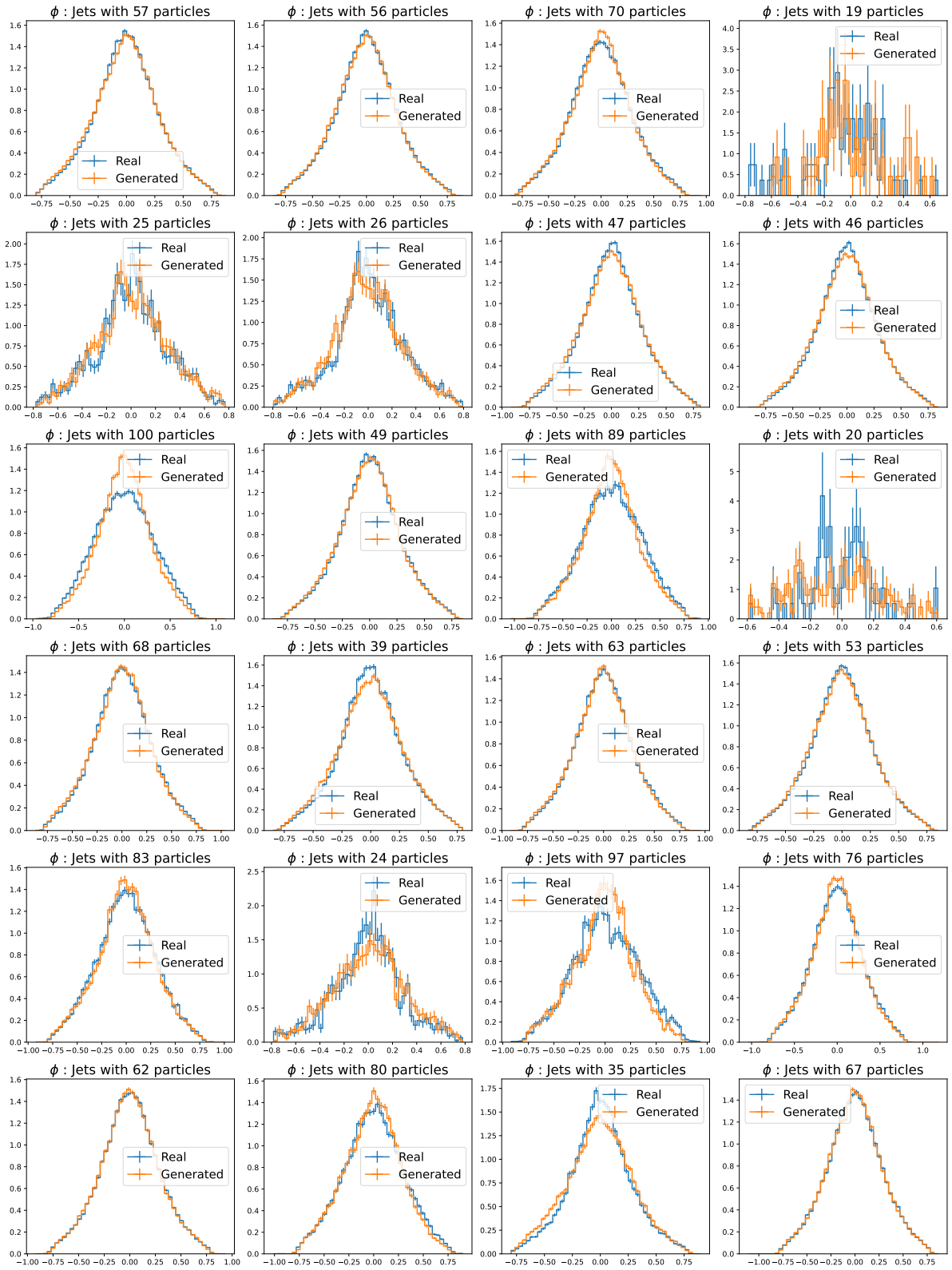


Figure 7.14: ϕ distribution for QCD jets of various sizes

Chapter *8*

Summary, Conclusions, and Future Outlook

We summarise the major results of this thesis below:

- In chapter 5, we performed a case study of applying deep learning in a new-physics search for a heavy B quark. We discussed the event-generation pipeline, the kinematic and topological cuts applied, and the choice of features reconstructed. We compared the performances of two ML approaches: a DNN model and a BDT model. At the benchmark point of $M_B = 1.5$ TeV and $M_\Phi = 0.4$ TeV, the analysis cuts only achieved a discovery sensitivity score of 0.38σ, while the BDT achieved a score of 2.76σ, and the DNN achieved a score of 3.81σ. The DNN outperformed the BDT by roughly $\sim 20\% - 40\%$ across all mass points. The study's result showed that, with our method, a B VLQ of mass up to 1.7 TeV could be discovered (5σ) at the HL-LHC. We then used Integrated Gradients to obtain global feature importances. We addressed the issue of an appropriate baseline by averaging the feature attributions over multiple baselines sampled from the background distribution to obtain a robust and better attribution score.

  For future work, we believe that newer neural network models which operate on raw, unstructured data, such as graph neural networks, will provide significant improvements. Furthermore, expanding interpretability methods to these architectures is a promising future direction. Finally, one can also think about the inverse problem of reconstructing meaningful features using the DNN feature attributions to recognize connections.

- In chapter 6, we built upon the modification to the loss introduced in chapter 5 and asked the general question — whether one could modify the loss function to align the DNN training with physics goals. We investigated two losses: (1) weighted cross entropy loss with weights calculated from theory, and (2) a theoretically derived surrogate loss for maximizing the discovery sensitivity, Eq. (3.28), based on the Lovasz extension. For the case of weighted losses, we experimented with different weights as functions of the cross-section of the processes. For the surrogate loss, we showed that, with slight modifications, equation (3.28) is submodular under the set of misclassifications $(\mathbf{n}, \mathbf{p})$, i.e., false negatives and

false positives. We used this property to construct the surrogate loss. We also showed that equation (3.25) is not submodular under the same set. We benchmarked the two losses and found that both perform better than the simple binary cross entropy loss in all the test cases. However, we found that both methods require some finetuning. Reducing the need for such finetuning would be an excellent future goal.

- Finally, in chapter 7, we addressed the problem of data generation, which is essential for collider projection studies, especially for those based on deep learning. Specifically, we developed a generative model based on the GAN framework to generate particle jets (as particle clouds). This generative model could enable faster and computationally cheaper generation for jet datasets. Jet generation poses a few problems that are not common in other domains, namely, the variability in the size of the jet and the permutation invariance of particle clouds. We proposed a two-stage pipeline with two generators and two discriminators trained on the LSGAN formulation to resolve these issues. We evaluated the trained generator on a few metrics designed to capture the quality of the generated data. We found that the generator performed well in producing jets that mimicked the IRC-safe observables. However, there is still scope for improvement, indicated by the relatively poor MMD and FPD values.

  For future work, we suspect improvements can come from using deeper networks for both the generator and the discriminator and training the models on a larger dataset. Future studies could also explore other formulations, such as the Wasserstein GAN formulation, but this formulation would require modifications for jet data. Other directions could also include introducing conditional variables and developing generative models using different kinds of generative models, such as VAEs, or diffusion models.

Though deep learning models are ubiquitous in particle physics searches, their inner workings are poorly understood. This thesis points out some ways to interpret these models better while proposing multiple avenues to investigate these models further. It also proposes novel methods to maximise the sensitivity of a particle physics experiment. It also explores the prospects of generative models for quick and scalable data generation.

Related Publications

1. Machine learning-enhanced search for a vectorlike-singlet **B** quark decaying to a singlet scalar or pseudoscalar

   Jai Bardhan, Tanumoy Mandal, Subhadip Mitra, Cyrin Neeraj

   (Published: *Physical Review* D **107**, 115001 (2023), https://doi.org/10.1103/PhysRevD.107.115001)

2. Targeted Losses for Deep Learning in Particle Physics

   Jai Bardhan, Cyrin Neeraj, Subhadip Mitra, Tanumoy Mandal

   (Under preparation)

3. A Simple Generative Adversarial Network for generating jets at LHC

   Jai Bardhan, Subhadip Mitra

   (Under preparation)

Bibliography

[1] **Particle Data Group** collaboration, R. L. Workman et al., *Review of Particle Physics*, *PTEP* **2022** (2022) 083C01.

[2] K. Olive, et al., the Particle Data Group and J. Rademacker, *Review of particle physics (2014)*, *Chinese Physics C* **38** (Aug., 2014) .

[3] G. Cowan, K. Cranmer, E. Gross and O. Vitells, *Asymptotic formulae for likelihood-based tests of new physics*, *Eur. Phys. J. C* **71** (2011) 1554, [1007.1727]. [Erratum: Eur.Phys.J.C 73, 2501 (2013)].

[4] Wikipedia contributors, "Collider — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Collider&oldid=1156154818, 2023.

[5] V. Shiltsev and F. Zimmermann, *Modern and future colliders*, *Rev. Mod. Phys.* **93** (Mar, 2021) 015006.

[6] A. D. Martin, *Proton structure, Partons, QCD, DGLAP and beyond*, *Acta Phys. Polon. B* **39** (2008) 2025–2062, [0802.0161].

[7] L. Lonnblad and H. Jung, *Monte Carlo generators and the CCFM equation*, in *8th International Workshop on Deep Inelastic Scattering and QCD (DIS 2000)*, pp. 297–300, 6, 2000. hep-ph/0006166.

[8] B. Anderson, *The Lund string model*. Adam Hilger Ltd, United Kingdom, 1985.

[9] D. Amati and G. Veneziano, *Preconfinement as a property of perturbative qcd*, *Physics Letters B* **83** (1979) 87–92.

[10] A. Bassetto, M. Ciafaloni and G. Marchesini, *Color singlet distributions and mass damping in perturbative qcd*, *Physics Letters B* **83** (1979) 207–212.

[11] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce et al., *Geant4—a simulation toolkit*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **506** (2003) 250–303.

[12] **DELPHES 3** collaboration, J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens et al., *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **02** (2014) 057, [1307.6346].

[13] J. Bauer, "Prospects for the Observation of Electroweak Top Quark Production with the CMS Experiment. Perspektiven zur Beobachtung der elektroschwachen Produktion einzelner Top-Quarks mit dem CMS Experiment." http://cds.cern.ch/record/1308713, 2010.

[14] CERN, "The large hadron collider." https://home.web.cern.ch/science/accelerators/large-hadron-collider.

[15] CERN, "How a detector works." https://home.cern/science/experiments/how-detector-works.

[16] CERN, "Cms detector." https://cms.cern/detector.

[17] R. Atkin, *Review of jet reconstruction algorithms*, *Journal of Physics: Conference Series* **645** (sep, 2015) 012008.

[18] S. Catani, Y. L. Dokshitzer, M. H. Seymour and B. R. Webber, *Longitudinally invariant K_t clustering algorithms for hadron hadron collisions*, *Nucl. Phys. B* **406** (1993) 187–224.

[19] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, *JHEP* **04** (2008) 063, [0802.1189].

[20] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer and T. Stelzer, *MadGraph 5: going beyond*, *Journal of High Energy Physics* **2011** (June, 2011) 128.

[21] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An introduction to pythia 8.2*, *Computer Physics Communications* **191** (2015) 159–177.

[22] M. Cacciari, G. P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J.* **C72** (2012) 1896, [1111.6097].

[23] M. Bähr, S. Gieseke, M. A. Gigg, D. Grellscheid, K. Hamilton, O. Latunde-Dada et al., *Herwig++ physics and manual*, *The European Physical Journal C* **58** (Dec., 2008) 639–707.

[24] E. Bothmann, G. S. Chahal, S. Höche, J. Krause, F. Krauss, S. Kuttimalai et al., *Event generation with Sherpa 2.2*, *SciPost Phys.* **7** (2019) 034.

[25] S. S. Wilks, *The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses*, *Annals Math. Statist.* **9** (1938) 60–62.

[26] A. Wald, *Tests of statistical hypotheses concerning several parameters when the number of observations is large*, *Transactions of the American Mathematical Society* **54** (1943) 426–482.

[27] A. G. Frodesen, O. Skjeggestad and H. Tofte, *Probability and Statistics in Particle Physics*. Universitetsforlaget, Bergen, Norway, 1979.

[28] F. James, *Statistical Methods in Experimental Physics*. WORLD SCIENTIFIC, 2nd ed., 2006, 10.1142/6096.

[29] J. Stupak, *A search for first generation leptoquarks in $\sqrt{s} = 7$ TeV pp collisions with the ATLAS detector*. PhD thesis, SUNY, Stony Brook, 8, 2012.

[30] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[31] HEP ML Community, "A Living Review of Machine Learning for Particle Physics." https://iml-wg.github.io/HEPML-LivingReview/.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair et al., *Generative adversarial nets*, in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[33] M. Arjovsky, S. Chintala and L. Bottou, *Wasserstein generative adversarial networks*, in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 214–223, PMLR, 06–11 Aug, 2017.

[34] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang and S. P. Smolley, *Least squares generative adversarial networks*, in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821, 2017. DOI.

[35] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. http://arxiv.org/abs/1312.6114v10.

[36] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1530–1538, JMLR.org, 2015.

[37] J. Ho, A. Jain and P. Abbeel, *Denoising diffusion probabilistic models*, in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds.), vol. 33, pp. 6840–6851, Curran Associates, Inc., 2020.

[38] Y. Xu, Z. Liu, M. Tegmark and T. Jaakkola, *Poisson flow generative models*, in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds.), vol. 35, pp. 16782–16795, Curran Associates, Inc., 2022.

[39] S. Bond-Taylor, A. Leach, Y. Long and C. G. Willcocks, *Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44** (2022) 7327–7347.

[40] L. Hyafil and R. L. Rivest, *Constructing optimal binary decision trees is np-complete*, *Information Processing Letters* **5** (1976) 15–17.

[41] K. Hornik, M. Stinchcombe and H. White, *Multilayer feedforward networks are universal approximators*, *Neural Networks* **2** (1989) 359–366.

[42] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 807–814, Omnipress, 2010.

[43] P. Ramachandran, B. Zoph and Q. V. Le, *Searching for activation functions*, *arXiv preprint arXiv:1710.05941* (2017) .

[44] D. Misra, *Mish: A self regularized non-monotonic activation function*, in *British Machine Vision Conference*, 2020.

[45] S. R. Dubey, S. K. Singh and B. B. Chaudhuri, *Activation functions in deep learning: A comprehensive survey and benchmark*, *Neurocomput.* **503** (sep, 2022) 92–108.

[46] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization.*, in *ICLR (Poster)* (Y. Bengio and Y. LeCun, eds.), 2015.

[47] 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* **15** (2014) 1929–1958.

[48] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 448–456, PMLR, 07–09 Jul, 2015.

[49] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, *Phys. Rev. D* **101** (2020) 056019, [1902.08570].

[50] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun, *Deep learning for 3d point clouds: A survey*, *IEEE transactions on pattern analysis and machine intelligence* **43** (2020) 4338–4364.

[51] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, *Neural message passing for quantum chemistry*, in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, PMLR, 06–11 Aug, 2017.

[52] J. You, Z. Ying and J. Leskovec, *Design space for graph neural networks*, in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds.), vol. 33, pp. 17009–17021, Curran Associates, Inc., 2020.

[53] S. Thais, P. Calafiura, G. Chachamis, G. DeZoort, J. Duarte, S. Ganguly et al., *Graph Neural Networks in Particle Physics: Implementations, Innovations, and Challenges*, in *Snowmass 2021*, 3, 2022. 2203.12852.

[54] L. Panizzi, *Vector-like quarks: t' and partners*, *Nuovo Cim. C* **037** (2014) 69–79.

[55] **CMS** collaboration, A. M. Sirunyan et al., *Search for pair production of vectorlike quarks in the fully hadronic final state*, *Phys. Rev. D* **100** (2019) 072001, [1906.11903].

[56] **CMS** collaboration, A. M. Sirunyan et al., *A search for bottom-type, vector-like quark pair production in a fully hadronic final state in proton-proton collisions at $\sqrt{s} = 13$ TeV*, *Phys. Rev. D* **102** (2020) 112004, [2008.09835].

[57] **ATLAS** collaboration, *Search for pair-production of vector-like quarks in pp collision events at $\sqrt{s} = 13$ TeV with at least one leptonically-decaying Z boson and a third-generation quark with the ATLAS detector*, ATLAS-CONF-2021-024, 6, 2021.

[58] **ATLAS** collaboration, *Search for single production of vector-like T quarks decaying to Ht or Zt in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, ATLAS-CONF-2021-040, 2021.

[59] **ATLAS** collaboration, G. Aad et al., *Search for single production of a vector-like T quark decaying into a Higgs boson and top quark with fully hadronic final states using the ATLAS detector*, 2201.07045.

[60] S. Gopalakrishna, T. Mandal, S. Mitra and G. Moreau, *LHC Signatures of Warped-space Vectorlike Quarks*, *JHEP* **08** (2014) 079, [1306.2656].

[61] S. Gopalakrishna, T. S. Mukherjee and S. Sadhukhan, *Extra neutral scalars with vectorlike fermions at the LHC*, *Phys. Rev. D* **93** (2016) 055004, [1504.01074].

[62] B. A. Dobrescu and F. Yu, *Exotic Signals of Vectorlike Quarks*, *J. Phys. G* **45** (2018) 08LT01, [1612.01909].

[63] J. A. Aguilar-Saavedra, D. E. López-Fogliani and C. Muñoz, *Novel signatures for vector-like quarks*, *JHEP* **06** (2017) 095, [1705.02526].

[64] H. Han, L. Huang, T. Ma, J. Shu, T. M. P. Tait and Y. Wu, *Six Top Messages of New Physics at the LHC*, *JHEP* **10** (2019) 008, [1812.11286].

[65] R. Benbrik et al., *Signatures of vector-like top partners decaying into new neutral scalar or pseudoscalar bosons*, *JHEP* **05** (2020) 028, [1907.05929].

[66] A. Bhardwaj, T. Mandal, S. Mitra and C. Neeraj, *Roadmap to explore vectorlike quarks decaying to a new scalar or pseudoscalar*, *Phys. Rev. D* **106** (2022) 095014, [2203.13753].

[67] G. Balossini, G. Montagna, C. M. Carloni Calame, M. Moretti, O. Nicrosini, F. Piccinini et al., *Combination of electroweak and QCD corrections to single W production at the Fermilab Tevatron and the CERN LHC*, *JHEP* **01** (2010) 013, [0907.0276].

[68] S. Catani, L. Cieri, G. Ferrera, D. de Florian and M. Grazzini, *Vector boson production at hadron colliders: A fully exclusive qcd calculation at next-to-next-to-leading order*, *Phys. Rev. Lett.* **103** (Aug, 2009) 082001.

[69] C. Muselli, M. Bonvini, S. Forte, S. Marzani and G. Ridolfi, *Top Quark Pair Production beyond NNLO*, *JHEP* **08** (2015) 076, [1505.02006].

[70] N. Kidonakis, *Theoretical results for electroweak-boson and single-top production*, *PoS* **DIS2015** (2015) 170, [1506.04072].

[71] J. M. Campbell, R. K. Ellis and C. Williams, *Vector boson pair production at the LHC*, *JHEP* **07** (2011) 018, [1105.0020].

[72] A. Kulesza, L. Motyka, D. Schwartländer, T. Stebel and V. Theeuwes, *Associated production of a top quark pair with a heavy electroweak gauge boson at NLO+NNLL accuracy*, *Eur. Phys. J. C* **79** (2019) 249, [1812.08622].

[73] **LHC Higgs Cross Section Working Group** collaboration, D. de Florian et al., *Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector*, 1610.07922.

[74] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr and B. Fuks, *FeynRules 2.0 - A complete toolbox for tree-level phenomenology*, *Comput. Phys. Commun.* **185** (2014) 2250–2300, [1310.1921].

[75] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer and T. Reiter, *UFO - The Universal FeynRules Output*, *Comput. Phys. Commun.* **183** (2012) 1201–1214, [1108.2040].

[76] **ATLAS** collaboration, G. Aad et al., *Electron and photon performance measurements with the ATLAS detector using the 2015–2017 LHC proton-proton collision data*, *JINST* **14** (2019) P12006, [1908.00005].

[77] **CMS** collaboration, A. M. Sirunyan et al., *Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV*, *JINST* **13** (2018) P05011, [1712.07158].

[78] J. Bardhan, T. Mandal, S. Mitra and C. Neeraj, *Machine learning-enhanced search for a vectorlike singlet b quark decaying to a singlet scalar or pseudoscalar*, *Phys. Rev. D* **107** (Jun, 2023) 115001.

[79] T. Chen and C. Guestrin, *XGBoost: A scalable tree boosting system*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016. DOI.

[80] Z. Allen-Zhu, Y. Li and Y. Liang, *Learning and generalization in overparameterized neural networks, going beyond two layers*, in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[81] M. Sundararajan, A. Taly and Q. Yan, *Axiomatic attribution for deep networks*, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 3319–3328, JMLR.org, 2017.

[82] P. Sturmfels, S. Lundberg and S.-I. Lee, *Visualizing the impact of feature attribution baselines*, *Distill* (2020) . https://distill.pub/2020/attribution-baselines.

[83] D. Smilkov, N. Thorat, B. Kim, F. Viégas and M. Wattenberg, *Smoothgrad: removing noise by adding noise*, 2017. 10.48550/ARXIV.1706.03825.

[84] L. Lovász, *Submodular functions and convexity*, pp. 235–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. 10.1007/978-3-642-68874-4_10.

[85] M. Berman and M. B. Blaschko, *Optimization of the jaccard index for image segmentation with the lovász hinge*, *CoRR* **abs/1705.08790** (2017) , [1705.08790].

[86] J. Yu and M. Blaschko, *The lovász hinge: A novel convex surrogate for submodular losses*, 2015. 10.48550/ARXIV.1512.07797.

[87] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, *Focal loss for dense object detection*, in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[88] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, in *International Conference on Learning Representations*, 2017.

[89] Y. S. Aurelio, G. M. de Almeida, C. L. de Castro and A. P. Braga, *Learning from imbalanced data sets with weighted cross-entropy function*, *Neural Processing Letters* **50** (2019) 1937–1949.

[90] Z. Zhou, H. Huang and B. Fang, *Application of weighted cross-entropy loss function in intrusion detection*, *Journal of Computer and Communications* **09** (2021) 1–21.

[91] D. Zhang, Y. Fang and H. Tang, *Research on naive Bayes algorithm based on feature weighted*, in *International Conference on Electronic Information Engineering and Computer Technology (EIECT 2021)* (S. Verma and N. Rajathi, eds.), vol. 12087, p. 120871A, International Society for Optics and Photonics, SPIE, 2021. DOI.

[92] J. Brehmer, K. Cranmer, G. Louppe and J. Pavez, *A Guide to Constraining Effective Field Theories with Machine Learning*, *Phys. Rev. D* **98** (2018) 052004, [1805.00020].

[93] J. Brehmer, G. Louppe, J. Pavez and K. Cranmer, *Mining gold from implicit models to improve likelihood-free inference*, *Proc. Nat. Acad. Sci.* **117** (2020) 5242–5249, [1805.12244].

[94] B. Nachman, *A guide for deploying Deep Learning in LHC searches: How to achieve optimality and account for uncertainty*, *SciPost Phys.* **8** (2020) 090, [1909.03081].

[95] J. Shelton, *Jet Substructure*, in *Theoretical Advanced Study Institute in Elementary Particle Physics: Searching for New Physics at Small and Large Scales*, pp. 303–340, 2013. 1302.0260. DOI.

[96] C. J. Maddison, A. Mnih and Y. W. Teh, *The concrete distribution: A continuous relaxation of discrete random variables*, in *International Conference on Learning Representations*, 2017.

[97] E. Jang, S. Gu and B. Poole, *Categorical reparameterization with gumbel-softmax*, in *International Conference on Learning Representations*, 2017.

[98] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi and Y. W. Teh, *Set transformer: A framework for attention-based permutation-invariant neural networks*, in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, PMLR, 09–15 Jun, 2019.

[99] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez et al., *Attention is all you need*, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

[100] C. Joshi, *Transformers are graph neural networks*, *The Gradient* **12** (2020) .

[101] T. Lin, Y. Wang, X. Liu and X. Qiu, *A survey of transformers*, *AI Open* (2022) .

[102] B. Xu, N. Wang, T. Chen and M. Li, *Empirical evaluation of rectified activations in convolutional network*, *arXiv preprint arXiv:1505.00853* (2015) .

[103] J. L. Ba, J. R. Kiros and G. E. Hinton, *Layer normalization*, 2016.

[104] G. Kasieczka, T. Plehn, J. Thompson and M. Russel, *Top quark tagging reference dataset*, Mar., 2019. 10.5281/zenodo.2603256.

[105] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida, *Spectral normalization for generative adversarial networks*, 2018.

[106] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan et al., *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[107] M. Fey and J. E. Lenssen, *Fast graph representation learning with PyTorch Geometric*, in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[108] W. Falcon et al., *Pytorch lightning*, *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* **3** (2019) .

[109] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari et al., *Evaluating generative models in high energy physics*, *Phys. Rev. D* **107** (2023) 076017, [2211.10295].

[110] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan et al., eds.), vol. 30, Curran Associates, Inc., 2017.

[111] P. T. Komiske, E. M. Metodiev and J. Thaler, *Metric space of collider events*, *Physical Review Letters* **123** (Jul, 2019) .

[112] P. T. Komiske, E. M. Metodiev and J. Thaler, *Energy flow polynomials: a complete linear basis for jet substructure*, *Journal of High Energy Physics* **2018** (Apr, 2018) .

[113] A. Butter, T. Plehn and R. Winterhalder, *How to gan lhc events*, *SciPost Physics* **7** (Dec, 2019) .

[114] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol et al., *Getting high: High fidelity simulation of high granularity calorimeters with high speed*, *Computing and Software for Big Science* **5** (May, 2021) .

[115] P. Musella and F. Pandolfi, *Fast and accurate simulation of particle detectors using generative adversarial networks*, *Computing and Software for Big Science* **2** (Nov, 2018) .

[116] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini et al., *Particle Cloud Generation with Message Passing Generative Adversarial Networks*, in *35th Conference on Neural Information Processing Systems*, 6, 2021. 2106.11535.

[117] B. Käch, D. Krücker, I. Melzer-Pellmann, M. Scham, S. Schnake and A. Verney-Provatas, *JetFlow: Generating Jets with Conditioned and Mass Constrained Normalising Flows*, 2211.13630.

[118] V. Mikuni, B. Nachman and M. Pettee, *Fast Point Cloud Generation with Diffusion Models in High Energy Physics*, 2304.01266.